Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore for additional books and resources.
Chapter 29
The CALIS Procedure

Contents

Overview: CALIS Procedure .......................................................... 1158
  Compatibility with the CALIS Procedure in SAS/STAT 9.2 or Earlier ........ 1162
  Compatibility with the TCALIS Procedure in SAS/STAT 9.2 ................. 1169
  A Guide to the PROC CALIS Documentation .................................. 1174
Getting Started: CALIS Procedure ................................................. 1184
  A Structural Equation Example .................................................... 1184
  A Factor Model Example ............................................................ 1190
  Direct Covariance Structures Analysis ........................................ 1192
  Which Modeling Language? .......................................................... 1194
Syntax: CALIS Procedure .............................................................. 1195
  Classes of Statements in PROC CALIS ......................................... 1196
  Single-Group Analysis Syntax ...................................................... 1199
  Multiple-Group Multiple-Model Analysis Syntax ................................ 1199
  PROC CALIS Statement .............................................................. 1200
  BOUNDS Statement .................................................................... 1241
  BY Statement ........................................................................... 1241
  COSAN Statement ..................................................................... 1242
  COV Statement .......................................................................... 1252
  DETERM Statement ................................................................. 1256
  EFFPART Statement .................................................................. 1257
  FACTOR Statement ................................................................... 1258
  FITINDEX Statement .................................................................. 1268
  FREQ Statement ....................................................................... 1271
  GROUP Statement ....................................................................... 1272
  LINCON Statement ................................................................. 1274
  LINEQS Statement ................................................................. 1275
  LISMOD Statement ................................................................. 1282
  LMTESTS Statement .............................................................. 1285
  MATRIX Statement ................................................................. 1295
  MEAN Statement ....................................................................... 1309
  MODEL Statement ................................................................. 1311
  MSTRUCT Statement .............................................................. 1313
  NLINCON Statement .............................................................. 1316
  NLOPTIONS Statement ............................................................ 1316
  OUTFILES Statement .............................................................. 1317
  PARAMETERS Statement .......................................................... 1319
<table>
<thead>
<tr>
<th>SAS Procedure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARTIAL Statement</td>
<td>1322</td>
</tr>
<tr>
<td>PATH Statement</td>
<td>1322</td>
</tr>
<tr>
<td>PATHDIAGRAM Statement</td>
<td>1332</td>
</tr>
<tr>
<td>PCOV Statement</td>
<td>1343</td>
</tr>
<tr>
<td>PVAR Statement</td>
<td>1345</td>
</tr>
<tr>
<td>RAM Statement</td>
<td>1348</td>
</tr>
<tr>
<td>REFMODEL Statement</td>
<td>1354</td>
</tr>
<tr>
<td>RENAMEPARM Statement</td>
<td>1356</td>
</tr>
<tr>
<td>SAS Programming Statements</td>
<td>1357</td>
</tr>
<tr>
<td>SIMTESTS Statement</td>
<td>1357</td>
</tr>
<tr>
<td>STD Statement</td>
<td>1358</td>
</tr>
<tr>
<td>STRUCTEQ Statement</td>
<td>1358</td>
</tr>
<tr>
<td>TESTFUNC Statement</td>
<td>1358</td>
</tr>
<tr>
<td>VAR Statement</td>
<td>1359</td>
</tr>
<tr>
<td>VARIANCE Statement</td>
<td>1362</td>
</tr>
<tr>
<td>VARNAMES Statement</td>
<td>1366</td>
</tr>
<tr>
<td>WEIGHT Statement</td>
<td>1367</td>
</tr>
</tbody>
</table>

**Details: CALIS Procedure**

- Input Data Sets ............................................ 1367
- Output Data Sets ............................................ 1371
- Default Analysis Type and Default Parameterization .... 1388
- The COSAN Model ............................................ 1390
- The FACTOR Model .......................................... 1394
- The LINEQS Model ......................................... 1401
- The LISMOD Model and Submodels .......................... 1408
- The MSTRUCT Model ....................................... 1416
- The PATH Model ............................................ 1419
- The RAM Model ............................................. 1425
- Naming Variables and Parameters ....................... 1433
- Setting Constraints on Parameters ...................... 1434
- Automatic Variable Selection ............................ 1439
- Path Diagrams: Layout Algorithms, Default Settings, and Customization 1440
- Estimation Criteria ....................................... 1463
- Relationships among Estimation Criteria ............... 1473
- Gradient, Hessian, Information Matrix, and Approximate Standard Errors 1475
- Counting the Degrees of Freedom ....................... 1478
- Assessment of Fit ......................................... 1481
- Case-Level Residuals, Outliers, Leverage Observations, and Residual Diagnostics 1494
- Total, Direct, and Indirect Effects ................... 1498
- Standardized Solutions .................................. 1500
- Modification Indices ..................................... 1501
- Missing Values and the Analysis of Missing Patterns 1503
- Measures of Multivariate Kurtosis ..................... 1504
- Initial Estimates ......................................... 1506
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use of Optimization Techniques</td>
<td>1507</td>
</tr>
<tr>
<td>Computational Problems</td>
<td>1515</td>
</tr>
<tr>
<td>Displayed Output</td>
<td>1518</td>
</tr>
<tr>
<td>ODS Table Names</td>
<td>1523</td>
</tr>
<tr>
<td>ODS Graphics</td>
<td>1538</td>
</tr>
<tr>
<td>Examples: CALIS Procedure</td>
<td>1539</td>
</tr>
<tr>
<td>Example 29.1: Estimating Covariances and Correlations</td>
<td>1539</td>
</tr>
<tr>
<td>Example 29.2: Estimating Covariances and Means Simultaneously</td>
<td>1544</td>
</tr>
<tr>
<td>Example 29.3: Testing Uncorrelatedness of Variables</td>
<td>1545</td>
</tr>
<tr>
<td>Example 29.4: Testing Covariance Patterns</td>
<td>1547</td>
</tr>
<tr>
<td>Example 29.5: Testing Some Standard Covariance Pattern Hypotheses</td>
<td>1550</td>
</tr>
<tr>
<td>Example 29.6: Linear Regression Model</td>
<td>1553</td>
</tr>
<tr>
<td>Example 29.7: Multivariate Regression Models</td>
<td>1557</td>
</tr>
<tr>
<td>Example 29.8: Measurement Error Models</td>
<td>1574</td>
</tr>
<tr>
<td>Example 29.9: Testing Specific Measurement Error Models</td>
<td>1580</td>
</tr>
<tr>
<td>Example 29.10: Measurement Error Models with Multiple Predictors</td>
<td>1586</td>
</tr>
<tr>
<td>Example 29.11: Measurement Error Models Specified As Linear Equations</td>
<td>1591</td>
</tr>
<tr>
<td>Example 29.12: Confirmatory Factor Models</td>
<td>1597</td>
</tr>
<tr>
<td>Example 29.13: Confirmatory Factor Models: Some Variations</td>
<td>1608</td>
</tr>
<tr>
<td>Example 29.14: Residual Diagnostics and Robust Estimation</td>
<td>1616</td>
</tr>
<tr>
<td>Example 29.15: The Full Information Maximum Likelihood Method</td>
<td>1629</td>
</tr>
<tr>
<td>Example 29.16: Comparing the ML and FIML Estimation</td>
<td>1639</td>
</tr>
<tr>
<td>Example 29.17: Path Analysis: Stability of Alienation</td>
<td>1644</td>
</tr>
<tr>
<td>Example 29.18: Simultaneous Equations with Mean Structures and Reciprocal Paths</td>
<td>1658</td>
</tr>
<tr>
<td>Example 29.19: Fitting Direct Covariance Structures</td>
<td>1665</td>
</tr>
<tr>
<td>Example 29.20: Confirmatory Factor Analysis: Cognitive Abilities</td>
<td>1668</td>
</tr>
<tr>
<td>Example 29.21: Testing Equality of Two Covariance Matrices Using a Multiple-Group Analysis</td>
<td>1680</td>
</tr>
<tr>
<td>Example 29.22: Testing Equality of Covariance and Mean Matrices between Independent Groups</td>
<td>1685</td>
</tr>
<tr>
<td>Example 29.23: Illustrating Various General Modeling Languages</td>
<td>1708</td>
</tr>
<tr>
<td>Example 29.24: Testing Competing Path Models for the Career Aspiration Data</td>
<td>1718</td>
</tr>
<tr>
<td>Example 29.25: Fitting a Latent Growth Curve Model</td>
<td>1730</td>
</tr>
<tr>
<td>Example 29.26: Higher-Order and Hierarchical Factor Models</td>
<td>1736</td>
</tr>
<tr>
<td>Example 29.27: Linear Relations among Factor Loadings</td>
<td>1750</td>
</tr>
<tr>
<td>Example 29.28: Multiple-Group Model for Purchasing Behavior</td>
<td>1759</td>
</tr>
<tr>
<td>Example 29.29: Fitting the RAM and EQS Models by the COSAN Modeling Language</td>
<td>1784</td>
</tr>
<tr>
<td>Example 29.30: Second-Order Confirmatory Factor Analysis</td>
<td>1799</td>
</tr>
<tr>
<td>Example 29.31: Linear Relations among Factor Loadings: COSAN Model Specification</td>
<td>1804</td>
</tr>
<tr>
<td>Example 29.32: Ordinal Relations among Factor Loadings</td>
<td>1809</td>
</tr>
<tr>
<td>Example 29.33: Longitudinal Factor Analysis</td>
<td>1812</td>
</tr>
<tr>
<td>References</td>
<td>1819</td>
</tr>
</tbody>
</table>
Overview: CALIS Procedure

Structural equation modeling is an important statistical tool in social and behavioral sciences. Structural equations express relationships among a system of variables that can be either observed variables (manifest variables) or unobserved hypothetical variables (latent variables). For an introduction to latent variable models, see Loehlin (2004); Bollen (1989b); Everitt (1984), or Long (1983); and for manifest variables with measurement errors, see Fuller (1987).

In structural models, as opposed to functional models, all variables are taken to be random rather than having fixed levels. For maximum likelihood (default) and generalized least squares estimation in PROC CALIS, the random variables are assumed to have an approximately multivariate normal distribution. Nonnormality, especially high kurtosis, can produce poor estimates and grossly incorrect standard errors and hypothesis tests, even in large samples. Consequently, the assumption of normality is much more important than in models with nonstochastic exogenous variables. You should remove outliers and consider transformations of nonnormal variables before using PROC CALIS with maximum likelihood (default) or generalized least squares estimation. Alternatively, model outliers can be downweighted during model estimation with robust methods. If the number of observations is sufficiently large, Browne’s asymptotically distribution-free (ADF) estimation method can be used. If your data sets contain random missing data, the full information maximum likelihood (FIML) method can be used.

You can use the CALIS procedure to estimate parameters and test hypotheses for constrained and unconstrained problems in various situations, including but not limited to the following:

- exploratory and confirmatory factor analysis of any order
- linear measurement-error models or regression with errors in variables
- multiple and multivariate linear regression
- multiple-group structural equation modeling with mean and covariance structures
- path analysis and causal modeling
- simultaneous equation models with reciprocal causation
- structured covariance and mean matrices in various forms

To specify models in PROC CALIS, you can use a variety of modeling languages:

- **COSAN**—a generalized version of the COSAN program (McDonald 1978, 1980), uses general mean and covariance structures to define models
- **FACTOR**—supports the input of latent factor and observed variable relations
- **LINEQS**—like the EQS program (Bentler 1995), uses equations to describe variable relationships
- **LISMOD**—utilizes LISREL (Jöreskog and Sörbom 1985) model matrices to define models
- **MSTRUCT**—supports direct parameterizations in the mean and covariance matrices
• **PATH**—provides an intuitive causal path specification interface

• **RAM**—utilizes the formulation of the reticular action model (McArdle and McDonald 1984) to define models

• **REFMODEL**—provides a quick way for model referencing and respecification

Various modeling languages are provided to suit a wide range of researchers’ background and modeling philosophy. However, statistical situations might arise where one modeling language is more convenient than the others. This will be discussed in the section “Which Modeling Language?” on page 1194.

In addition to basic model specification, you can set various parameter constraints in PROC CALIS. Equality constraints on parameters can be achieved by simply giving the same parameter names in different parts of the model. **Boundary**, **linear**, and **nonlinear** constraints are supported as well. If parameters in the model are dependent on additional parameters, you can define the dependence by using the **PARAMETERS** and the **SAS programming statements**.

Before the data are analyzed, researchers might be interested in studying some statistical properties of the data. PROC CALIS can provide the following statistical summary of the data:

• covariance and mean matrices and their properties

• descriptive statistics like means, standard deviations, univariate skewness, and kurtosis measures

• multivariate measures of kurtosis

• coverage of covariances and means, missing patterns summary, and means of the missing patterns when the FIML estimation is used

• weight matrix and its descriptive properties

• robust covariance and mean matrices with the robust methods

After a model is fitted and accepted by the researcher, PROC CALIS can provide the following supplementary statistical analysis:

• computing squared multiple correlations and determination coefficients

• direct and indirect effects partitioning with standard error estimates

• model modification tests such as Lagrange multiplier and Wald tests

• computing fit summary indices

• computing predicted moments of the model

• residual analysis on the covariances and means

• case-level residual diagnostics with graphical plots

• factor rotations

• standardized solutions with standard errors

• testing parametric functions, individually or simultaneously
When fitting a model, you need to choose an estimation method. The following estimation methods are supported in the CALIS procedure:

- diagonally weighted least squares (DWLS, with optional weight matrix input)
- full information maximum likelihood (FIML, which can treat observations with random missing values)
- generalized least squares (GLS, with optional weight matrix input)
- maximum likelihood (ML, for multivariate normal data); this is the default method
- robust estimation with maximum likelihood model evaluation (ROBUST option)
- unweighted least squares (ULS)
- weighted least squares or asymptotically distribution-free method (WLS or ADF, with optional weight matrix input)

Estimation methods implemented in PROC CALIS do not exhaust all alternatives in the field. For example, the partial least squares (PLS) method is not implemented. See the section “Estimation Criteria” on page 1463 for details about estimation criteria used in PROC CALIS. Note that there is a SAS/STAT procedure called PROC PLS, which employs the partial least squares technique but for a different class of models than those of PROC CALIS. For general path analysis with latent variables, consider using PROC CALIS.

All estimation methods need some starting values for the parameter estimates. You can provide starting values for any parameters. If there is any estimate without a starting value provided, PROC CALIS determines the starting value by using one or any combination of the following methods:

- approximate factor analysis
- default initial values
- instrumental variable method
- matching observed moments of exogenous variables
- McDonald’s method (McDonald and Hartmann 1992) method
- ordinary least squares estimation
- random number generation, if a seed is provided
- two-stage least squares estimation

Although no methods for initial estimates are completely foolproof, the initial estimation methods provided by PROC CALIS behave reasonably well in most common applications.

With initial estimates, PROC CALIS will iterate the solutions so as to achieve the optimum solution as defined by the estimation criterion. This is a process known as optimization. Because numerical problems can occur in any optimization process, the CALIS procedure offers several optimization algorithms so that you can choose alternative algorithms when the one being used fails. The following optimization algorithms are supported in PROC CALIS:
• Levenberg-Marquardt algorithm (Moré 1978)
• trust-region algorithm (Gay 1983)
• Newton-Raphson algorithm with line search
• ridge-stabilized Newton-Raphson algorithm
• various quasi-Newton and dual quasi-Newton algorithms: Broyden-Fletcher-Goldfarb-Shanno and Davidon-Fletcher-Powell, including a sequential quadratic programming algorithm for processing nonlinear equality and inequality constraints
• various conjugate gradient algorithms: automatic restart algorithm of Powell (1977), Fletcher-Reeves, Polak-Ribiere, and conjugate descent algorithm of Fletcher (1980)
• iteratively reweighted least squares for robust estimation

In addition to the ability to save output tables as data sets by using the ODS OUTPUT statement, PROC CALIS supports the following types of output data sets so that you can save your analysis results for later use:

• OUTEST= data sets for storing parameter estimates and their covariance estimates
• OUTFIT= data sets for storing fit indices and some pertinent modeling information
• OUTMODEL= data sets for storing model specifications and final estimates
• OUTSTAT= data sets for storing descriptive statistics, robust covariances and means, residuals, predicted moments, and latent variable scores regression coefficients
• OUTWGT= data sets for storing the weight matrices used in the modeling

The OUTEST=, OUTMODEL=, and OUTWGT= data sets can be used as input data sets for subsequent analyses. That is, in addition to the input data provided by the DATA= option, PROC CALIS supports the following input data sets for various purposes in the analysis:

• INEST= data sets for providing initial parameter estimates. An INEST= data set could be an OUTEST= data set created from a previous analysis.
• INMODEL= data sets for providing model specifications and initial estimates. An INMODEL= data set could be an OUTMODEL= data set created from a previous analysis.
• INWGT= data sets for providing the weight matrices. An INWGT= data set could be an OUTWGT= data set created from a previous analysis.

The CALIS procedure uses ODS Graphics to create high-quality graphs as part of its output. You can produce the following graphical output by specifying the PLOTS= option or the PATHDIAGRAM statement:

• histogram for mean, covariance, or correlation residuals
• histogram for case-level residual M-distances
Chapter 29: The CALIS Procedure

- case-level residual diagnostic plots such as residual by leverage plot, residual by predicted plot, PP-plot, and QQ-plot

- path diagram for initial model specification, unstandardized solution, or standardized solution

See Chapter 21, “Statistical Graphics Using ODS,” for general information about ODS Graphics. See the section “ODS Graphics” on page 1538 and the PLOTS= option on page 1232 for specific information about the statistical graphics available with the CALIS procedure. For more information about producing customized path diagrams, see the options of the PATHDIAGRAM statement.

Compatibility with the CALIS Procedure in SAS/STAT 9.2 or Earlier

In addition to the many important feature enhancements of the CALIS procedure since SAS/STAT 9.22, there have also been some rudimentary changes in the procedure. To help users make a smoother transition from earlier versions (SAS/STAT 9.2 or earlier), this section describes some of the major changes of the CALIS procedure since SAS/STAT 9.22.

Changes in Default Analysis Type and Parameterization

Table 29.1 lists some important changes in the default analysis type and parameterization since SAS/STAT 9.22. Some items that did not change are also included to make the scope of the changes clear. Notice that the part of this table about default parameterization applies only to models that have functional relations between variables. This class of functional models includes the following types of models: FACTOR, LINEQS, LISMOD, PATH, and RAM, although LISMOD and PATH models did not exist prior to SAS/STAT 9.22. This table does not apply to the default parameterization of the COSAN and MSTRUCT models. For these models, see the descriptions in the COSAN and MSTRUCT statements, or see the sections “The MSTRUCT Model” on page 1416 and “The COSAN Model” on page 1390.

Table 29.1 Default Analysis Type and Parameterization

<table>
<thead>
<tr>
<th>Default Setting</th>
<th>SAS/STAT 9.22 or Later</th>
<th>Prior to SAS/STAT 9.22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis type</td>
<td>Covariance analysis</td>
<td>Correlation analysis</td>
</tr>
<tr>
<td>Variances of independent factors and errors</td>
<td>Free parameters</td>
<td>Zero variances</td>
</tr>
<tr>
<td>Variances of independent observed variables</td>
<td>Free parameters</td>
<td>Free parameters</td>
</tr>
<tr>
<td>Covariances between independent factors</td>
<td>Free parameters¹</td>
<td>Fixed zeros</td>
</tr>
<tr>
<td>Covariances between error variables</td>
<td>Fixed zeros</td>
<td>Fixed zeros</td>
</tr>
<tr>
<td>Covariances between independent observed variables</td>
<td>Free parameters</td>
<td>Free parameters</td>
</tr>
</tbody>
</table>

1. The exploratory FACTOR model is an exception. Covariances between unrotated factors are set to zeros by default.
• Covariance structure analysis is the default analysis type in SAS/STAT 9.22 or later.

Covariance structure analysis has been the default since SAS/STAT 9.22. The statistical theory for structural equation modeling has been developed largely for covariance structures rather than correlation structures. Also, most practical structural equation models nowadays are concerned with covariance structures. Therefore, the default analysis type with covariance structure analysis is more reasonable. You must now use the CORR option for correlation structure analysis.

• Variances of any types of variables are free parameters by default in SAS/STAT 9.22 or later.

Variances of any types of variables are assumed to be free parameters in almost all applications for functional models. Prior to SAS/STAT 9.22, default variances of independent factors and errors were set to fixed zeros. This default has changed since SAS/STAT 9.22. Variances of any types of variables in functional models are now free parameters by default. This eliminates the need to specify these commonly assumed variance parameters.

• Covariances between all exogenous variables (factors or observed variables), except for error variables, are free parameters by default in functional models in SAS/STAT 9.22 or later.

Since SAS/STAT 9.22, covariances between all exogenous variables, except for error variables, are free parameters by default in functional models such as LINEQS, LISMOD, PATH, RAM, and confirmatory FACTOR models. In exploratory FACTOR models, covariances between unrotated factors are set to zeros by default. This change of default setting reflects the structural equation modeling practice much better. The default covariances between error variables, or between errors and other exogenous variables, are fixed at zero in all versions. Also, the default covariances between independent observed variables are free parameters in all versions.

Certainly, you can override all default parameters by specifying various statements in SAS/STAT 9.22 or later. You can use the PVAR, RAM, VARIANCE, or specific MATRIX statement to override default fixed or free variance parameters. You can use the COV, PCOV, RAM, or specific MATRIX statement to override default fixed or free covariance parameters.

Changes in the Role of the VAR Statement in Model Specification

Like many other SAS procedures, PROC CALIS enables you to use the VAR statement to define the set of observed variables in the data set for analysis. Unlike many other SAS procedures, PROC CALIS has other statements for more flexible model specifications. Because you can specify observed variables in these model specification statements and in the VAR statement with PROC CALIS, a question might arise when the set of observed variables specified in the VAR statement is not exactly the same as the set of observed variables specified in the model specification statements. Which set of observed variables does PROC CALIS use for analysis? The answer might depend on which version of PROC CALIS you use.

For observed variables that are specified in both the VAR statement and at least one of the model specification statements (such as the LINEQS, PATH, RAM, COV, PCOV, PVAR, STD, and VARIANCE statements), PROC CALIS recognizes the observed variables without any difficulty (that is, if they actually exist in the data set) no matter which SAS version you are using.
For observed variables that are specified only in the model specification statements and not in the VAR statement specifications (if the VAR statement is used at all), PROC CALIS does not recognize the observed variables as they are because the VAR statement has been used to preselect the legitimate set of observed variables to analyze. In most models, these observed variables are instead treated as latent variables. Again, this behavior is the same for all versions of PROC CALIS. If mistreating observed variables as latent variables is a major concern, a simple solution is not to use the VAR statement at all. This ensures that PROC CALIS uses all the observed variable as they are if they are specified in the model specification statements.

Finally, for observed variables that are specified only in the VAR statement and not in any of the model specification statements, the behavior depends on which SAS version you are using. Prior to SAS/STAT 9.22, PROC CALIS simply ignored these observed variables. For SAS/STAT 9.22 or later, PROC CALIS still includes these observed variables for analysis. In this case, PROC CALIS treats these “extra” variables as exogenous variables in the model so that some default parameterization applies. See the section “Default Analysis Type and Default Parameterization” on page 1388 for explanations of why this could be useful.

Changes in the LINEQS Model Specifications

Prior to SAS/STAT 9.22, LINEQS was the most popular modeling language supported by PROC CALIS. Since then, PROC CALIS has implemented a new syntax system that does not require the use of parameter names. Together with the change in default parameterization and some basic modeling methods, the specification of LINEQS models becomes much simpler and intuitive in SAS/STAT 9.22 or later. Table 29.2 lists the major syntax changes in LINEQS model specifications, followed by some notes.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SAS/STAT 9.22 or Later</th>
<th>Prior to SAS/STAT 9.22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free parameter specifications</td>
<td>Parameter names optional</td>
<td>Parameter names required</td>
</tr>
<tr>
<td>Error terms in equations</td>
<td>Required</td>
<td>Not required</td>
</tr>
<tr>
<td>Mean structure analysis</td>
<td>With the MEANSTR option</td>
<td>With the UCOV and AUG options</td>
</tr>
<tr>
<td>Intercept parameter specifications</td>
<td>With the Intercept or Intercep variable</td>
<td>With the INTERCEP variable</td>
</tr>
<tr>
<td>Mean parameter specifications</td>
<td>With the MEAN statement specifications</td>
<td>As covariances between variables and the INTERCEP variable</td>
</tr>
<tr>
<td>Treatment of short parameter lists</td>
<td>Free parameters after the last parameter specification</td>
<td>Replicating the last parameter specification</td>
</tr>
<tr>
<td>Parameter-prefix notation</td>
<td>With two trailing underscores (__) as suffix</td>
<td>With the suffix ‘:’</td>
</tr>
</tbody>
</table>

Table 29.2  Changes in the LINEQS Model Syntax
The explanations of these changes are as follows:

- The use of parameter names for free parameters is optional in SAS/STAT 9.22 or later.

Prior to SAS/STAT 9.22, you must use parameter names to specify free parameters. In SAS/STAT 9.22 or later, the use of parameter names is optional. For example, prior to SAS/STAT 9.22, you might use the following LINEQS model specifications:

```sas
lineqs
   A = x1 * B + x2(.5) * C + E1;
std
   B = var_b, C = var_c(1.2);
cox
   B C = cov_b_c;
```

In SAS/STAT 9.22 or later, you can simply use the following:

```sas
lineqs
   A = * B + (.5) * C + E1;
var
   B, C = (1.2);
cox
   B C;
```

This example shows that the parameter names x1, x2, var_b, var_c, and cov_b_c for free parameters are not necessary in SAS/STAT 9.22 or later. PROC CALIS generates names for these unnamed free parameters automatically. Also, you can specify initial estimates in parentheses without using parameter names. Certainly, you can still use parameter names wherever you want to, especially when you need to constrain parameters by referring to their names. Notice that the STD statement prior to SAS/STAT 9.22 has a more meaningful alias, VARIANCE, in SAS/STAT 9.22 or later.

- An error term is required in each equation in the LINEQS statement in SAS/STAT 9.22 or later.

Prior to SAS/STAT 9.22, you can set an equation in the LINEQS statement without providing an error term such as the following:

```sas
lineqs
   A = x1 * F1;
```

This means that A is perfectly predicted from F1 without an error. In SAS/STAT 9.22 or later, you need to provide an error term in each question, and the preceding specification is not syntactically valid. If a perfect relationship is indeed desirable, you can equivalently set the corresponding error variance to zero. For example, the following specification in SAS/STAT 9.22 or later achieves the purpose of specifying a perfect relationship between A and F1:
In the VARIANCE statement, the error variance of $E_1$ is fixed at zero, resulting in a perfect relationship between $A$ and $F_1$.

- Mean structure analysis is invoked by the MEANSTR option in SAS/STAT 9.22 or later.

Prior to SAS/STAT 9.22, you use the AUG and UCOV options together in the PROC CALIS statement to invoke the analysis of mean structures. These two options became obsolete in SAS/STAT 9.22. This change actually reflects more than a name change. Prior to SAS/STAT 9.22, mean structures are analyzed as augmented covariance structures in the uncorrected covariance matrix (hence the AUG and UCOV options). There are many problems with this “augmented” matrix approach. In SAS/STAT 9.22 or later, the augmented matrix was abandoned in favor of the direct parameterization approach for the mean structures. Now you must use the MEANSTR option in the PROC CALIS statement to invoke the analysis of mean structures. Alternatively, you can specify the intercepts directly in the LINEQS statement or specify the means directly in the MEAN statement. See the next two items for more information.

- Intercept parameters are set as the coefficient effects of the Intercept variable in SAS/STAT 9.22 or later.

Prior to SAS/STAT 9.22, you specify the intercept variable in the equations of the LINEQS statement by using the special “variable” named ‘INTERCEP’. For example, in the following specification, $a_1$ is the intercept parameter for the equation with dependent variable $A$:

```sas
proc calis ucov aug;
lineqs
  A = a1 * INTERCEP + b1 * B + E1;
```

In SAS/STAT 9.22 or later, although ‘INTERCEP’ is still accepted by PROC CALIS, a more meaningful alias ‘Intercept’ is also supported, as shown in the following:

```sas
proc calis;
lineqs
  A = a1 * Intercept + b1 * B + E1;
```

Intercept or INTERCEP can be typed in uppercase, lowercase, or mixed case in all versions. In addition, with the use of the Intercept variable in the LINEQS statement specification, mean structure analysis is automatically invoked for all parts of the model without the need to use the MEANSTR option in the PROC CALIS statement in SAS/STAT 9.22 or later.

- Mean parameters are specified directly in the MEAN statement in SAS/STAT 9.22 or later.

Prior to SAS/STAT 9.22, you need to specify mean parameters as covariances between the corresponding variables and the INTERCEP variable. For example, the following statements specify the mean parameters, $\text{mean}_b$ and $\text{mean}_c$, of variables B and C, respectively:
proc calis ucov aug;
  lineqs
    A = a1 * INTERCEP + b1 * B + b2 * C + E1;
  cov
    B INTERCEP = mean_b (3),
    C INTERCEP = mean_c;

In SAS/STAT 9.22 or later, you can specify these mean parameters directly in the MEAN statement, as in the following example:

proc calis;
  lineqs
    A = a1 * Intercept + b1 * B + b2 * C + E1;
  mean
    B = mean_b (3),
    C = mean_c;

This way, the types of parameters that are being specified are clearer.

- Short parameter lists do not generate constrained parameters in SAS/STAT 9.22 or later.

Prior to SAS/STAT 9.22, if you provide a shorter parameter list than expected, the last parameter specified is replicated automatically. For example, the following specification results in replicating varx as the variance parameters for variables x2–x10:

std
  x1-x10 = var0 varx;

This means that all variances for the last nine variables in the list are constrained to be the same. In SAS/STAT 9.22 or later, this is not the case. That is, var0 and varx are the variance parameters for x1 and x2, respectively, while the variances for x3–x10 are treated as unconstrained free parameters.

If you want to constrain the remaining parameters to be the same in the current version of PROC CALIS, you can use the following continuation syntax [...] at the end of the parameter list:

std
  x1-x10 = var0 varx [...];

The continuation syntax [...] repeats the specification of varx for all the remaining variance parameters for x3–x10.

- The parameter-prefix uses a new notation in SAS/STAT 9.22 or later.

Prior to SAS/STAT 9.22, you can use the parameter-prefix to generate parameter names as in the following example:
Chapter 29: The CALIS Procedure

\begin{verbatim}
lineqs
  A = x: * B + x: * C + E1;
std
  B = var:, C = var: ;
\end{verbatim}

In SAS/STAT 9.22 or later, you must replace the ‘:' notation with two trailing underscores, ‘__’, as in the following:

\begin{verbatim}
lineqs
  A = x__ * B + x__ * C + E1;
variance
  B = var__, C = var__;
\end{verbatim}

Both versions generate parameter names by appending unique integers to the prefix, as if the following has been specified:

\begin{verbatim}
lineqs
  A = x1 * B + x2 * C + E1;
variance
  B = var1, C = var2;
\end{verbatim}

In SAS/STAT 9.22 or later, you can even omit the parameter-prefix altogether. For example, you can specify the following with a null parameter-prefix:

\begin{verbatim}
lineqs
  A = __ * B + __ * C + E1;
variance
  B = __, C = __;
\end{verbatim}

In this case, PROC CALIS uses the internal prefix ‘_Parm’ to generate names, as if the following has been specified:

\begin{verbatim}
lineqs
  A = _Parm1 * B + _Parm2 * C + E1;
variance
  B = _Parm3, C = _Parm4;
\end{verbatim}
Prior to the extensive changes and feature enhancements of the CALIS procedure in SAS/STAT 9.22, an experimental version of the CALIS procedure, called PROC TCALIS, was made available in SAS/STAT 9.2. In fact, the CALIS procedure in SAS/STAT 9.22 or later builds on the foundations of the TCALIS procedure. Although the experimental TCALIS procedure and the current CALIS procedure have some similar features, they also have some major differences. This section describes these major differences so that users who have experience in using the TCALIS procedure can adapt better to the current version of the CALIS procedure. In this section, whenever the CALIS procedure is mentioned without version reference, it is assumed to be the PROC CALIS version in SAS/STAT 9.22 or later.

**Naming Parameters Is Optional in PROC CALIS**

In essence, the CALIS procedure does not require the use of parameter names in specifications, whereas the TCALIS procedure (like the PROC CALIS versions prior to SAS/STAT 9.22) does require the use of parameter names. For example, in the TCALIS procedure you might specify the following LINEQS model:

```sas
proc tcalis;
  lineqs
    X1 = 1. * F1 + E1,
    X2 = 12 * F1 + E2,
    X3 = 13 (.2) * F1 + E3;
  cov
    E1 E2 = cv12;
run;
```

Two parameters for factor loadings are used in the specification: l1 and l2. The initial value of l2 is set to 0.2. The covariance between the error terms E1 and E2 is named cv12. These parameters are not constrained, and therefore names for these parameters are not required in PROC CALIS, as shown in the following specifications:

```sas
proc calis;
  lineqs
    X1 = 1. * F1 + E1,
    X2 = * F1 + E2,
    X3 = (0.2) * F1 + E3;
  cov
    E1 E2;
run;
```

Parameter names for the two loadings in the second and the third equations are automatically generated by PROC CALIS. So is the error covariance parameter between E1 and E2. Except for the names of these parameters, the preceding PROC TCALIS and PROC CALIS specifications generate the same model.

Names for parameters are only optional in PROC CALIS, but they are not prohibited. PROC CALIS enables you to specify models efficiently without the burden of having to create parameter names unnecessarily. But you can still use parameter names and the corresponding syntax in PROC CALIS wherever you want to, much as you do in PROC TCALIS.
Another example is the following PATH model specification in PROC TCALIS:

```sas
proc tcalis;
  path
    X1 <--- F1  1. ,
    X2 <--- F1 l2 ,
    X3 <--- F1 l3 (0.2);
  pcov
    X1 X2 = cv12;
run;
```

Again, naming the unconstrained parameters l1, l2, and cv12 is not required with the CALIS procedure, as shown in the following specification:

```sas
proc calis;
  path
    X1 <--- F1 = 1. , /* path entry 1 */
    X2 <--- F1 , /* path entry 2 */
    X3 <--- F1 = (0.2); /* path entry 3 */
  pcov
    X1 X2;
run;
```

Without any parameter specification (that is, neither a name nor a value), the second path entry specifies a free parameter for the path effect (or coefficient) of F1 on X2. The corresponding parameter name for the effect is generated by PROC CALIS internally. In the third path entry, only an initial value is specified. Again, a parameter name is not necessary there. Similarly, PROC CALIS treats this as a free path effect parameter with a generated parameter name. Lastly, the error covariance between X1 and X2 is also a free parameter with a generated parameter name.

Although in PROC CALIS naming unconstrained free parameters in the PATH model is optional, PROC CALIS requires the use of equal signs before the specifications of parameters, fixed values, or initial values. The TCALIS procedure does not enforce this rule. Essentially, this stricter syntax rule in PROC CALIS makes the parameter specifications distinguishable from path specifications in path entries and therefore is necessary for the development of the multiple-path syntax, which is explained in the next section.

### Changes in the PATH Statement Syntax

The PATH statement syntax was first available in the TCALIS procedure and was also a major addition to the CALIS procedure in SAS/STAT 9.22. This statement is essential to the PATH modeling language for specifying general structural equation models. Table 29.3 summarizes the major differences between the two versions.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>PROC CALIS</th>
<th>PROC TCALIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naming unconstrained free parameters</td>
<td>Optional</td>
<td>Required</td>
</tr>
<tr>
<td>Equal signs before parameter specifications</td>
<td>Required</td>
<td>Not used</td>
</tr>
<tr>
<td>Treatment of unspecified path parameters</td>
<td>Free parameters</td>
<td>Fixed constants at 1</td>
</tr>
<tr>
<td>Multiple-path syntax</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>Extended path syntax</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
The following example shows how to specify a PATH model in PROC TCALIS:

```
proc tcalis;
  path
    F1 ---> X1 ,
    F1 ---> X2  a2 (.3),
    F1 ---> X3 a3 ,
    F2 ---> X4  1. ,
    F2 ---> X5 a5 ,
    F2 ---> X6 a6 ;
  pvar
    F1 F2 = fvar1 fvar2,
    X1-X6 = evar1-evar6;
  pcov
    F1 F2 = covF1F2;
run;
```

The following statements show to specify the preceding PATH model in PROC CALIS equivalently:

```
proc calis;
  path
    F1 ---> X1 = 1. , /* path entry 1 */
    F1 ---> X2 = (.3) , /* path entry 2 */
    F1 ---> X3 , /* path entry 3 */
    F2 ---> X4-X6 = 1. a5 a6 , /* path entry 4 */
    F1 <--> F1 , /* path entry 5 */
    F2 <--> F2 , /* path entry 6 */
    F1 <--> F2 = covF1F2 , /* path entry 7 */
    <--> X1-X6 = evar1-evar6; /* path entry 8 */
run;
```

The differences in specifications between the two versions are as follows:

- In PROC CALIS, naming unconstrained free parameters in the PATH statement is optional. For example, in the PROC TCALIS specification, the parameter names `a2` and `a3` have been used for path entries 2 and 3, respectively. They can be omitted with the PROC CALIS specification.

- In PROC CALIS, you must use equal signs before parameter specifications in all path entries. For example, equal signs are necessary in path entries 1, 2, 4, 7, and 8 to separate the parameter specifications from the path specifications. However, because equal signs are not part of the syntax in PROC TCALIS, the TCALIS procedure cannot distinguish parameter specifications from path specifications in path entries. Consequently, PROC TCALIS is incapable of handling multiple-path syntax such as path entry 4 and extended path syntax such as path entry 8.

- In PROC CALIS, unspecified parameters are treated as free parameters. For example, path entries 3, 5, and 6 are free parameters for the path effect of F1 on X3, the variance of F1, and the variance of F2, respectively. In contrast, these unspecified parameters are treated as fixed constants 1 in PROC TCALIS. That is also why path entry 1 must specify a fixed
value of 1 explicitly with the PROC CALIS specification. Otherwise, PROC CALIS treats it as a free parameter for the path effect.

- In PROC CALIS, you can use the multiple-path syntax.

For example, path entry 4 specifies three different paths from F2 in a single entry. The parameter specifications after the equal sign are distributed to the multiple paths specified in order (that is, to X4, X5, and X6, respectively). However, in PROC TCALIS you must specify these paths separately in three path entries.

- In PROC CALIS, you can use the extended path syntax to specify all kinds of parameters in the PATH statement.

For example, path entries 5 and 6 specify the variance parameters for F1 and F2, respectively. Because these variances are unconstrained free parameters, you do not need to use parameter names (but you can use them if you want). In PROC TCALIS, however, you must specify these variance parameters in the PVAR statement. Path entries 7 and 8 in the PROC CALIS specification provide other examples of the extended path syntax available only in PROC CALIS. Path entry 7 specifies the covariance parameter between F1 and F2 as CovF1F2 (although the name for this free parameter could have been omitted). You must specify this covariance parameter in the PCOV statement if you use PROC TCALIS. Path entry 8 specifies the error variances for X1–X6 as evar1–evar6, respectively. You must specify these error variance parameters in the PVAR statement if you use PROC TCALIS. Essentially, in PROC CALIS you can specify all types of parameters in the PATH model as path entries in the PATH statement. See the PATH statement for details about the extended path syntax.

### Changes in the Automatic Free Parameters in Functional Models

The CALIS and the TCALIS procedures differ in their treatment of automatic free parameters in functional models. Functional models refer to those models that can (but do not necessarily have to) analyze predictor-outcome or exogenous-endogenous relationships. In general, models that use the following statements are functional models: FACTOR, LINEQS, LISMOD, PATH, and RAM. Automatic free parameters in these functional models are those parameters that are free to estimate by default even if you do not specify them explicitly in the syntax. Table 29.4 indicates which types of parameters are automatic free parameters in PROC CALIS and PROC TCALIS.

<table>
<thead>
<tr>
<th>Automatic Free Parameters</th>
<th>PROC CALIS</th>
<th>PROC TCALIS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variances</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exogenous observed variables</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Exogenous latent factors</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Error variables</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Covariances</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between exogenous observed variables</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Between exogenous latent factors</td>
<td>Yes¹</td>
<td>No</td>
</tr>
<tr>
<td>Between exogenous observed variables and exogenous latent factors</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 29.4  continued

<table>
<thead>
<tr>
<th>Automatic Free Parameters</th>
<th>PROC CALIS</th>
<th>PROC TCALIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between error variables</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**Means**

<table>
<thead>
<tr>
<th></th>
<th>PROC CALIS</th>
<th>PROC TCALIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exogenous observed variables</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Exogenous latent factors</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**Intercepts**

<table>
<thead>
<tr>
<th></th>
<th>PROC CALIS</th>
<th>PROC TCALIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endogenous observed variables</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Endogenous latent factors</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

1. This does not apply to exploratory FACTOR models, where covariances between latent factors in the unrotated factor solution are fixed zeros.

Regarding the treatment of automatic free parameters, this table shows that unlike the CALIS procedure, PROC TCALIS does not set default free parameters in (1) the covariances between exogenous latent factors themselves and between any pairs of exogenous latent factors and observed variables, and (2) the intercepts for the endogenous observed variables.

You can compare these two schemes of setting automatic free parameters in the following three scenarios.

First, when no functional relationships are specified in the model (and hence no latent factors and no intercept parameters), the treatment of automatic free parameters by either PROC CALIS or PROC TCALIS leads to just-identified or saturated covariance and mean structures, which is certainly a reasonable “baseline” parameterization that saturates the relationships among observed variables.

Second, when there are functional relationships between observed variables in the model and no latent factors are involved, the treatment by PROC CALIS is more reasonable because it leads to a parameterization similar to that of linear regression models. That is, PROC CALIS sets the intercepts to free parameters. However, the treatment by PROC TCALIS would lead to restrictive linear regression models with zero intercepts.

Finally, when latent factors are involved, the treatment by PROC CALIS is more “natural” in the sense that the covariances among all exogenous variables are saturated in the model, rather than being restricted to zeros for the parts pertaining to latent factors, as in PROC TCALIS. Saturating the covariances between latent factors is seen to be more natural because all variables in empirical research are usually believed to be correlated, no matter how small the correlation.

Therefore, in general the PROC CALIS treatment of automatic free parameters is recommended. The treatment by PROC TCALIS might be more compatible to models that assume independent or uncorrelated latent factors such as the unrotated exploratory factor model. In this situation, to use PROC CALIS you must use the PATH, PVAR, RAM, VARIANCE, or specific MATRIX statements to set the covariances between factors to fixed zeros.
A Guide to the PROC CALIS Documentation

The CALIS procedure uses a variety of modeling languages to fit structural equation models. This chapter provides documentation for all of them. Additionally, some sections provide introductions to the model specification, the theory behind the software, and other technical details. While some introductory material and examples are provided, this chapter is not a textbook for structural equation modeling and related topics. For didactic treatment of structural equation models with latent variables, see Bollen (1989b) and Loehlin (2004).

Reading this chapter sequentially is not a good strategy for learning about PROC CALIS. This section provides a guide or “road map” to the rest of the PROC CALIS chapter, starting with the basics and continuing through more advanced topics. Many sections assume that you already have a basic understanding of structural equation modeling.

The following table shows three different skill levels of using the CALIS procedure (basic, intermediate, and advanced) and their milestones.

<table>
<thead>
<tr>
<th>Level</th>
<th>Milestone</th>
<th>Starting Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>You are able to specify simple models, but might make mistakes.</td>
<td>“Guide to the Basic Skill Level” on page 1174</td>
</tr>
<tr>
<td>Intermediate</td>
<td>You are able to specify more sophisticated models with few syntactic and semantic mistakes.</td>
<td>“Guide to the Intermediate Skill Level” on page 1180</td>
</tr>
<tr>
<td>Advanced</td>
<td>You are able to use the advanced options provided by PROC CALIS.</td>
<td>“Guide to the Advanced Skill Level” on page 1181</td>
</tr>
</tbody>
</table>

In the next three sections, each skill level is discussed, followed by an introductory section of the reference topics that are not covered in any of the skill levels.

Guide to the Basic Skill Level

- Overview of PROC CALIS
- Basic Model Specification
- Syntax Overview
- Details about Various Types of Models
Overview of PROC CALIS
The section “Overview: CALIS Procedure” on page 1158 gives you an overall picture of the CALIS procedure but without the details.

Basic Model Specification
The structural equation example in the section “Getting Started: CALIS Procedure” on page 1184 provides the starting point to learn the basic model specification. You learn how to represent your theory by using a path diagram and then translate the diagram into the PATH model for PROC CALIS to analyze. Because the PATH modeling language is new, this example is useful whether or not you have previous experience with PROC CALIS. The PATH model is specified in the section “PATH Model” on page 1186. The corresponding results are shown and discussed in Example 29.17.

After you learn about the PATH modeling language and an example of its application, you can do either of the following:

- You can continue to learn more modeling languages in the section “Getting Started: CALIS Procedure” on page 1184.
- You can skip to the section “Syntax Overview” on page 1179 for an overview of the PROC CALIS syntax and learn other modeling languages at a later time.

You do not need to learn all of the modeling languages in PROC CALIS. Any one of the modeling languages (LINEQS, LISMOD, PATH, or RAM) is sufficient for specifying a very wide class of structural equation models. PROC CALIS provides different kinds of modeling languages because different researchers might have previously learned different modeling languages or approaches. To get a general idea about different kinds of modeling languages, the following subsections in the “Getting Started: CALIS Procedure” section are useful:

- LINEQS: Section “LINEQS Model” on page 1188
- RAM: Section “RAM Model” on page 1187
- LISMOD: Section “LISMOD Model” on page 1189
- FACTOR: Section “A Factor Model Example” on page 1190
- MSTRUCT: Section “Direct Covariance Structures Analysis” on page 1192

After studying the examples in the “Getting Started: CALIS Procedure” section, you can strengthen your understanding of the various modeling languages by studying more examples such as those in section “Examples: CALIS Procedure” on page 1539. Unlike the examples in the “Getting Started: CALIS Procedure” section, the examples in the “Examples: CALIS Procedure” section include the analysis results in addition to the explanations of the model specifications.

You can start with the following two sets of basic examples:

- MSTRUCT model examples
  The basic MSTRUCT model examples demonstrate the testing of covariance structures directly on the covariance matrices. Although the MSTRUCT model is not the most common structural equation models in applications, these MSTRUCT examples can help you understand the basic form of covariance structures and the corresponding specifications in PROC CALIS.
• PATH model examples
The basic PATH model examples demonstrate how you can represent your model by path diagrams and by the PATH modeling language. These examples show the most common applications of structural equation modeling.

The following is a summary of the basic MSTRUCT model examples:

• “Example 29.1: Estimating Covariances and Correlations” on page 1539 shows how you can estimate the covariances and correlations with standard error estimates for the variables in your model. The model you fit is a saturated covariance structure model.

• “Example 29.2: Estimating Covariances and Means Simultaneously” on page 1544 extends Example 29.1 to include the mean structures in the model. The model you fit is a saturated mean and covariance structure model.

• “Example 29.3: Testing Uncorrelatedness of Variables” on page 1545 shows a very basic covariance structure model, in which the covariance structures can be specified directly. The variables in this model are uncorrelated. You learn how to specify the covariance pattern directly.

• “Example 29.4: Testing Covariance Patterns” on page 1547 extends Example 29.3 to include other covariance structures that you can specify directly.

• “Example 29.5: Testing Some Standard Covariance Pattern Hypotheses” on page 1550 illustrates the use of built-in covariance patterns supported by PROC CALIS.

The following is a summary of the basic PATH model examples:

• “Example 29.6: Linear Regression Model” on page 1553 shows how you can fit a linear regression model with the PATH modeling language of PROC CALIS. This example also introduces the path diagram representation of “causal” models. You compare results obtained from PROC CALIS and from the REG procedure, which is designed specifically for regression analysis.

• “Example 29.7: Multivariate Regression Models” on page 1557 extends Example 29.6 in several different ways. You fit covariance structure models with more than one predictor, with direct and indirect effects. This example also discusses how you can choose the “best” model for your data.

• “Example 29.8: Measurement Error Models” on page 1574 explores the case where the predictor in simple linear regression is measured with error. The concept of latent true score variable is introduced. You use PROC CALIS to fit a simple measurement error model.

• “Example 29.9: Testing Specific Measurement Error Models” on page 1580 extends Example 29.8 to test special measurement error models with constraints. By using PROC CALIS, you can constrain your measurement error models in many different ways. For example, you can constrain the error variances or the intercepts to test specific hypotheses.

• “Example 29.10: Measurement Error Models with Multiple Predictors” on page 1586 extends Example 29.8 to include more predictors in the measurement error models. The measurement errors in the predictors can be correlated in the model.

More elaborate examples about the MSTRUCT and PATH models are listed as follows:
“Example 29.17: Path Analysis: Stability of Alienation” on page 1644 shows you how to specify a simple PATH model and interpret the basic estimation results. The results are shown in considerable detail. The output and analyses include: a model summary, an initial model specification, an initial estimation method, an optimization history and results, residual analyses, residual graphics, estimation results, squared multiple correlations, and standardized results.

“Example 29.19: Fitting Direct Covariance Structures” on page 1665 shows you how to fit your covariance structures directly on the covariance matrix by using the MSTRUCT modeling language. You also learn how to use the FITINDEX statement to create a customized model fit summary and how to save the fit summary statistics into an external file.

“Example 29.21: Testing Equality of Two Covariance Matrices Using a Multiple-Group Analysis” on page 1680 uses the MSTRUCT modeling language to illustrate a simple multiple-group analysis. You also learn how to use the ODS SELECT statement to customize your printed output.

“Example 29.22: Testing Equality of Covariance and Mean Matrices between Independent Groups” on page 1685 uses the COVPATTERN= and MEANPATTERN= options to show some tests of equality of covariance and mean matrices between independent groups. It also illustrates how you can improve your model fit by the exploratory use of the Lagrange multiplier statistics for releasing equality constraints.

“Example 29.24: Testing Competing Path Models for the Career Aspiration Data” on page 1718 illustrates how you can fit competing models by using the OUTMODEL= and INMODEL= data sets for transferring and modifying model information from one analysis to another. This example also demonstrates how you can choose the best model among several competing models for the same data.

After studying the PATH and MSTRUCT modeling languages, you are able to specify most commonly used structural equation models by using PROC CALIS. To broaden your scope of structural equation modeling, you can study some basic examples that use the FACTOR and LINEQS modeling languages. These basic examples are listed as follows:

“Example 29.11: Measurement Error Models Specified As Linear Equations” on page 1591 explores another way to specify measurement error models in PROC CALIS. The LINEQS modeling language is introduced. You learn how to specify linear equations of the measurement error model by using the LINEQS statement. Unlike the PATH modeling language, in the LINEQS modeling language, you need to specify the error terms explicitly in the model specification.

“Example 29.12: Confirmatory Factor Models” on page 1597 introduces a basic confirmatory factor model for test items. You use the FACTOR modeling language to specify the factor-variable relationships.

“Example 29.13: Confirmatory Factor Models: Some Variations” on page 1608 extends Example 29.12 to include some variants of the confirmatory factor model. With the flexibility of the FACTOR modeling language, this example shows how you fit models with parallel items, tau-equivalent items, or partially parallel items.

More advanced examples that use the PATH, LINEQS, and FACTOR modeling languages are listed as follows:
• “Example 29.14: Residual Diagnostics and Robust Estimation” on page 1616 illustrates the use of several graphical residual plots to detect model outliers and leverage observations, to study the departures from the theoretical case-level residual distribution, and to examine the linearity and homoscedasticity of variance. In addition, this example illustrates the use of robust estimation technique to downweight the outliers and to estimate the model parameters.

• “Example 29.15: The Full Information Maximum Likelihood Method” on page 1629 shows how you can use the full information maximum likelihood (FIML) method to estimate your model when your data contain missing values. It illustrates the analysis of the data coverage of the sample variances, covariances, and means and the analysis of missing patterns and the mean profile. It also shows that the full information maximum likelihood method makes the maximum use of the available information from the data, as compared with the default ML (maximum likelihood) methods.

• “Example 29.16: Comparing the ML and FIML Estimation” on page 1639 discusses the similarities and differences between the ML and FIML estimation methods as implemented in PROC CALIS. It uses an empirical example to show how ML and FIML obtain the same estimation results when the data do not contain missing values.

• “Example 29.18: Simultaneous Equations with Mean Structures and Reciprocal Paths” on page 1658 is an econometric example that shows you how to specify models using the LINEQS modeling language. This example also illustrates the specification of reciprocal effects, the simultaneous analysis of the mean and covariance structures, the setting of bounds for parameters, and the definitions of metaparameters by using the PARAMETERS statement and SAS programming statements. You also learn how to shorten your output results by using some global display options such as the PSHORT and NOSTAND options in the PROC CALIS statement.

• “Example 29.20: Confirmatory Factor Analysis: Cognitive Abilities” on page 1668 uses the FACTOR modeling language to illustrate confirmatory factor analysis. In addition, you use the MODIFICATION option in the PROC CALIS statement to compute LM test indices for model modifications.

• “Example 29.25: Fitting a Latent Growth Curve Model” on page 1730 is an advanced example that illustrates the use of structural equation modeling techniques for fitting latent growth curve models. You learn how to specify random intercepts and random slopes by using the LINEQS modeling language. In addition to the modeling of the covariance structures, you also learn how to specify the mean structure parameters.

If you are familiar with the traditional Keesling-Wiley-Jöreskog measurement and structural models (Keesling 1972; Wiley 1973; Jöreskog 1973) or the RAM model (McArdle 1980), you can use the LISMOD or RAM modeling languages to specify structural equation models. The following example shows how to specify these types of models:

• “Example 29.23: Illustrating Various General Modeling Languages” on page 1708 extends Example 29.17, which uses the PATH modeling language, and shows how to use the other general modeling languages: RAM, LINEQS, and LISMOD. These modeling languages enable you to specify the same path model as in Example 29.17 and get equivalent results. This example shows the connections between the general modeling languages supported in PROC CALIS. A good understanding of Example 29.17 is a prerequisite for this example.
Once you are familiar with various modeling languages, you might wonder which modeling language should be used in a given situation. The section “Which Modeling Language?” on page 1194 provides some guidelines and suggestions.

**Syntax Overview**

The section “Syntax: CALIS Procedure” on page 1195 shows the syntactic structure of PROC CALIS. However, reading the “Syntax: CALIS Procedure” section sequentially might not be a good strategy. The statements used in PROC CALIS are classified in the section “Classes of Statements in PROC CALIS” on page 1196. Understanding this section is a prerequisite for understanding single-group and multiple-group analyses in PROC CALIS. Syntax for single-group analyses is described in the section “Single-Group Analysis Syntax” on page 1199, and syntax for multiple-group analyses is described in the section “Multiple-Group Multiple-Model Analysis Syntax” on page 1199.

You might also want to get an overview of the options in the PROC CALIS statement. However, you can skip the detailed listing of the available options in the PROC CALIS statement. Most of these details serve as references, so you can consult them only when you need to. You can just read the summary tables for the available options in the PROC CALIS statement in the following subsections:

- “Data Set Options” on page 1200
- “Model and Estimation Options” on page 1201
- “Options for Fit Statistics” on page 1201
- “Options for Statistical Analysis” on page 1202
- “Global Display Options” on page 1203
- “Optimization Options” on page 1204

**Details about Various Types of Models**

Several subsections in the section “Details: CALIS Procedure” on page 1367 can help you gain a deeper understanding of the various types of modeling languages, as shown in the following table:

<table>
<thead>
<tr>
<th>Language</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>COSAN</td>
<td>“The COSAN Model” on page 1390</td>
</tr>
<tr>
<td>FACTOR</td>
<td>“The FACTOR Model” on page 1394</td>
</tr>
<tr>
<td>LINEQS</td>
<td>“The LINEQS Model” on page 1401</td>
</tr>
<tr>
<td>LISMOD</td>
<td>“The LISMOD Model and Submodels” on page 1408</td>
</tr>
<tr>
<td>MSTRUCT</td>
<td>“The MSTRUCT Model” on page 1416</td>
</tr>
<tr>
<td>PATH</td>
<td>“The PATH Model” on page 1419</td>
</tr>
<tr>
<td>RAM</td>
<td>“The RAM Model” on page 1425</td>
</tr>
</tbody>
</table>

The specification techniques you learn from the examples cover only parts of the modeling language. A more complete treatment of the modeling languages is covered in these subsections. In addition, you can also learn the mathematical models, model restrictions, and default parameterization of all supported modeling languages in these subsections. To get an overall idea about the default parameterization rules used in PROC CALIS, the section “Default Analysis Type and Default Parameterization” on page 1388 would be very useful. Understanding how PROC CALIS set default parameters would help you specify your models more efficiently and accurately.
Guide to the Intermediate Skill Level

At the intermediate level, you learn to minimize your mistakes in model specification and to establish more sophisticated modeling techniques. The following topics in the “Details: CALIS Procedure” section or elsewhere can help:

- The section “Naming Variables and Parameters” on page 1433 summarizes the naming rules and conventions for variable and parameter names in specifying models.

- The section “Setting Constraints on Parameters” on page 1434 covers various techniques of constraining parameters in model specifications.

- The section “Automatic Variable Selection” on page 1439 discusses how PROC CALIS treats variables in the models and variables in the data sets. It also discusses situations where the VAR statement specification is deemed necessary.

- The section “Computational Problems” on page 1515 discusses computational problems that occur quite commonly in structural equation modeling. It also discusses some possible remedies of the computational problem.

- The section “Missing Values and the Analysis of Missing Patterns” on page 1503 describes the default treatment of missing values.

- The statements REFMODEL on page 1354 and RENAMEPARM on page 1356 are useful when you need to make references to well-defined models when specifying a “new” model. See Example 29.28 for an application.

Revisit topics and examples covered at the basic level, as needed, to help you better understand the topics at the intermediate level.

You can also study the following more advanced examples:

- “Example 29.26: Higher-Order and Hierarchical Factor Models” on page 1736 is an advanced example for confirmatory factor analysis. It involves the specifications of higher-order and hierarchical factor models. Because higher-order factor models cannot be specified by the FACTOR modeling language, you need to use the LINEQS model specification instead. A second-order factor model and a bifactor model are fit. Linear constraints on parameters are illustrated by using the PARAMETERS statement and SAS programming statements. Relationships between the second-order factor model and the bifactor model are numerically illustrated.

- “Example 29.27: Linear Relations among Factor Loadings” on page 1750 is an advanced example of a first-order confirmatory factor analysis that uses the FACTOR modeling language. In this example, you learn how to use the PARAMETERS statement and SAS programming statements to set up dependent parameters in your model. You also learn how to specify the correlation structures for a specific confirmatory factor model.

- “Example 29.28: Multiple-Group Model for Purchasing Behavior” on page 1759 is a sophisticated example of analyzing a path model. The PATH modeling language is used. In this example, a two-group analysis of mean and covariance structures is conducted. You learn how to use the REFMODEL statement to reference properly defined models and the SIMTESTS statement to test a priori simultaneous hypotheses.
• “Example 29.29: Fitting the RAM and EQS Models by the COSAN Modeling Language” on page 1784 introduces the COSAN modeling language by connecting it with general RAM and EQS models. The model matrices of the RAM or EQS model are described. You specify these model matrices and the associated parameters in the COSAN modeling language.

• “Example 29.30: Second-Order Confirmatory Factor Analysis” on page 1799 constructs the covariance structure model of the second-order confirmatory factor model. You define the model matrices by using the COSAN modeling language.

• “Example 29.31: Linear Relations among Factor Loadings: COSAN Model Specification” on page 1804 shows how you can set linear constraints among model parameters under the COSAN model.

• “Example 29.32: Ordinal Relations among Factor Loadings” on page 1809 shows how you can set ordinal constraints among model parameters under the COSAN model.

• “Example 29.33: Longitudinal Factor Analysis” on page 1812 defines the covariance structures of a longitudinal factor model and shows how you can specify the covariance structure model with the COSAN modeling language.

Guide to the Advanced Skill Level

At the advanced level, you learn to use the advanced data analysis and output control tools supported by PROC CALIS.

Advanced Data Analysis Tools

The following advanced data analysis topics are discussed:

• Assessment of fit

  The section “Assessment of Fit” on page 1481 presents the fit indices used in PROC CALIS. However, the more important topics covered in this section are about how model fit indices are organized and used, how residuals can be used to gauge the fitting of individual parts of the model, and how the coefficients of determination are defined for equations.

  To customize your fit summary table, you can use the options on the FITINDEX statement.

• Case-level residual diagnostics

  The section “Case-Level Residuals, Outliers, Leverage Observations, and Residual Diagnostics” on page 1494 describes details about how the residual diagnostics at the individual data level are accomplished in general structural equation modeling, and how they lead to the graphical techniques for detecting outliers and leverage observations, studying residual distributions, and examining linear relationships and heteroscedasticity of error variances.

• Control and customization of path diagrams

  The section “Path Diagrams: Layout Algorithms, Default Settings, and Customization” on page 1440 discusses the path diagram layout algorithms that the CALIS procedure uses. It also illustrates useful options that control and customize path diagrams.
• Effect partitioning

The section “Total, Direct, and Indirect Effects” on page 1498 discusses the total, direct, and indirect effects and their computations. The stability coefficient of reciprocal causation is also defined.

To customize the effect analysis, you can use the `EFFPART` statement.

• Counting and adjusting degrees of freedom

The section “Counting the Degrees of Freedom” on page 1478 describes how PROC CALIS computes model fit degrees of freedom and how you can use some options on the PROC CALIS statement to make degrees-of-freedom adjustments.

To adjust the model fit degrees of freedom, you can use the `DFREDUCE=` and `NOADJDF` options in the PROC CALIS statement.

• Standardized solutions

Standardization schemes used in PROC CALIS are described and discussed in the section “Standardized Solutions” on page 1500.

Standardized solutions are displayed by default. You can turn them off by using the `NOSTAND` option of the PROC CALIS statement.

• Model modifications

In the section “Modification Indices” on page 1501, modification indices such as Lagrange multiplier test indices and Wald statistics are defined and discussed. These indices can be used either to enhance your model fit or to make your model more precise.

To limit the modification process only to those parameters of interest, you can use the `LMTESTS` statement to customize the sets of LM tests conducted on potential parameters.

• A Priori Parametric Function Testing

You can use the `TESTFUNC` statement to test a priori hypotheses individually. You can use the `SIMTESTS` statement to test a priori hypotheses simultaneously.

**Advanced Output Control Tools**

To be more effective in presenting your analysis results, you need to be more sophisticated in controlling your output. Some customization tools have been discussed in the previous section “Advanced Data Analysis Tools” on page 1181 and might have been mentioned in the examples included in the basic and the intermediate levels. In the following topics, these output control tools are presented in a more organized way so that you can have a systematic study scheme of these tools.

• Global output control tools in PROC CALIS

You can control output displays in PROC CALIS either by the `global display options` or by the individual output printing options. Each global display option typically controls more than one output display, while each individual output display controls only one output display. The global display options can both enable and suppress output displays, and they can also alter the format of the output.

See the `ALL`, `PRINT`, `PSHORT`, `PSUMMARY`, and `NOPRINT` options for ways to control the appearances of the output. See the section “Global Display Options” on page 1203 for details about the global display options and their relationships with the individual output display options. Also see
the ORDERALL, ORDERGROUPS, ORDERMODELS, ORDERSPEC, PARMNAME, PRIMAT, NOORDERSPEC, NOPARMNAME, NOSTAND, and NOSE options which control the output formats.

• Customized analysis tools in PROC CALIS

Many individual output displays in PROC CALIS can be customized via specific options or statements. If you do not use these customization tools, the default output will usually contain a large number of displays or displays with very large dimensions. These customized analysis tools are as follows:

– The ON=, OFF=, ON(ONLY)= options in the FITINDEX statement enable you to select individual or groups of model fit indices or modeling information to display. You can still save the information of all fit indices in an external file by using the OUTFIT= option.

– The EFFPART statement enables you to customize the effect analysis. You display only those effects of substantive interest.

– The LMTESTS statement enables you to customize the sets of LM tests of interest. You test only those potential parameters that are theoretically and substantively possible.

• Output selection and destinations by the ODS system

This kind of output control is used not only for PROC CALIS, but is used for all procedures that support the ODS system. The most common uses include output selection and output destinations assignment. You use the ODS SELECT statement together with the ODS table names or graph names to select particular output displays. See the section “ODS Table Names” on page 1523 for these names in PROC CALIS.

The default output destination of PROC CALIS is the listing destination. You can add or change the destinations by using statements such as ods html (for html output), ods rtf (for rich text output), and so on. For details, see Chapter 20, “Using the Output Delivery System.”

Reference Topics

Some topics in the “Details: CALIS Procedure” section are intended primarily for references—you consult them only when you encounter specific problems in the PROC CALIS modeling or when you need to know the very fine technical details in certain special situations. Many of these reference topics in the “Details: CALIS Procedure” section are not required for practical applications of structural equation modeling. The following technical topics are discussed:

• Measures of multivariate kurtosis and skewness

This is covered in the section “Measures of Multivariate Kurtosis” on page 1504.

• Estimation criteria and the mathematical functions for estimation

The section “Estimation Criteria” on page 1463 presents formulas for various estimation criteria. The relationships among these criteria are shown in the section “Relationships among Estimation Criteria” on page 1473. To optimize an estimation criterion, you usually need its gradient and Hessian functions. These functions are detailed in the section “Gradient, Hessian, Information Matrix, and Approximate Standard Errors” on page 1475, where you can also find information about the computation of the standard error estimates in PROC CALIS. Unlike other estimation methods, the robust estimation methods do not optimize a discrepancy function themselves. The robust estimation methods that are implemented in PROC CALIS use the iteratively reweighted least squares (IRLS) method to obtain
Chapter 29: The CALIS Procedure

parameter convergence. The robust estimation technique is detailed in the section “Robust Estimation” on page 1470.

• Initial estimation

Initial estimates are necessary for all kinds of iterative optimization techniques. They are described in section “Initial Estimates” on page 1506.

• Use of optimization techniques

Optimization techniques are covered in section “Use of Optimization Techniques” on page 1507. See this section if you need to fine-tune the optimization.

• Output displays and control

The output displays in PROC CALIS are listed in the section “Displayed Output” on page 1518. General requirements for the displays are also shown.

With the ODS system, each table and graph has a name, which can be used on the ODS OUTPUT or ODS SELECT statement. See the section “ODS Table Names” on page 1523 for the ODS table and graph names.

• Input and output files

PROC CALIS supports several input and output data files for data, model information, weight matrices, estimates, fit indices, and estimation and descriptive statistics. The uses and the structures of these input and output data files are described in the sections “Input Data Sets” on page 1367 and “Output Data Sets” on page 1371.

---

Getting Started: CALIS Procedure

A Structural Equation Example

This example from Wheaton et al. (1977) illustrates the basic uses of the CALIS procedure and the relationships among the LINEQS, LISMOD, PATH, and RAM modeling languages. Different structural models for these data are analyzed in Jöreskog and Sörbom (1985) and in (Bentler 1995, p. 28). The data contain the following six (manifest) variables collected from 932 people in rural regions of Illinois:

Anomie67: Anomie 1967
Powerless67: Powerlessness 1967
Anomie71: Anomie 1971
Powerless71: Powerlessness 1971
Education: Education level (years of schooling)
SEI: Duncan’s socioeconomic index (SEI)
The covariance matrix of these six variables is stored in the data set named Wheaton.

It is assumed that anomie and powerlessness are indicators of an alienation factor and that education and SEI are indicators for a socioeconomic status (SES) factor. Hence, the analysis contains three latent variables (factors):

- Alien67: Alienation 1967
- Alien71: Alienation 1971
- SES: Socioeconomic status (SES)

The following path diagram shows the structural model used in Bentler (1985, p. 29) and slightly modified in Jöreskog and Sörbom (1985, p. 56):

**Figure 29.1** Path Diagram of Stability and Alienation Example

In the path diagram shown in **Figure 29.1**, regressions of variables are represented by one-headed arrows. Regression coefficients are indicated along these one-headed arrows. Variances and covariances among the variables are represented by two-headed arrows. Error variances and covariances are also represented by two-headed arrows. This scheme of representing paths, variances and covariances, and error variances and covariances (McArdle 1988; McDonald 1985) is helpful in translating the path diagram to the PATH or RAM model input in the CALIS procedure.
Chapter 29: The CALIS Procedure

PATH Model

Specification by using the PATH modeling language is direct and intuitive in PROC CALIS once a path diagram is drawn. The following statements specify the path diagram almost intuitively:

```plaintext
proc calis nobs=932 data=Wheaton;
path
  Anomie67  <=> Alien67  = 1.0,
  Powerless67  <=> Alien67  = 0.833,
  Anomie71  <=> Alien71  = 1.0,
  Powerless71  <=> Alien71  = 0.833,
  Education  <=> SES       = 1.0,
  SEI       <=> SES       = lambda,
  Alien67  <=> SES       = gamma1,
  Alien71  <=> SES       = gamma2,
  Alien71  <=> Alien67  = beta;
pvar
  Anomie67  = theta1,
  Powerless67  = theta2,
  Anomie71  = theta1,
  Powerless71  = theta2,
  Education  = theta3,
  SEI       = theta4,
  Alien67  = psi1,
  Alien71  = psi2,
  SES       = phi;
pcov
  Anomie67  Anomie71  = theta5,
  Powerless67  Powerless71  = theta5;
run;
```

In the PROC CALIS statement, you specify Wheaton as the input data set, which contains the covariance matrix of the variables.

In the PATH model specification, all the one-headed arrows in the path diagram are represented as path entries in the PATH statement, with entries separated by commas. In each path entry, you specify a pair of variables and the direction of the path (either `<==` or `===>`), followed by a path coefficient, which is either a fixed constant or a parameter with a name in the specification.

All the two-headed arrows each with the same source and destination are represented as entries in the PVAR statement, with entries separated by commas. In the PVAR statement, you specify the variance or error (or partial) variance parameters. In each entry, you specify a variable and then a parameter name or a fixed parameter value. If the variable involved is exogenous in the model (serves only as a predictor; never being pointed at by one-headed arrows), you are specifying a variance parameter for an exogenous variable in the PVAR statement. Otherwise, you are specifying an error variance (or a partial variance) parameter for an endogenous variable.

All other two-headed arrows are represented as entries in the PCOV statement, with entries separated by commas. In the PCOV statement, you specify the covariance or error (or partial) covariance parameters. In each entry, you specify a pair of variables and then a parameter name or a fixed parameter value. If both variables involved in an entry are exogenous, you are specifying a covariance parameter. If both variables involved in an entry are endogenous, you are specifying an error (or partial) covariance parameter. When one variable is exogenous and the other is endogenous in an entry, you are specifying a partial covariance.
parameter that can be interpreted as the covariance between the exogenous variable and the error of the endogenous variable.

See Example 29.17 for the results of the current PATH model analysis. For more information about the PATH modeling language, see the section “The PATH Model” on page 1419 and the PATH statement on page 1322.

**RAM Model**

The PATH modeling language is not the only specification method that you can use to represent the path diagram. You can also use the RAM, LINEQS or LISMOD modeling language to represent the diagram equivalently.

The RAM model specification in PROC CALIS resembles that of the PATH model, as shown in the following statements:

```plaintext
proc calis nobs=932 data=Wheaton;
  ram
    var = Anomie67 /* 1 */
         Powerless67 /* 2 */
         Anomie71 /* 3 */
         Powerless71 /* 4 */
         Education /* 5 */
         SEI /* 6 */
         Alien67 /* 7 */
         Alien71 /* 8 */
         SES, /* 9 */
    _A_ 1 7 1.0,
    _A_ 2 7 0.833,
    _A_ 3 8 1.0,
    _A_ 4 8 0.833,
    _A_ 5 9 1.0,
    _A_ 6 9 lambda,
    _A_ 7 9 gamma1,
    _A_ 8 9 gamma2,
    _A_ 8 7 beta,
    _P_ 1 1 theta1,
    _P_ 2 2 theta2,
    _P_ 3 3 theta1,
    _P_ 4 4 theta2,
    _P_ 5 5 theta3,
    _P_ 6 6 theta4,
    _P_ 7 7 psi1,
    _P_ 8 8 psi2,
    _P_ 9 9 phi,
    _P_ 1 3 theta5,
    _P_ 2 4 theta5;
run;
```

In the RAM statement, you specify a list of entries for parameters, with entries separated by commas. In each entry, you specify the type of parameter (PATH, PVAR, or PCOV in the code), the associated variable or pair of variables and the path direction if applicable, and then a parameter name or a fixed parameter value. The types of parameters you specify in this RAM model are for path coefficients, variances or partial variances, and covariances or partial covariances. They bear the same meanings as those in the PATH model specified previously. The RAM model specification is therefore quite similar to the PATH model specification—except that in the RAM model you put all parameter specification in the same list under the RAM statement, whereas you specify different types of parameters separately under different statements in the PATH model.
Chapter 29: The CALIS Procedure

See Example 29.23 for partial results of the current RAM model analysis. For more information about the RAM modeling language, see the section “The RAM Model” on page 1425 and the RAM statement on page 1348.

LINEQS Model

The LINEQS modeling language uses equations to specify functional relationships among variables, as shown in the following statements:

```sas
proc calis nobs=932 data=Wheaton;
lineqs
   Anomie67   = 1.0   * f_Alien67 + E1,
   Powerless67 = 0.833 * f_Alien67 + E2,
   Anomie71   = 1.0   * f_Alien71 + E3,
   Powerless71 = 0.833 * f_Alien71 + E4,
   Education = 1.0   * f_SES + E5,
   SEI       = lambda * f_SES + E6,
   f_Alien67  = gamma1 * f_SES + psi1,
   f_Alien71  = gamma2 * f_SES + beta * Alien67 + psi2;
std
   E1 = theta1,
   E2 = theta2,
   E3 = theta1,
   E4 = theta2,
   E5 = theta3,
   E6 = theta4,
   D1 = psi1,
   D2 = psi2,
   f_SES = phi;
cov
   E1 E3 = theta5,
   E2 E4 = theta5;
run;
```

In the LINEQS statement, equations are separated by commas. In each equation, you specify an endogenous variable on the left-hand side, and then predictors and path coefficients on the right-hand side of the equal side. The set of equations specified in this LINEQS model is equivalent to the system of paths specified in the preceding PATH (or RAM) model. However, there are some notable differences between the LINEQS and the PATH specifications.

First, in the LINEQS modeling language you must specify the error terms explicitly as exogenous variables. For example, E1, E2, and D1 are error terms in the specification. In the PATH (or RAM) modeling language, you do not need to specify error terms explicitly.

Second, equations specified in the LINEQS modeling language are oriented by the endogenous variables. Each endogenous variable can appear on the left-hand side of an equation only once in the LINEQS statement. All the corresponding predictor variables must then be specified on the right-hand side of the equation. For example, f_Alien71 is predicted from f_Alien67 and f_SES in the last equation of the LINEQS statement. In the PATH or RAM modeling language, however, you would specify the same functional relationships in two separate paths.

Third, you must follow some naming conventions for latent variables when using the LINEQS modeling language. The names of latent variables that are not errors or disturbances must start with an ‘f’ or ‘F’.
Also, the names of the error variables must start with ‘e’ or ‘E’ and the names of the disturbance variables must start with ‘d’ or ‘D’. For example, variables Alien67, Alien71, and SES serve as latent factors in the previous PATH or RAM model specification. To comply with the naming conventions, these variables are named with an extra prefix ‘f_’ in the LINEQS model specification—that is, f_Alien67, f_Alien71, and f_SES, respectively. In addition, because of the naming conventions of the LINEQS modeling language, E1–E6 serve as error terms and D1–D1 serve as disturbances in the specification.

A consequence of explicit specification of error terms in the LINEQS statement is that the partial variance and partial covariance concepts used in the PATH and RAM modeling languages are no longer needed. They are replaced by the variances or covariances of the error terms or disturbances. Errors and disturbances are exogenous variables by nature. Hence, in terms of variance and covariance specification, they are treated exactly the same way as other non-error exogenous variables in the LINEQS modeling language. That is, variance parameters for all exogenous variables, including errors and disturbances, are specified in the VARIANCE statement, and covariance parameters among exogenous variables, including errors and disturbances, are specified in COV statement.

See Example 29.23 for partial results of the current LINEQS model analysis. For more information about the LINEQS modeling language, see the section “The LINEQS Model” on page 1401 and the LINEQS statement on page 1275.

**LISMOD Model**

The LISMOD language is quite different from the LINEQS, PATH, and RAM modeling languages. In the LISMOD specification, you define parameters as entries in model matrices, as shown in the following statements:

```plaintext
proc calis nobs=932 data=Wheaton;
   lismod
       yvar = Anomie67 Powerless67 Anomie71 Powerless71,
       xvar = Education SEI,
       etav = Alien67 Alien71,
       xiv = SES;
   matrix _LAMBDAY_ [1,1] = 1.0,
                    [2,1] = 0.833,
                    [3,2] = 1.0,
                    [4,2] = 0.833;
   matrix _LAMBDAX_ [1,1] = 1.0,
                    [2,1] = lambda;
   matrix _GAMMA_  [1,1] = gamma1,
                    [2,1] = gamma2;
   matrix _BETA_   [2,1] = beta;
   matrix _THETAY_ [1,1] = theta1,
                    [2,2] = theta2,
                    [3,3] = theta1,
                    [4,4] = theta2,
                    [3,1] = theta5,
                    [4,2] = theta5;
   matrix _THETAX_ [1,1] = theta3,
                    [2,2] = theta4;
   matrix _PSI_    [1,1] = psi1,
                    [2,2] = psi2;
   matrix _PHI_    [1,1] = phi;
run;
```
In the **LISMOD** statement, you specify the lists of variables in the model. In the **MATRIX** statements, you specify the parameters in the LISMOD model matrices. Each **MATRIX** statement contains the matrix name of interest and then locations of the parameters, followed by the parameter names or fixed parameter values. It would be difficult to explain the LISMOD specification here without better knowledge about the formulation of the mathematical model. For this purpose, see the section “The LISMOD Model and Submodels” on page 1408 and the **LISMOD statement** on page 1282. See also Example 29.23 for partial results of the current LISMOD model analysis.

**COSAN Model**

The COSAN model specification is even more abstract than all of the modeling languages considered. Like the LISMOD model specification, to specify a COSAN model you need to define parameters as entries in model matrices. In addition, you must also provide the definitions of the model matrices and the matrix formula for the covariance structures in the COSAN model specification. Therefore, the COSAN model specification requires sophisticated knowledge about the formulation of the mathematical model. For this reason, the COSAN model specification of the preceding path model is not discussed here (but see Example 29.29). For more details about the COSAN model specification, see the section “The COSAN Model” on page 1390 and the **COSAN statement** on page 1242.

**A Factor Model Example**

In addition to the general modeling languages such as PATH, RAM, LINEQS, and LISMOD, the CALIS procedure provides a specialized language for factor analysis. In the **FACTOR** modeling language, you can specify either exploratory or confirmatory factor models. For exploratory factor models, you can specify the number of factors, factor extraction method, and rotation algorithm, among many other options. For confirmatory factor models, you can specify the variable-factor relationships, factor variances and covariances, and the error variances.

For example, the following is an exploratory factor model fitted to the Wheaton et al. (1977) data by using **PROC CALIS**:

```
proc calis nobs=932 data=Wheaton corr;
   factor n=2 rotate=varimax;
run;
```

In this model, you want to get the varimax-rotated solution with two factors by analyzing the correlation matrix (with the **CORR** option). By default, the factor extraction method is maximum likelihood (**METHOD=ML**). Maximum likelihood exploratory factor analysis by **PROC CALIS** can also be done equivalently by the **FACTOR** procedure, as shown in the following statements for the Wheaton et al. (1977) data:

```
proc factor nobs=932 data=Wheaton n=2 rotate=varimax method=ml;
run;
```

Note that **METHOD=ML** is necessary because maximum likelihood is not the default method in **PROC FACTOR**.
Whereas you can use either the CALIS or FACTOR procedure to fit certain exploratory factor models for correlations, you can only use the CALIS procedure to fit confirmatory factor models. In a confirmatory factor model, you are assumed to have some prior knowledge about the variable-factor relations. For example, in your substantive theory, some observed variables are not related to certain factors in the model. The following statements illustrate the specification of a confirmatory factor model for Wheaton et al. (1977) data:

```sas
proc calis nobs=932 data=Wheaton;
factor
   Alien67 ===> Anomie67 Powerless67 = 1.0 load1,
   Alien71 ===> Anomie71 Powerless71 = 1.0 load2,
   SES    ===> Education SEI = 1.0 load3;
pvar
   Alien67 = phi11,
   Alien71 = phi22,
   SES    = phi33,
   Anomie67 = theta1,
   Powerless67 = theta2,
   Anomie71 = theta3,
   Powerless71 = theta4,
   Education = theta5,
   SEI    = theta6;
cov
   Alien71 Alien67 = phi21,
   SES    Alien67 = phi31,
   SES    Alien71 = phi32;
run;
```

Unlike the model fitted by the PATH, RAM, LINEQS, or LISMOD modeling language in previous sections, the confirmatory factor model considered here is purely a measurement model—that is, there are no functional relationships among factors in the model (beyond the covariances among factors) and hence it is a different model. In the FACTOR statement, you specify factors on the left-hand side of the entries, followed by arrows and the manifest variables that are related to the factors. On the right-hand side of the entries, you specify either parameter names or fixed parameter values for the corresponding factor loadings. In this example, there are three factors with three loadings to estimate. In the PVAR statement, you specify the parameters for factor variances and error variances of manifest variables. In the COV statement, you specify the factor covariances. As compared with the PATH, RAM, LINEQS, or LISMOD, the factor modeling language has more restrictions on parameters. These restrictions are listed as follows:

- factor-factor paths and variable-to-factor paths are not allowed
- error covariances and factor-error covariances are not allowed

For more information about exploratory and confirmatory factor models and the FACTOR modeling language, see the section “The FACTOR Model” on page 1394 or the FACTOR statement on page 1258.
Direct Covariance Structures Analysis

Previous examples are concerned with the implied covariance structures from the functional relationships among manifest and latent variables. In some cases, direct modeling of the covariance structures is not only possible, but indeed more convenient. The MSTRUCT modeling language in PROC CALIS is designed for this purpose. Consider the following four variables from the Wheaton et al. (1977) data:

Anomie67: Anomie 1967
Powerless67: Powerlessness 1967
Anomie71: Anomie 1971
Powerless71: Powerlessness 1971

The covariance structures are hypothesized as follows:

\[
\Sigma = \begin{pmatrix}
\phi_1 & \theta_1 & \theta_2 & \theta_1 \\
\theta_1 & \phi_2 & \theta_1 & \theta_3 \\
\theta_2 & \theta_1 & \phi_1 & \theta_1 \\
\theta_1 & \theta_3 & \theta_1 & \phi_2 \\
\end{pmatrix}
\]

where:

\(\phi_1\): Variance of Anomie
\(\phi_2\): Variance of Powerlessness
\(\theta_1\): Covariance between Anomie and Powerlessness
\(\theta_2\): Covariance between Anomie measures
\(\theta_3\): Covariance between Powerlessness measures

In the hypothesized covariance structures, the variances of Anomie and Powerlessness measures are assumed to stay constant over the two time points. Their covariances are also independent of the time of measurements. To test the tenability of this covariance structure model, you can use the following statements of the MSTRUCT modeling language:

```sas
proc calis nobs=932 data=Wheaton;
   mstruct
       var = Anomie67 Powerless67 Anomie71 Powerless71;
       matrix _COV_ [1,1] = phi1,
                      [2,2] = phi2,
                      [3,3] = phi1,
                      [4,4] = phi2,
                      [2,1] = theta1,
                      [3,1] = theta2,
                      [3,2] = theta1,
                      [4,1] = theta1,
                      [4,2] = theta3,
                      [4,3] = theta1;
   run;
```


In the MSTRUCT statement, you specify the list of variables of interest with the VAR= option. The order of the variables in the list will be the order in the hypothesized covariance matrix. Next, you use the MATRIX _COV_ statement to specify the parameters in the covariance matrix. The specification is a direct translation from the hypothesized covariance matrix. For example, the \( [1, 1] \) element of the covariance matrix is fitted by the free parameter \( \phi_1 \). Depending on the hypothesized model, you can also specify fixed constants for the elements in the covariance matrix. If an element in the covariance matrix is not specified by either a parameter name or a constant, it is assumed to be a fixed zero.

The analysis of this model is carried out in Example 29.19.

The MSTRUCT modeling language appears to be more restrictive than any of the other modeling languages discussed, in regard to the following limitations:

- It does not explicitly support latent variables in modeling.
- It does not explicitly support modeling of linear functional relations among variables (for example, paths).

However, these limitations are more apparent than real. In PROC CALIS, the parameters defined in models can be dependent. These dependent parameters can be defined further as functions of other parameters in the PARAMETERS and the SAS programming statements. With these capabilities, it is possible to fit structural models with latent variables and with linear functional relations by using the MSTRUCT modeling language. However, this requires a certain level of sophistication in statistical knowledge and in programming. Therefore, it is recommended that the MSTRUCT modeling language be used only when the covariance and mean structures are modeled directly.

For more information about the MSTRUCT modeling language, see the section “The MSTRUCT Model” on page 1416 and the MSTRUCT statement on page 1313.
Various modeling languages are supported in PROC CALIS because researchers are trained in or adhere to different schools of modeling. Different modeling languages reflect different modeling terminology and philosophies. The statistical and mathematical consequences by using these various modeling languages, however, might indeed be the same. In other words, you can use more than one modeling languages for certain types of models without affecting the statistical analysis. Given the choices, which modeling language is preferred? There are two guidelines for this:

- Use the modeling language that you are most familiar with.
- Use the most specialized modeling language whenever it is possible.

The first guideline calls for researchers’ knowledge about a particular modeling language. Use the language you know the best. For example, some researchers might find equation input language like LINEQS the most suitable, while others might feel more comfortable using matrix input language like LISMOD.

The second guideline depends on the nature of the model at hand. For example, to specify a factor analysis model in the CALIS procedure, the specialized FACTOR language, instead of the LISMOD language, is recommended. Using a more specialized the modeling language is less error-prone. In addition, using a specialized language like FACTOR in this case amounts to giving the CALIS procedure additional information about the specific mathematical properties of the model. This additional information is used to enhance computational efficiency and to provide more specialized results. Another example is fitting an equi-covariance model. You can simply use the MSTRUCT model specification, in which you specify the same parameter for all off-diagonal elements of the covariance elements. This is direct and intuitive. Alternatively, you could tweak a LINEQS model that would predict the same covariance for all variables. However, this is indirect and error-prone, especially for novice modelers.

In PROC CALIS, the FACTOR and MSTRUCT modeling languages are considered more specialized, while other languages are more general in applications. Whenever possible, you should use the more specialized languages. However, if your model involves some novel covariance or mean structures that are not covered by the more specialized modeling languages, you can consider the more generalized modeling languages. See Example 29.33 for an application of the generalized COSAN model.
Syntax: CALIS Procedure

The following statements are available in the CALIS procedure:

```
PROC CALIS < options > ;
   BOUNDS boundary-constraints ;
   BY variables ;
   COSAN model Specifications ;
   COV covariance-parameters ;
   DETERM variables ;
   EFFPART effects ;
   FACTOR model Specifications ;
   FITINDEX fit-options ;
   FREQ variable ;
   GROUP group-number < / options > ;
   LINCON linear-constraints ;
   LINEQS equations ;
   LISMOD variable-lists ;
   LMTESTS test-options ;
   MATRIX parameter-specifications ;
   MEAN mean-parameters ;
   MODEL model-number < / options > ;
   MSTRUCT variable-list ;
   NLINCON nonlinear-constraints ;
   NLOPTIONS optimization-options ;
   OUTFILES file-options ;
   PARAMETERS parameter-specifications ;
   PARTIAL variables ;
   PATH paths ;
   PATHDIAGRAM < options > ;
   PCOV covariance-parameters ;
   PVAR variance-parameters ;
   RAM model Specifications ;
   REFMODEL model-number < / options > ;
   RENAMEPARM parameter-assignments ;
   SIMTESTS simultaneous-tests ;
   STD variance-parameters ;
   STRUCTEQ variables < label > ;
   TESTFUNC parametric-functions ;
   VAR variables ;
   VARIANCE variance-parameters ;
   VARNAMES name-assignments ;
   WEIGHT variable ;
   Programming statements ;
```
Classes of Statements in PROC CALIS

To better understand the syntax of PROC CALIS, it is useful to classify the statements into classes. These classes of statements are described in the following sections.

PROC CALIS Statement

is the main statement that invokes the CALIS procedure. You can specify options for input and output data sets, printing, statistical analysis, and computations in this statement. The options specified in the PROC CALIS statement will propagate to all groups and models, but are superseded by the options specified in the individual GROUP or MODEL statements.

GROUP Statement

signifies the beginning of a group specification. A group in the CALIS procedure is an independent sample of observations. You can specify options for input and output data sets, printing, and statistical computations in this statement. Some of these group options in the GROUP statement can also be specified in the MODEL or PROC CALIS statement, but the options specified in the GROUP statement supersede those specified in the MODEL or PROC CALIS statement for the group designated in the GROUP statement. For group options that are available in both of the GROUP and PROC CALIS statements, see the section “Options Available in the GROUP and PROC CALIS Statements” on page 1273. For group options that are available in the GROUP, MODEL, and PROC CALIS statements, see the section “Options Available in GROUP, MODEL, and PROC CALIS Statements” on page 1273. If no GROUP statement is used, a single-group analysis is assumed. The group options for a single-group analysis are specified in the PROC CALIS statement.

The GROUP statement can be followed by subsidiary group specification statements, which specify further data processing procedures for the group designated in the GROUP statement.

Subsidiary Group Specification Statements

are for specifying additional data processing attributes for the input data. These statements are summarized in the following table:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREQ on page 1271</td>
<td>Specifies the frequency variable for the input observations</td>
</tr>
<tr>
<td>PARTIAL on page 1322</td>
<td>Specifies the partial variables</td>
</tr>
<tr>
<td>VAR on page 1359</td>
<td>Specifies the set of variables in analysis</td>
</tr>
<tr>
<td>WEIGHT on page 1367</td>
<td>Specifies the weight variable for the input observations</td>
</tr>
</tbody>
</table>

These statements can be used after the PROC CALIS statement or each GROUP statement. Again, the specifications within the scope of the GROUP statement supersede those specified after the PROC CALIS statement for the group designated in the GROUP statement.

MODEL Statement

signifies the beginning of a model specification. In the MODEL statement, you can specify the fitted groups, input and output data sets for model specification or estimates, printing options, statistical analysis, and computational options. Some of the options in the MODEL statement can also be specified in the PROC CALIS statement. These options are called model options. Model options specified in the MODEL statement
supersede those specified in the PROC CALIS statement. For model options that are available in both of the MODEL and PROC CALIS statements, see the section “Options Available in the MODEL and PROC CALIS Statements” on page 1312. If no MODEL statement is used, a single model is assumed and the model options are specified in the PROC CALIS statement.

Some of the options in the MODEL statement can also be specified in the GROUP statement. These options are called group options. The group options in the MODEL statement are transferred to the groups being fitted, but they are superseded by the group options specified in the associated GROUP statement. For group options that are available in the GROUP and the MODEL statements, see the section “Options Available in GROUP, MODEL, and PROC CALIS Statements” on page 1273.

The MODEL statement itself does not define the model being fitted to the data; the main and subsidiary model specification statements that follow immediately after the MODEL statement do. These statements are described in the next two sections.

**Main Model Specification Statements**

are for specifying the type of the modeling language and the main features of the model. These statements are summarized in the following table:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COSAN on page 1242</td>
<td>Specifies general mean and covariance structures in matrix terms</td>
</tr>
<tr>
<td>FACTOR on page 1258</td>
<td>Specifies confirmatory or exploratory factor models</td>
</tr>
<tr>
<td>LINEQS on page 1275</td>
<td>Specifies models by using linear equations</td>
</tr>
<tr>
<td>LISMOD on page 1282</td>
<td>Specifies models in terms of LISREL-like model matrices</td>
</tr>
<tr>
<td>MSTRUCT on page 1313</td>
<td>Specifies parameters directly in the mean and covariance matrices</td>
</tr>
<tr>
<td>PATH on page 1322</td>
<td>Specifies models by using the causal paths of variables</td>
</tr>
<tr>
<td>RAM on page 1348</td>
<td>Specifies models by using RAM-like lists of parameters</td>
</tr>
<tr>
<td>REFMODEL on page 1354</td>
<td>Specifies a base model from which the target model is modified</td>
</tr>
</tbody>
</table>

You can use one of these statements for specifying one model. Each statement in the list represents a particular type of modeling language. After the main model specification statement, you might need to add subsidiary model specification statements, as described in the following section, to complete the model specification.

**Subsidiary Model Specification Statements**

are used to supplement the model specification. They are specific to the types of the modeling languages invoked by the main model specification statements, as shown in the following table:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Specification</th>
<th>Modeling Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>COV on page 1252</td>
<td>Covariance parameters</td>
<td>FACTOR, LINEQS</td>
</tr>
<tr>
<td>MATRIX on page 1295</td>
<td>Parameters in matrices</td>
<td>COSAN, LISMOD, MSTRUCT</td>
</tr>
<tr>
<td>MEAN on page 1309</td>
<td>Mean or intercept parameters</td>
<td>FACTOR, LINEQS, PATH</td>
</tr>
<tr>
<td>PCOV on page 1343</td>
<td>(Partial) covariance parameters</td>
<td>PATH</td>
</tr>
<tr>
<td>PVAR on page 1345</td>
<td>(Partial) variance parameters</td>
<td>FACTOR, PATH</td>
</tr>
<tr>
<td>RENAMETRAPM on page 1356</td>
<td>New parameters by renaming</td>
<td>REFMODEL</td>
</tr>
<tr>
<td>VARIANCE on page 1362</td>
<td>Variance parameters</td>
<td>LINEQS</td>
</tr>
</tbody>
</table>

Notice that the RAM modeling language does not have any subsidiary model specification statements, because all model specification can be done in the RAM statement.
Chapter 29: The CALIS Procedure

Model Analysis Statements

are used to request specific statistical analysis, as shown in the following table:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETERM on page 1256</td>
<td>Sets variable groups for computing the determination coefficients; same as the STRUCTEQ statement</td>
</tr>
<tr>
<td>EFFPART on page 1215</td>
<td>Displays and partitions the effects in the model</td>
</tr>
<tr>
<td>FITINDEX on page 1268</td>
<td>Controls the fit summary output</td>
</tr>
<tr>
<td>LMTESTS on page 1285</td>
<td>Defines the Lagrange multiplier test regions</td>
</tr>
<tr>
<td>SIMTESTS on page 1357</td>
<td>Defines simultaneous parametric function tests</td>
</tr>
<tr>
<td>STRUCTEQ on page 1256</td>
<td>Sets variable groups for computing the determination coefficients; same as the DETERM statement</td>
</tr>
<tr>
<td>TESTFUNC on page 1358</td>
<td>Tests individual parametric functions</td>
</tr>
</tbody>
</table>

Notice that the DETERM and the STRUCTEQ statements function exactly the same way.

Optimization Statements

are used to define additional parameters and parameter constraints, to fine-tune the optimization techniques, or to set the printing options in optimization, as shown in the following table:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOUNDS on page 1241</td>
<td>Defines the bounds of parameters</td>
</tr>
<tr>
<td>LINCON on page 1274</td>
<td>Defines the linear constraints of parameters</td>
</tr>
<tr>
<td>NLINCON on page 1316</td>
<td>Defines the nonlinear constraints of parameters</td>
</tr>
<tr>
<td>NLOPTIONS on page 1316</td>
<td>Sets the optimization techniques and printing options</td>
</tr>
</tbody>
</table>

Other Statements

that are not listed in preceding sections are summarized in the following table:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BY on page 1241</td>
<td>Fits a model to different groups separately</td>
</tr>
<tr>
<td>OUTFILES on page 1317</td>
<td>Controls multiple output data sets</td>
</tr>
<tr>
<td>PARAMETERS on page 1319</td>
<td>Defines additional parameters or superparameters</td>
</tr>
<tr>
<td>SAS programming statements on page 1357</td>
<td>Define parameters or functions</td>
</tr>
</tbody>
</table>

Note that SAS programming statements include the ARRAY statement and the mathematical statements for defining parameter interdependence.
Single-Group Analysis Syntax

PROC CALIS < options > ;
   subsidiary group specification statements ;
   main model specification statement ;
   subsidiary model specification statements ;
   model analysis statements ;
   optimization statements ;
   other statements ;

In a single-group analysis, there is only one group and one model. Because all model or group specifications are unambiguous, the MODEL and GROUP statements are not necessary. The order of the statements is not important for parsing purposes, although you might still like to order them in a particular way to aid understanding. Notice that the OUTFILES statement is not necessary in single-group analyses, as it is designed for multiple-group situations. Output file options in a single-group analysis can be specified in the PROC CALIS statement.

Multiple-Group Multiple-Model Analysis Syntax

PROC CALIS < options > ;
   subsidiary group specification statements ;
   model analysis statements ;
   GROUP 1 < / group options > ;
       subsidiary group specification statements ;
   GROUP 2 < / group options > ;
       subsidiary group specification statements ;
   MODEL 1 < / model options > ;
       main model specification statement ;
       subsidiary model specification statements ;
       model analysis statements ;
   MODEL 2 < / model options > ;
       main model specification statement ;
       subsidiary model specification statements ;
       model analysis statements ;
   optimization statements ;
   other statements ;

The multiple uses of the GROUP and the MODEL statements characterize the multiple-group multiple-model analysis. Unlike the single-group analysis, the order of some statements in a multiple-group multiple-model syntax is important for parsing purposes.

A GROUP statement signifies the beginning of a group specification block and designates a group number for the group. The scope of a GROUP statement extends to the subsequent subsidiary group specification statements until another MODEL or GROUP statement is encountered. In the preceding syntax, GROUP 1 and GROUP 2 have separate blocks of subsidiary group specification statements. By using additional GROUP statements, you can add as many groups as your situation calls for. Subsidiary group specification statements declared before the first GROUP statement are in the scope of the PROC CALIS statement. This
means that these subsidiary group specification statements are applied globally to all groups unless they are respecified locally within the scopes of individual GROUP statements.

A MODEL statement signifies the beginning of a model specification block and designates a model number for the model. The scope of a MODEL statement extends to the subsequent main and subsidiary model specification statements until another MODEL or GROUP statement is encountered. In the preceding syntax, MODEL 1 and MODEL 2 have separate blocks of main and subsidiary model specification statements. By using additional MODEL statements, you can add as many models as your situation calls for. If you use at least one MODEL statement, any main and subsidiary model specification statements declared before the first MODEL statement are ignored.

Some model analysis statements are also bounded by the scope of the MODEL statements. These statements are: DETERM, EFFPART, LMTESTS, and STRUCTEQ. These statements are applied only locally to the model in which they belong. To apply these statements globally to all models, put these statements before the first MODEL statement.

Other model analysis statements are not bounded by the scope of the MODEL statements. These statements are: FITINDEX, SIMTESTS, and TESTFUNC. Because these statements are not model-specific, you can put these statements anywhere in a PROC CALIS run.

Optimization and other statements are not bounded by the scope of either the GROUP or MODEL statements. You can specify them anywhere between the PROC CALIS and the run statements without affecting the parsing of the models and the groups. For clarity of presentation, they are shown as last statement block in the syntax. Notice that the BY statement is not supported in a multiple-group setting.

**PROC CALIS Statement**

```
PROC CALIS <options>;
```

The PROC CALIS statement invokes the CALIS procedure. There are many options in the PROC CALIS statement. These options, together with brief descriptions, are classified into different categories in the next few sections. An alphabetical listing of these options with more details then follows.

**Data Set Options**

You can use the following options to specify input and output data sets:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASEFIT=</td>
<td>Inputs the fit information of the customized baseline model</td>
</tr>
<tr>
<td>DATA=</td>
<td>Inputs the data</td>
</tr>
<tr>
<td>INEST=</td>
<td>Inputs the initial values and constraints</td>
</tr>
<tr>
<td>INMODEL=</td>
<td>Inputs the model specifications</td>
</tr>
<tr>
<td>INWGT=</td>
<td>Inputs the weight matrix</td>
</tr>
<tr>
<td>OUTEST=</td>
<td>Outputs the estimates and their covariance matrix</td>
</tr>
<tr>
<td>OUTFIT=</td>
<td>Outputs the fit indices</td>
</tr>
<tr>
<td>OUTMODEL=</td>
<td>Outputs the model specifications</td>
</tr>
<tr>
<td>OUTSTAT=</td>
<td>Outputs the statistical results</td>
</tr>
<tr>
<td>OUTWGT=</td>
<td>Outputs the weight matrix</td>
</tr>
<tr>
<td>READADDPARM</td>
<td>Inputs the generated default parameters in the INMODEL= data set</td>
</tr>
</tbody>
</table>
Model and Estimation Options

You can use these *options* to specify details about estimation, models, and computations:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORRELATION</td>
<td>Analyzes correlation matrix</td>
</tr>
<tr>
<td>COVARIANCE</td>
<td>Analyzes covariance matrix</td>
</tr>
<tr>
<td>COVPATTERN=</td>
<td>Specifies one of the built-in covariance structures</td>
</tr>
<tr>
<td>DEMPHAS=</td>
<td>Emphasizes the diagonal entries</td>
</tr>
<tr>
<td>EDF=</td>
<td>Defines number of observations by the number of error degrees of freedom</td>
</tr>
<tr>
<td>INWGTINV</td>
<td>Specifies that the INWGT= data set contains the inverse of the weight matrix</td>
</tr>
<tr>
<td>MEANPATTERN=</td>
<td>Specifies one of the built-in mean patterns</td>
</tr>
<tr>
<td>MEANSTR</td>
<td>Analyzes the mean structures</td>
</tr>
<tr>
<td>METHOD=</td>
<td>Specifies the estimation method</td>
</tr>
<tr>
<td>NOBS=</td>
<td>Defines the number of observations</td>
</tr>
<tr>
<td>NOMEANSTR</td>
<td>Deactivates the inherited MEANSTR option</td>
</tr>
<tr>
<td>RANDOM=</td>
<td>Specifies the seed for randomly generated initial values</td>
</tr>
<tr>
<td>RDF=</td>
<td>Defines nobs by the number of regression df</td>
</tr>
<tr>
<td>RIDGE=</td>
<td>Specifies the ridge factor for the covariance matrix</td>
</tr>
<tr>
<td>ROBPHI=</td>
<td>Specifies the tuning parameter for robust methods</td>
</tr>
<tr>
<td>ROBUST=</td>
<td>Specifies the type of robust method</td>
</tr>
<tr>
<td>START=</td>
<td>Specifies a constant for initial values</td>
</tr>
<tr>
<td>VARDEF=</td>
<td>Specifies the variance divisor</td>
</tr>
<tr>
<td>WPENALTY=</td>
<td>Specifies the penalty weight to fit correlations</td>
</tr>
<tr>
<td>WRIDGE=</td>
<td>Specifies the ridge factor for the weight matrix</td>
</tr>
</tbody>
</table>

Options for Fit Statistics

You can use these *options* to modify the default behavior of fit index computations, to control the display of fit indices, and specify output file for fit indices:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHAECV=</td>
<td>Specifies the $\alpha$ level for computing the confidence interval of ECV</td>
</tr>
<tr>
<td></td>
<td>(Browne and Cudeck 1993)</td>
</tr>
<tr>
<td>ALPHARMS=</td>
<td>Specifies the $\alpha$ level for computing the confidence interval of RMSEA</td>
</tr>
<tr>
<td></td>
<td>(Steiger and Lind 1980)</td>
</tr>
<tr>
<td>BASEFUNC=</td>
<td>Specifies the function value and the degrees of freedom of the customized</td>
</tr>
<tr>
<td></td>
<td>baseline model</td>
</tr>
<tr>
<td>CHICORRECT=</td>
<td>Specifies the chi-square correction factor</td>
</tr>
<tr>
<td>CLOSEFIT=</td>
<td>Defines the close fit value</td>
</tr>
<tr>
<td>DFRREduce=</td>
<td>Reduces the degrees of freedom for model fit chi-square test</td>
</tr>
<tr>
<td>NOADJDF</td>
<td>Requests no degrees-of-freedom adjustment be made for active constraints</td>
</tr>
<tr>
<td>NOINDEXTYPE=</td>
<td>Suppresses the printing of fit index types</td>
</tr>
<tr>
<td>OUTFIT=</td>
<td>Specifies the output data set for storing fit indices</td>
</tr>
</tbody>
</table>

These *options* can also be specified in the FITINDEX statement. However, to control the display of individual fit indices, you must use the ON= and OFF= options of the FITINDEX statement.
Options for Statistical Analysis

You can use these options to request specific statistical analysis and display and to set the parameters for statistical analysis:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHA=</td>
<td>Specifies the $\alpha$-level for confidence intervals</td>
</tr>
<tr>
<td>ALPHALEV=</td>
<td>Specifies the $\alpha$-level criterion for detecting leverage points</td>
</tr>
<tr>
<td>ALPHAOUT=</td>
<td>Specifies the $\alpha$-level criterion for detecting outliers</td>
</tr>
<tr>
<td>ASYCOV=</td>
<td>Specifies the formula for computing asymptotic covariances</td>
</tr>
<tr>
<td>BIASKUR</td>
<td>Computes the skewness and kurtosis without bias corrections</td>
</tr>
<tr>
<td>CI</td>
<td>Prints the confidence limits</td>
</tr>
<tr>
<td>EFFPART</td>
<td>TOTEFF</td>
</tr>
<tr>
<td>EXTENDPATH</td>
<td>Displays the extended path estimates</td>
</tr>
<tr>
<td>G4=</td>
<td>Specifies the algorithm for computing standard errors</td>
</tr>
<tr>
<td>KURTOSIS</td>
<td>Computes and displays kurtosis</td>
</tr>
<tr>
<td>MAXLEVERAGE=</td>
<td>Specifies the maximum number of leverage observations to display</td>
</tr>
<tr>
<td>MAXMISSPAT=</td>
<td>Specifies the maximum number of missing patterns to display</td>
</tr>
<tr>
<td>MAXOUTLIER=</td>
<td>Specifies the maximum number of outliers to display</td>
</tr>
<tr>
<td>MODIFICATION</td>
<td>Computes modification indices</td>
</tr>
<tr>
<td>NOMISSPAT</td>
<td>Suppresses the display of missing pattern analysis</td>
</tr>
<tr>
<td>NOMOD</td>
<td>Suppresses modification indices</td>
</tr>
<tr>
<td>NOSTAND</td>
<td>Suppresses the standardized output</td>
</tr>
<tr>
<td>NOSTDERR</td>
<td>Suppresses standard error computations</td>
</tr>
<tr>
<td>PCORR</td>
<td>Displays analyzed and estimated moment matrix</td>
</tr>
<tr>
<td>PCOVES</td>
<td>Displays the covariance matrix of estimates</td>
</tr>
<tr>
<td>PDETERM</td>
<td>Computes the determination coefficients</td>
</tr>
<tr>
<td>PESTIM</td>
<td>Prints parameter estimates</td>
</tr>
<tr>
<td>PINITIAL</td>
<td>Prints initial pattern and values</td>
</tr>
<tr>
<td>PLATCOV</td>
<td>Computes the latent variable covariances and score coefficients</td>
</tr>
<tr>
<td>PLOTS=</td>
<td>Specifies ODS Graphics selection</td>
</tr>
<tr>
<td>PWEIGHT</td>
<td>Displays the weight matrix</td>
</tr>
<tr>
<td>RESIDUAL=</td>
<td>Specifies the type of residuals being computed</td>
</tr>
<tr>
<td>SIMPLE</td>
<td>Prints univariate statistics</td>
</tr>
<tr>
<td>SLMW=</td>
<td>Specifies the probability limit for Wald tests</td>
</tr>
<tr>
<td>STDERR</td>
<td>Computes the standard errors</td>
</tr>
<tr>
<td>TMISSPAT=</td>
<td>Specifies the data proportion threshold for displaying the missing patterns</td>
</tr>
</tbody>
</table>
Global Display Options

There are two different kinds of global display options: one is for selecting output; the other is for controlling the format or order of output.

You can use the following options to select printed output:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOPRINT</td>
<td>Suppresses the displayed output</td>
</tr>
<tr>
<td>PALL</td>
<td>Displays all displayed output (ALL)</td>
</tr>
<tr>
<td>PRINT</td>
<td>Adds default displayed output</td>
</tr>
<tr>
<td>PSHORT</td>
<td>Reduces default output (SHORT)</td>
</tr>
<tr>
<td>PSUMMARY</td>
<td>Displays fit summary only (SUMMARY)</td>
</tr>
</tbody>
</table>

In contrast to individual output printing options described in the section “Options for Statistical Analysis” on page 1202, the global display options typically control more than one output or analysis. The relations between these two types of options are summarized in the following table:

<table>
<thead>
<tr>
<th>Options</th>
<th>PALL</th>
<th>PRINT</th>
<th>default</th>
<th>PSHORT</th>
<th>PSUMMARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>fit indices</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>linear dependencies</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>PESTIM</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>iteration history</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>PINITIAL</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIMPLE</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STDERR</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESIDUAL</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KURTOSIS</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLATCOV</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTEFF</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCORR</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODIFICATION</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWEIGHT</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCOVES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDETERM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRIMAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each column in the table represents a global display option. An “*” in the column means that the individual output or analysis option listed in the corresponding row turns on when the global display option in the corresponding column is specified.

Note that the column labeled with “default” is for default printing. If the NOPRINT option is not specified, a default set of output is displayed. The PRINT and PALL options add to the default output, while the PSHORT and PSUMMARY options reduce from the default output.

Note also that the PCOVES, PDETERM, and PRIMAT options cannot be turned on by any global display options. They must be specified individually.
The following global display options are for controlling formats and order of the output:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOORDERSPEC</td>
<td>Displays model specifications and results according to the input order</td>
</tr>
<tr>
<td>NOPARMNAME</td>
<td>Suppresses the printing of parameter names in results</td>
</tr>
<tr>
<td>ORDERALL</td>
<td>Orders all output displays according to the model numbers, group numbers, and parameter types</td>
</tr>
<tr>
<td>ORDERGROUPS</td>
<td>Orders the group output displays according to the group numbers</td>
</tr>
<tr>
<td>ORDERMODELS</td>
<td>Orders the model output displays according to the model numbers</td>
</tr>
<tr>
<td>ORDERSPEC</td>
<td>Orders the model output displays according to the parameter types within each model</td>
</tr>
<tr>
<td>PARMNAME</td>
<td>Displays parameter names in model specifications and results</td>
</tr>
<tr>
<td>PRIMAT</td>
<td>Displays estimation results in matrix form</td>
</tr>
</tbody>
</table>

**Optimization Options**

You can use the following *options* to control the behavior of the optimization. Most of these *options* are also available in the NLOPTIONS statement.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASINGULAR=</td>
<td>Specifies the absolute singularity criterion for inverting the information matrix</td>
</tr>
<tr>
<td>COVSING=</td>
<td>Specifies the singularity tolerance of the information matrix</td>
</tr>
<tr>
<td>FCONV=</td>
<td>Specifies the relative function convergence criterion</td>
</tr>
<tr>
<td>GCONV=</td>
<td>Specifies the gradient convergence criterion</td>
</tr>
<tr>
<td>INSTEP=</td>
<td>Specifies the initial step length (RADIUS=, SALPHA=)</td>
</tr>
<tr>
<td>LINESEARCH=</td>
<td>Specifies the line-search method</td>
</tr>
<tr>
<td>LSPRECISION=</td>
<td>Specifies the line-search precision (SPRECISION=)</td>
</tr>
<tr>
<td>MAXFUNC=</td>
<td>Specifies the maximum number of function calls</td>
</tr>
<tr>
<td>MAXITER=</td>
<td>Specifies the maximum number of iterations</td>
</tr>
<tr>
<td>MSINGULAR=</td>
<td>Specifies the relative M singularity of the information matrix</td>
</tr>
<tr>
<td>OMETHOD</td>
<td>TECHNIQUE=</td>
</tr>
<tr>
<td>ROBITER=</td>
<td>Specifies the maximum number of iterations for estimating robust covariance and mean matrices</td>
</tr>
<tr>
<td>SINGULAR=</td>
<td>Specifies the singularity criterion for matrix inversion</td>
</tr>
<tr>
<td>UPDATE=</td>
<td>Specifies the update method for some optimization techniques</td>
</tr>
<tr>
<td>VSINGULAR=</td>
<td>Specifies the relative V singularity of information matrix</td>
</tr>
<tr>
<td>XCONV=</td>
<td>Specifies the relative parameter convergence criterion</td>
</tr>
</tbody>
</table>
PROC CALIS Statement Options

**ALPHA=\(\alpha\)**
specifies that interval estimation of parameters be done at the \((1 - \alpha)100\%\) confidence level. The smaller the \(\alpha\) value, the higher the confidence level. By default, \(\alpha = 0.05\), which corresponds to a 95% confidence interval.

**ALPHAECV=\(\alpha\)**
specifies a \((1 - \alpha)100\%\) confidence interval \((0 \leq \alpha \leq 1)\) for the Browne and Cudeck (1993) expected cross-validation index (ECVI). By default, \(\alpha = 0.1\), which corresponds to a 90% confidence interval for the ECVI.

**ALPHALEV=\(\alpha\)**
**ALPHALEVERAGE=\(\alpha\)**
specifies the \(\alpha\)-level criterion for detecting leverage observations (or leverage points) in case-level (observation-level) residual diagnostics. The default ALPHALEV= value is 0.01. An observation is a leverage observation if the \(p\)-value of its squared Mahalanobis distance (M-distance) for its predictor variables (including observed and latent variables) is smaller than the specified \(\alpha\)-level, where the \(p\)-value is computed according to an appropriate theoretical chi-square distribution. The larger the ALPHALEV= value, the more liberal the criterion for detecting leverage observations.

In addition to displaying the leverage observations as defined by the ALPHALEV= criterion, PROC CALIS also displays the next 5 observations with the largest leverage M-distances for reference. However, the total number of observations in the displayed output cannot exceed 30 or the number of original observations, whichever is smaller.

This option is relevant only when residual analysis is requested with the RESIDUAL option and with raw data input.

**ALPHAOUT=\(\alpha\)**
**ALPHAOUTLIER=\(\alpha\)**
specifies the \(\alpha\)-level criterion for detecting outliers in case-level (observation-level) residual diagnostics. The default ALPHAOUT= value is 0.01. An observation is an outlier if the \(p\)-value of its squared residual M-distance is smaller than the specified \(\alpha\)-level, where the \(p\)-value is computed according to an appropriate theoretical chi-square distribution. The larger the ALPHAOUT= value, the more liberal the criterion for detecting outliers.

In addition to displaying the outliers as defined by the ALPHAOUT= criterion, PROC CALIS also displays the next 5 observations with the largest residual M-distances for reference. However, the total number of observations in the displayed output in the displayed output cannot exceed 30 or the number of original observations, whichever is smaller.

This option is relevant only when residual analysis is requested with the RESIDUAL option and with raw data input.

**ALPHARMS=\(\alpha\)**
**ALPHARMSEA=\(\alpha\)**
specifies a \((1 - \alpha)100\%\) confidence interval \((0 \leq \alpha \leq 1)\) for the Steiger and Lind (1980) root mean square error of approximation (RMSEA) coefficient (see Browne and Du Toit 1992). The default value is \(\alpha = 0.1\), which corresponds to a 90% confidence interval for the RMSEA.
ASINGULAR=r

specifies an absolute singularity criterion \( r (r > 0) \), for the inversion of the information matrix, which is needed to compute the covariance matrix. The default value for \( r \) or ASING= is the square root of the smallest positive double precision value.

When inverting the information matrix, the following singularity criterion is used for the diagonal pivot \( d_{j,j} \) of the matrix:

\[
|d_{j,j}| \leq \max(\text{ASING}, \text{VSING} \times |H_{j,j}|, \text{MSING} \times \max(|H_{1,1}|, \ldots, |H_{n,n}|))
\]

where VSING and MSING are the specified values in the VSINGULAR= and MSINGULAR= options, respectively, and \( H_{j,j} \) is the \( j \)th diagonal element of the information matrix. Note that in many cases a normalized matrix \( D^{-1}HD^{-1} \) is decomposed (where \( D^2 = \text{diag}(H) \)), and the singularity criteria are modified correspondingly.

ASYCOV=name

specifies the formula for asymptotic covariances used in the weight matrix \( W \) for WLS and DWLS estimation. The ASYCOV option is effective only if METHOD= WLS or METHOD=DWLS and no INWGT= input data set is specified. The following formulas are implemented:

**BIASED:** Browne (1984) formula (3.4)
biased asymptotic covariance estimates; the resulting weight matrix is at least positive semidefinite. This is the default for analyzing a covariance matrix.

**UNBIASED:** Browne (1984) formula (3.8)
asymptotic covariance estimates corrected for bias; the resulting weight matrix can be indefinite (that is, can have negative eigenvalues), especially for small \( N \).

**CORR:** Browne and Shapiro (1986) formula (3.2)
(identical to De Leeuw (1983) formulas (2,3,4)) the asymptotic variances of the diagonal elements are set to the reciprocal of the value \( r \) specified by the WPENALTY= option (default: \( r=100 \)). This formula is the default for analyzing a correlation matrix.

By default, ASYCOV=BIASED is used for covariance analyses and ASYCOV=CORR is used for correlation analyses. Therefore, in almost all cases you do not need to set the ASYCOV= option once you specify the covariance or correlation analysis by the COV or CORR option.

BASEFIT=SAS-data-set

INBASEFIT=SAS-data-set

inputs the SAS-data-set that contains the fit information of the baseline model of your choice. This customized baseline model replaces the default uncorrelatedness model for computing several fit indices of your target model. Typically, you create the BASEFIT= data set by using the OUTFIT= option in a previous PROC CALIS fitting of your customized baseline model. Using the BASEFIT= option assumes that you fit your customized baseline model and your target model with the same data, number of groups (for multiple-group analysis), and estimation method. Typically, your baseline model should be more restrictive (or have fewer parameters) than your target model.

For example, the following statements use the compound symmetry model (COVPATTERN=COMPSYM) as the customized baseline model for the subsequent factor model with two factors:
The fit information of the customized baseline model is saved as an OUTFIT= data set called outf, which is then read as a BASEFIT= data set in the next PROC CALIS run for fitting the target factor model. Notice that in this example the baseline model and the target factor model use the same data set, abc, and the same GLS estimation method.

Alternatively, you can use the BASEFUNC= option to input the function value and the degrees of freedom of your customized baseline model. See the BASEFUNC= option for details. The BASEFIT= option is ignored if you also specify the BASEFUNC= option.

Notice that the fit information in the BASEFIT= data set does not affect the computation of all fit indices. Mainly, it affects the incremental fit indices, because these indices are defined as the incremental fit of a target model over a baseline model. Among all absolute and parsimonious fit indices, only the parsimonious goodness-of-fit (PGFI) index (Mulaik et al. 1989) is affected by the input fit information provided in the BASEFIT= data set.

If you specify METHOD=LSDWLS, LSFIML, LSGLS, or LSML for your target model, the fit information in the BASEFIT= data set is assumed to have been obtained from the DWLS, FIML, GLS, or ML estimation of your customized baseline model. Hence, the fit information in the BASEFIT= data set applies only to the second estimation of your target model. The unweighted least squares (ULS) estimation of the target model still uses the uncorrelatedness model as the baseline model for computing fit indices.

If you use a BASEFIT= data set to input the fit information of your customized baseline model in a multiple-group analysis, then the baseline model function values, chi-squares, and degrees of freedom for individual groups are not known and hence not displayed in the multiple-group fit summary table. The Bentler-Bonett normed fit index (NFI) is also not displayed in the multiple-group fit summary table, because the baseline model function values for individual groups are not saved in the BASEFIT= data set.

\[
\text{BASEFUNC}=r(<\text{DF}=>)\]
\[
\text{BASEFUNC}(<\text{DF}=>)=r
\]
inputs the fit function value \( r \) and the degrees of freedom \( i \) of the baseline model of your choice. This customized baseline model replaces the default uncorrelatedness model for computing several fit indices of your target model. To use this option, you must first fit your customized baseline model and then use this option to input the baseline model fit information when you fit your target model.

Using the BASEFUNC= option assumes that you fit your customized baseline model and your target model with the same data, number of groups (for multiple-group analysis), and estimation method. Typically, your baseline model should be more restrictive (or have fewer parameters) than your target model.

For example, assume that after fitting a customized baseline model you find that the function value of the baseline model is 20.54 and the model degrees of freedom are 15. The following code inputs the
function value and the degrees of freedom of the customized baseline model by using the BASEFUNC= option:

```plaintext
proc calis data=abc basefunc(df=15)=20.54;
  path
    f1 ===> x1-x5 = 1.,
    f2 ===> x6-x10 = 1.,
    f1 ===> f2;
run;
```

You can use the following equivalent syntax to provide the same baseline model fit information:

```plaintext
basefunc(df=15)=20.54
basefunc(15)=20.54
basefunc=20.54(df=15)
basefunc=20.54(15)
```

It is emphasized here that you should input the fit function value, but not the model fit chi-square value, of the baseline model in the BASEFUNC= option. For all estimation methods except the full information maximum likelihood (FIML) method in PROC CALIS, the model fit chi-square values are some multiples of the fit function values. See the section “Estimation Criteria” on page 1463 for the definitions of the various fit functions that are assumed by the BASEFUNC= option.

Alternatively, it might be easier to use the BASEFIT= option to specify the SAS data set that contains the baseline model fit information. Such a SAS data set is typically created by using the OUTFIT= option in the PROC CALIS fitting of your customized baseline model. See the BASEFIT= option for details. However, the BASEFIT= option is ignored if you also specify the BASEFUNC= option.

Notice that the specified values in the BASEFUNC= option do not affect the computation of all fit indices. Mainly, they affect the incremental fit indices, because these indices are defined as the incremental fit of a target model over a baseline model. Among all absolute and parsimonious fit indices, only the parsimonious goodness-of-fit (PGFI) index (Mulaik et al. 1989) is affected by the values provided in the BASEFUNC= option.

If you specify METHOD=LSDWLS, LSFIML, LSGLS, or LSML for your target model, the fit information provided by the BASEFUNC= option is assumed to have been obtained from the DWLS, FIML, GLS, or ML estimation of your customized baseline model. Hence, the fit information provided by the BASEFUNC= option applies only to the second estimation of your target model. The unweighted least squares (ULS) estimation of the target model still uses the uncorrelatedness model as the baseline model for computing fit indices.

If you use the BASEFUNC= option to input the fit information of your customized baseline model in a multiple-group analysis, then the baseline model function values, chi-squares, and degrees of freedom for individual groups are not known and hence not displayed in the multiple-group fit summary table. The Bentler-Bonett NFI is also not displayed in the multiple-group fit summary table, because the baseline model function values for individual groups are not provided with the BASEFUNC= option.
CI
CL
prints the confidence intervals or limits for parameters. By default, PROC CALIS estimates the lower and upper confidence limits at the 95% confidence level, which corresponds to the default value of the ALPHA= option (0.05). You can change the default level of confidence by specifying the ALPHA= option.

BIASKUR
computes univariate skewness and kurtosis by formulas uncorrected for bias.

See the section “Measures of Multivariate Kurtosis” on page 1504 for more information.

CHICORRECT= name | c

CHICORR= name | c
specifies a correction factor $c$ for the chi-square statistics for model fit. You can specify a name for a built-in correction factor or a value between 0 and 1 as the CHICORRECT= value. The model fit chi-square statistic is computed as:

$$\chi^2 = (1 - c)(N - k)F$$

where $N$ is the total number of observations, $k$ is the number of independent groups, and $F$ is the optimized function value. Application of these correction factors requires appropriate specification of the covariance structural model suitable for the chi-square correction. For example, using CHICORRECT=UNCORR assumes that you are fitting a covariance structure with free parameters on the diagonal elements and fixed zeros off-diagonal elements of the covariance matrix. Because all the built-in correction factors assume multivariate normality in their derivations, the appropriateness of applying these built-in chi-square corrections to estimation methods other than METHOD=ML is not known.

Valid names for the CHICORRECT= value are as follows:

**COMPSYM | EQVARCOV** specifies the correction factor due to Box (1949) for testing equal variances and equal covariances in a covariance matrix. The correction factor is:

$$c = \frac{p(p + 1)^2(2p - 3)}{6n(p - 1)(p^2 + p - 4)}$$

where $p$ ($p > 1$) represents the number of variables and $n = (N - 1)$, with $N$ denoting the number of observations in a single group analysis. This option is not applied when you also analyze the mean structures or when you fit multiple-group models.

**EQCOVMAT** specifies the correction factor due to Box (1949) for testing equality of covariance matrices. The correction factor is:

$$c = \frac{2p^2 + 3p - 1}{6(p + 1)(k - 1)} \left( \frac{k}{\sum_{i=1}^{k} \frac{1}{n_i}} - \frac{1}{\sum_{i=1}^{k} \frac{1}{n_i}} \right)$$

where $p$ represents the number of variables, $k$ ($k > 1$) represents the number of groups, and $n_i = (N_i - 1)$, with $N_i$ denoting the number of observations in the $i$th group. This option is not applied when you also analyze the mean structures or when you fit single-group models.
**FIXCOV** specifies the correction factor due to Bartlett (1954) for testing a covariance matrix against a hypothetical fixed covariance matrix. The correction factor is:

\[ c = \frac{1}{6n}(2p + 1 - \frac{2}{p + 1}) \]

where \( p \) represents the number of variables and \( n = (N - 1) \), with \( N \) denoting the number of observations in a single group analysis. This option is not applied when you also analyze the mean structures or when you fit multiple-group models.

**SPHERICITY** specifies the correction factor due to Box (1949) for testing a spherical covariance matrix (Mauchly 1940). The correction factor is:

\[ c = \frac{2p^2 + p + 2}{6np} \]

where \( p \) represents the number of variables and \( n = (N - 1) \), with \( N \) denoting the number of observations in a single group analysis. This option is not applied when you also analyze the mean structures or when you fit multiple-group models.

**TYPEH** specifies the correction factor for testing the H pattern (Huynh and Feldt 1970) directly. The correction factor is:

\[ c = \frac{2p^2 - 3p + 3}{6n(p - 1)} \]

where \( p \) (\( p > 1 \)) represents the number of variables and \( n = (N - 1) \), with \( N \) denoting the number of observations in a single group analysis. This option is not applied when you also analyze the mean structures or when you fit multiple-group models.

This correction factor is derived by substituting \( p \) with \( p - 1 \) in the correction formula applied to Mauchly’s sphericity test. The reason is that testing the H pattern of \( p \) variables is equivalent to testing the sphericity of the \( (p - 1) \) orthogonal contrasts of the same set of variables (Huynh and Feldt 1970). See pp. 295–296 of Morrison (1990) for more details.

**UNCORR** specifies the correction factor due to Bartlett (1950) and Box (1949) for testing a diagonal pattern of a covariance matrix, while the diagonal elements (variances) are unconstrained. This test is sometimes called Bartlett’s test of sphericity—not to be confused with the sphericity test due to Mauchly (1940), which requires all variances in the covariance matrix to be equal. The correction factor is:

\[ c = \frac{2p + 5}{6n} \]

where \( p \) represents the number of variables and \( n = (N - 1) \), with \( N \) denoting the number of observations in a single group analysis. This option is not applied when you also analyze the mean structures or when you fit multiple-group models.
CLOSEFIT=$p$
defines the criterion value $p$ for indicating a close fit. The smaller the better fit. The default value for close fit is .05.

**CORRELATION**
**CORR**
analyzes the correlation matrix, instead of the default covariance matrix. See the **COVARIANCE** option for more details.

**COVARIANCE**
**COV**
analyzes the covariance matrix. Because this is also the default analysis in PROC CALIS, you can simply omit this option when you analyze covariance rather than correlation matrices. If the DATA= input data set is a TYPE=CORR data set (containing a correlation matrix and standard deviations), the default COV option means that the covariance matrix is computed and analyzed.

Unlike many other SAS/STAT procedures (for example, the FACTOR procedure) that analyze correlation matrices by default, PROC CALIS uses a different default because statistical theories of structural equation modeling or covariance structure analysis are mostly developed for covariance matrices. You must use the **CORR** option if correlation matrices are analyzed.

**COVPATTERN=**name
**COVPAT=**name
specifies one of the built-in covariance structures for the data. The purpose of this option is to fit some commonly-used direct covariance structures efficiently without the explicit use of the MSTRUCT model specifications. With this option, the covariance structures are defined internally in PROC CALIS. The following *names* for the built-in covariance structures are supported:

**COMPSYM | EQVARCOV** specifies the compound symmetry pattern for the covariance matrix. That is, a covariance matrix with equal variances for all variables and equal covariance between any pairs of variables (EQVARCOV). For example, if there are four variables in the analysis, the covariance pattern generated by PROC CALIS has the following form:

$$
\Sigma = \begin{pmatrix}
    v & c & c & c \\
    c & v & c & c \\
    c & c & v & c \\
    c & c & c & v
\end{pmatrix}
$$

PROC CALIS denotes the common variance parameter, $v$, by _varparm, and the common covariance parameter, $v$, by _covparm.

If you request a single-group maximum likelihood (METHOD=ML) covariance structure analysis by specifying the **COVPATTERN=COMPSYM** or **COVPATTERN=EQVARCOV** option and the mean structures are not modeled, the chi-square correction due to Box (1949) is applied automatically when the number of variables is greater than or equal to 2. See the **CHICORRECT=COMPSYM** option for the definition of the correction factor.
EQCOVMAT specifies the equality of covariance matrices between multiple groups. That is, this option tests the null hypothesis that

$$H_0 : \Sigma_1 = \Sigma_2 = \ldots = \Sigma_k = \Sigma$$

where $\Sigma$ is a common covariance matrix for the $k \Sigma_j$’s ($j = 1, \ldots, k; k > 1$). The elements of $\Sigma$ are named \_cov_xx_yy automatically by PROC CALIS, where xx and yy represents the row and column numbers such that xx is at least as large as yy. For example, if there are four variables in the analysis, the (1,1) element of $\Sigma$ is denoted by \_cov_1_1, the (4,3) or (3,4) element of $\Sigma$ is denoted by \_cov_4_3, and so on.

If you request a multiple-group maximum likelihood (METHOD=ML) covariance structure analysis by specifying the COVPATTERN=EQCOVMAT and the mean structures are not modeled, the chi-square correction due to Box (1949) is applied automatically. See the CHICORRECT=EQCOVMAT option for the definition of the correction factor.

SATURATED specifies a saturated covariance structure model. This is the default option when you specify the MEANPATTERN= option without using the COVPATTERN= option. The elements of $\Sigma$ are named \_cov_xx_yy automatically by PROC CALIS, where xx represents the row number and yy represents the column number. For example, if there are three variables in the analysis, the (1,1) element in $\Sigma$ is denoted by \_cov_1_1, the (3,2) or (2,3) element in $\Sigma$ is denoted by \_cov_3_2, and so on.

SPHERICITY | SIGSQI specifies the spheric pattern of the covariance matrix (Mauchly 1940). That is, this option tests the null hypothesis that

$$H_0 : \Sigma = \sigma^2 I$$

where $\sigma^2$ is a common variance parameter and $I$ is an identity matrix. For example, if there are three variables in the analysis, the covariance pattern generated by PROC CALIS is:

$$\Sigma = \begin{pmatrix} v & 0 & 0 \\ 0 & v & 0 \\ 0 & 0 & v \end{pmatrix}$$

PROC CALIS denotes the common variance parameter, $v$, by \_varparm.

If you request a single-group maximum likelihood (METHOD=ML) covariance structure analysis by specifying the COVPATTERN=SPHERICITY or COVPATTERN=SIGSQI option and the mean structures are not modeled, the chi-square correction due to Box (1949) is applied automatically. See the CHICORRECT=SPHERICITY option for the definition of the correction factor.

UNCORR | DIAG specifies the diagonal pattern of the covariance matrix. That is, this option tests the null hypothesis of uncorrelatedness—all correlations (or covariances) between variables are zero and the variances are unconstrained. For example, if there are three variables in the analysis, the covariance pattern generated by PROC CALIS is:

$$\Sigma = \begin{pmatrix} v_1 & 0 & 0 \\ 0 & v_2 & 0 \\ 0 & 0 & v_3 \end{pmatrix}$$
PROC CALIS denotes the variance parameters $v_1$, $v_2$, and $v_3$ by \_varparm\_1, \_varparm\_2, and \_varparm\_3\_\_mdl\_xx, respectively.

If you request a single-group maximum likelihood (METHOD=ML) covariance structure analysis by specifying the COVPATTERN=UNCORR or COVPATTERN=DIAG option and the mean structures are not modeled, the chi-square correction due to Bartlett (1950) is applied automatically. See the CHICORRECT=UNCORR option for the definition of the correction factor. Under the multivariate normal assumption, COVPATTERN=UNCORR is also a test of independence of the variables in the analysis.

When you specify the covariance structure model by means of the COVPATTERN= option, you can define the set of variables in the analysis by the VAR statement (either within the scope of the PROC CALIS statement or the GROUP statements). If the VAR statement is not used, PROC CALIS uses all numerical variables in the data sets.

Except for the EQCOVMAT pattern, all other built-in covariance patterns are primarily designed for single-group analysis. However, you can still use these covariance pattern options for multiple-group situations. For example, consider the following three-group analysis:

```
proc calis covpattern=compsym;
  group 1 / data=set1;
  group 2 / data=set2;
  group 3 / data=set3;
run;
```

In this specification, all three groups are fitted by the compound symmetry pattern. However, there would be no constraints across these groups. PROC CALIS generates two distinct parameters for each group: \_varparm\_mdl1 and \_covparm\_mdl1\_xx for Group 1, \_varparm\_mdl2 and \_covparm\_mdl2\_xx for Group 2, and \_varparm\_mdl3 and \_covparm\_mdl3\_xx for Group 3. Similarly, the \_mdlxx\_xx suffix, where xx represents the model number, is applied to the parameters defined by the SATURATED, SPHERICITY (or SIGSQI), and UNCORR (or DIAG) covariance patterns in multiple-group situations. However, chi-square correction, whenever it is applicable to single-group analysis, is not applied to such multiple-group analyses.

You can also apply the COVPATTERN= option partially to the groups in the analysis. For example, the following statements apply the spheric pattern to Group 1 and Group 2 only:

```
proc calis covpattern=sphericity;
  group 1 / data=set1;
  group 2 / data=set2;
  group 3 / data=set3;
  model 3 / group=3;
    path  x1 ===> y3;
run;
```

Group 3 is fitted by Model 3, which is specified explicitly by a PATH model with distinct covariance structures.

If the EQCOVMAT pattern is specified instead, as shown in the following statements, the equality of covariance matrices still holds for Groups 1 and 2:
Chapter 29: The CALIS Procedure

```
proc calis covpattern=eqcovmat;
  group 1 / data=set1;
  group 2 / data=set2;
  group 3 / data=set3;
  model 3 / group=3;
    path  x1 ===> y3;
run;
```

However, Group 3 has its own covariance structures as specified in Model 3. In this case, the chi-square correction due to Box (1949) is not applied because the null hypothesis is no longer testing the equality of covariance matrices among the groups in the analysis.

Use the MEANPATTERN= option if you also want to analyze some built-in mean structures along with the covariance structures.

**COVSING=r**

specifies a nonnegative threshold \( r \), which determines whether the eigenvalues of the information matrix are considered to be zero. If the inverse of the information matrix is found to be singular (depending on the VSINGULAR=, MSINGULAR=, ASINGULAR=, or SINGULAR= option), a generalized inverse is computed using the eigenvalue decomposition of the singular matrix. Those eigenvalues smaller than \( r \) are considered to be zero. If a generalized inverse is computed and you do not specify the NOPRINT option, the distribution of eigenvalues is displayed.

**DATA=SAS-data-set**

specifies an input data set that can be an ordinary SAS data set or a specially structured TYPE=CORR, TYPE=COV, TYPE=UCORR, TYPE=UCOV, TYPE=SSCP, or TYPE=FACTOR SAS data set, as described in the section “Input Data Sets” on page 1367. If the DATA= option is omitted, the most recently created SAS data set is used.

**DEMPHAS=r**

**DE=r**

changes the initial values of all variance parameters by the relationship:

\[
s_{new} = r(|s_{old}| + 1)
\]

where \( s_{new} \) is the new initial value and \( s_{old} \) is the original initial value. The value of \( r \) must be positive. If you specify an \( r \) value less than \( 1E-8 \), it is replaced with \( 1E-8 \).

The initial values of all variance parameters should always be nonnegative to generate positive definite predicted model matrices in the first iteration. By using values of \( r > 1 \), for example, \( r = 2 \), \( r = 10 \), and so on, you can increase these initial values to produce predicted model matrices with high positive eigenvalues in the first iteration. The DEMPHAS= option is effective independent of the way the initial values are set; that is, it changes the initial values set in the model specification as well as those set by an INMODEL= data set and those automatically generated for the FACTOR, LINEQS, LISMOD, PATH, or RAM models. It also affects the initial values set by the START= option, which uses, by default, DEMPHAS=100 if a covariance matrix is analyzed and DEMPHAS=10 for a correlation matrix.
DFREDUCE=i
DFRED=i

reduces the degrees of freedom of the model fit $\chi^2$ test by $i$. In general, the number of degrees of freedom is the total number of nonredundant elements in all moment matrices minus the number of parameters, $t$. Because negative values of $i$ are allowed, you can also increase the number of degrees of freedom by using this option.

EDF=n
DFE=n

makes the effective number of observations $n + 1$. You can also use the NOBS= option to specify the number of observations.

EFFPART
TOTEFF
TE

computes and displays total, direct, and indirect effects for the unstandardized and standardized estimation results. Standard errors for the effects are also computed. Note that this displayed output is not automatically included in the output generated by the PALL option.

Note also that in some situations computations of total effects and their partitioning are not appropriate. While total and indirect effects must converge in recursive models (models with no cyclic paths among variables), they do not always converge in nonrecursive models. When total or indirect effects do not converge, it is not appropriate to partition the effects. Therefore, before partitioning the total effects, the convergence criterion must be met. To check the convergence of the effects, PROC CALIS computes and displays the “stability coefficient of reciprocal causation”—that is, the largest modulus of the eigenvalues of the $\mathbf{ \beta }$ matrix, which is the square matrix that contains the path coefficients of all endogenous variables in the model. Stability coefficients less than one provide a necessary and sufficient condition for the convergence of the total and the indirect effects. Otherwise, PROC CALIS does not show results for total effects and their partitioning. See the section “Stability Coefficient of Reciprocal Causation” on page 1500 for more information about the computation of the stability coefficient.

EXTENDPATH
GENPATH

displays the extended path estimates such as the variances, covariances, means, and intercepts in the table that contains the ordinary path effect (coefficient) estimates. This option applies to the PATH model only.

FCONV=r
FTOL=r

specifies the relative function convergence criterion. The optimization process is terminated when the relative difference of the function values of two consecutive iterations is smaller than the specified value of $r$; that is,

$$\frac{|f(x^{(k)}) - f(x^{(k-1)})|}{\max(|f(x^{(k-1)})|,FSIZE)} \leq r$$

where $FSIZE$ can be defined by the FSIZE= option in the NLOPTIIONS statement. The default value is $r = 10^{-\text{FDIGITS}}$, where $\text{FDIGITS}$ either can be specified in the NLOPTIIONS statement or is set by default to $-<\log_{10}(\epsilon)$, where $\epsilon$ is the machine precision.
Chapter 29: The CALIS Procedure

G4=i

instructs that the algorithm to compute the approximate covariance matrix of parameter estimates used for computing the approximate standard errors and modification indices when the information matrix is singular. If the number of parameters $t$ used in the model you analyze is smaller than the value of $i$, the time-expensive Moore-Penrose (G4) inverse of the singular information matrix is computed by eigenvalue decomposition. Otherwise, an inexpensive pseudo (G1) inverse is computed by sweeping. By default, $i = 60$.

See the section “Estimation Criteria” on page 1463 for more details.

GCONV=r

specifies the relative gradient convergence criterion.

Termination of all techniques (except the CONGRA technique) requires the following normalized predicted function reduction to be smaller than $r$. That is,

$$\frac{[g(x^{(k)})][G^{(k)}]^{-1}g(x^{(k)})}{\max(|f(x^{(k)})|, FSIZE)} \leq r$$

where $FSIZE$ can be defined by the FSIZE= option in the NLOPTIONS statement. For the CONGRA technique (where a reliable Hessian estimate $G$ is not available),

$$\frac{\| g(x^{(k)}) \|_2^2 \| s(x^{(k)}) \|_2}{\| g(x^{(k)}) - g(x^{(k-1)}) \|_2 \max(|f(x^{(k)})|, FSIZE)} \leq r$$

is used. The default value is $r = 10^{-8}$.

INEST=SAS-data-set

INVAR=SAS-data-set

ESTDATA=SAS-data-set

specifies an input data set that contains initial estimates for the parameters used in the optimization process and can also contain boundary and general linear constraints on the parameters. Typical applications of this option are to specify an OUTEST= data set from a previous PROC CALIS analysis. The initial estimates are taken from the values of the PARMS observation in the INEST= data set.

INMODEL=SAS-data-set

INRAM=SAS-data-set

specifies an input data set that contains information about the analysis model. A typical use of the INMODEL= option is when you run an analysis with its model specifications saved as an OUTMODEL= data set from a previous PROC CALIS run. Instead of specifying the main or subsidiary model specification statements in the new run, you use the INMODEL= option to input the model specification saved from the previous run.

Sometimes, you might create an INMODEL= data set from modifying an existing OUTMODEL= data set. However, editing and modifying OUTMODEL= data sets requires good understanding of the formats and contents of the OUTMODEL= data sets. This process could be error-prone for novice users. For details about the format of INMODEL= or OUTMODEL= data sets, see the section “Input Data Sets” on page 1367.

It is important to realize that INMODEL= or OUTMODEL= data sets contain only the information about the specification of the model. These data sets do not store any information about the bounds on
parameters, linear and nonlinear parametric constraints, and programming statements for computing dependent parameters. If required, these types of information must be provided in the corresponding statement specifications (for example, BOUNDS, LINCONE, and so on) in addition to the INMODEL= data set.

An OUTMODEL= data set might also contain default parameters added automatically by PROC CALIS from a previous run (for example, observations with _TYPE_=ADDCOV, ADDMEAN, or ADDPVAR). When reading the OUTMODEL= model specification as an INMODEL= data set in a new run, PROC CALIS ignores these added parameters so that the model being read is exactly like the previous PROC CALIS specification (that is, before default parameters were added automatically). After interpreting the specification in the INMODEL= data set, PROC CALIS will then add default parameters appropriate to the new run. The purpose of doing this is to avoid inadvertent parameter constraints in the new run, where another set of automatic default parameters might have the same generated names as those of the generated parameter names in the INMODEL= data set.

If you want the default parameters in the INMODEL= data set to be read as a part of model specification, you must also specify the READADDPARM option. However, using the READADDPARM option should be rare.

**INSTEP=r**

For highly nonlinear objective functions, such as the EXP function, the default initial radius of the trust-region algorithms (TRUREG, DBLDOG, and LEVMAR) or the default step length of the line-search algorithms can produce arithmetic overflows. If an arithmetic overflow occurs, specify decreasing values of $0 < r < 1$ such as INSTEP=1E–1, INSTEP=1E–2, INSTEP=1E–4, and so on, until the iteration starts successfully.

- For trust-region algorithms (TRUREG, DBLDOG, and LEVMAR), the INSTEP option specifies a positive factor for the initial radius of the trust region. The default initial trust-region radius is the length of the scaled gradient, and it corresponds to the default radius factor of $r = 1$.
- For line-search algorithms (NEWRAP, CONGRA, and QUANEW), INSTEP specifies an upper bound for the initial step length for the line search during the first five iterations. The default initial step length is $r = 1$.

For more details, see the section “Computational Problems” on page 1515.

**INWGT< (INV) >=SAS-data-set**

**INWEIGHT< (INV) >=SAS-data-set**

specifies an input data set that contains the weight matrix $W$ used in generalized least squares (GLS), weighted least squares (WLS, ADF), or diagonally weighted least squares (DWLS) estimation, if you do not specify the INV option at the same time. The weight matrix must be positive definite because its inverse must be defined in the computation of the objective function. If the weight matrix $W$ defined by an INWGT= data set is not positive definite, it can be ridged using the WRIDGE= option. See the section “Estimation Criteria” on page 1463 for more information. If you specify the INWGT(INV)= option, the INWGT= data set contains the inverse of the weight matrix, rather than the weight matrix itself. Specifying the INWGT(INV)= option is equivalent to specifying the INWGT= and INWGTINV options simultaneously. With the INWGT(INV)= specification, the input matrix is not required to be positive definite. See the INWGTINV option for more details. If no INWGT= data set is specified, default settings for the weight matrices are used in the estimation process. The INWGT= data set is described in the section “Input Data Sets” on page 1367. Typically, this input data set is an OUTWGT= data set from a previous PROC CALIS analysis.
INWGTINV
specifies that the INWGT= data set contains the inverse of the weight matrix, rather than the weight matrix itself. This option is effective only with an input weight matrix specified in the INWGT= data set and with the generalized least squares (GLS), weighted least squares (WLS or ADF), or diagonally weighted least squares (DWLS) estimation. With this option, the input matrix provided in the INWGT= data set is not required to be positive definite. Also, the ridging requested by the WRIDGE= option is ignored when you specify the INWGTINV option.

KURTOSIS
KU
computes and displays univariate kurtosis and skewness, various coefficients of multivariate kurtosis, and the numbers of observations that contribute most to the normalized multivariate kurtosis. See the section “Measures of Multivariate Kurtosis” on page 1504 for more information. Using the KURTOSIS option implies the SIMPLE display option. This information is computed only if the DATA= data set is a raw data set, and it is displayed by default if the PRINT option is specified. The multivariate least squares kappa and the multivariate mean kappa are displayed only if you specify METHOD=WLS and the weight matrix is computed from an input raw data set. All measures of skewness and kurtosis are corrected for the mean. Using the BIASKUR option displays the biased values of univariate skewness and kurtosis.

LINESEARCH | LIS | SMETHOD | SM=i
LIS=i
SMETHOD=i
SM=i
specifies the line-search method for the CONGRA, QUANEW, and NEWRAP optimization techniques. See Fletcher (1980) for an introduction to line-search techniques. The value of \( i \) can be any integer between 1 and 8, inclusively; the default is \( i=2 \).

1 specifies a line-search method that needs the same number of function and gradient calls for cubic interpolation and cubic extrapolation; this method is similar to one used by the Harwell subroutine library.

2 specifies a line-search method that needs more function calls than gradient calls for quadratic and cubic interpolation and cubic extrapolation; this method is implemented as shown in Fletcher (1987) and can be modified to an exact line search by using the LSPRECISION= option.

3 specifies a line-search method that needs the same number of function and gradient calls for cubic interpolation and cubic extrapolation; this method is implemented as shown in Fletcher (1987) and can be modified to an exact line search by using the LSPRECISION= option.

4 specifies a line-search method that needs the same number of function and gradient calls for stepwise extrapolation and cubic interpolation.

5 specifies a line-search method that is a modified version of LIS=4.
**LSPRECISION | LSP=r**

LSP=r

**SPRECISION=r**

SP=r

specifies the degree of accuracy that should be obtained by the line-search algorithms LIS=2 and LIS=3. Usually an imprecise line search is inexpensive and successful. For more difficult optimization problems, a more precise and more expensive line search might be necessary (Fletcher 1980, p. 22). The second (default for NEWRAP, QUANEW, and CONGRA) and third line-search methods approach exact line search for small LSPRECISION= values. If you have numerical problems, you should decrease the LSPRECISION= value to obtain a more precise line search. The default LSPRECISION= values are displayed in the following table.

<table>
<thead>
<tr>
<th>OMETHOD=</th>
<th>UPDATE=</th>
<th>LSP default</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUANEW</td>
<td>DBFGS, BFGS</td>
<td>r = 0.4</td>
</tr>
<tr>
<td>QUANEW</td>
<td>DDFP, DFP</td>
<td>r = 0.06</td>
</tr>
<tr>
<td>CONGRA</td>
<td>all</td>
<td>r = 0.1</td>
</tr>
<tr>
<td>NEWRAP</td>
<td>no update</td>
<td>r = 0.9</td>
</tr>
</tbody>
</table>

For more details, see Fletcher (1980, pp. 25–29).

**MAXFUNC | MAXFU=i**

specifies the maximum number i of function calls in the optimization process. The default values are displayed in the following table.

<table>
<thead>
<tr>
<th>OMETHOD=</th>
<th>MAXFUNC default</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVMAR, NEWRAP, NRRIDG, TRUREG</td>
<td>i = 125</td>
</tr>
<tr>
<td>DBLDOG, QUANEW</td>
<td>i = 500</td>
</tr>
<tr>
<td>CONGRA</td>
<td>i = 1000</td>
</tr>
</tbody>
</table>

The default is used if you specify MAXFUNC=0. The optimization can be terminated only after completing a full iteration. Therefore, the number of function calls that is actually performed can exceed the number that is specified by the MAXFUNC= option.

**MAXITER=i<n>**

MAXIT=i<n>

specifies the maximum number i of iterations in the optimization process. Except for the iteratively reweighted least squares (IRLS) algorithm for the robust estimation of model parameters, the default values are displayed in the following table.

<table>
<thead>
<tr>
<th>OMETHOD=</th>
<th>MAXITER default</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVMAR, NEWRAP, NRRIDG, TRUREG</td>
<td>i = 50</td>
</tr>
<tr>
<td>DBLDOG, QUANEW</td>
<td>i = 200</td>
</tr>
<tr>
<td>CONGRA</td>
<td>i = 400</td>
</tr>
</tbody>
</table>

The default maximum number of iterations for IRLS is 5000. The default value is used if you specify MAXITER=0 or if you omit the MAXITER option.
The optional second value \( n \) is valid only for \( \text{OMETHOD=QUANEW} \) with nonlinear constraints. It specifies an upper bound \( n \) for the number of iterations of an algorithm and reduces the violation of nonlinear constraints at a starting point. The default is \( n = 20 \). For example, specifying

\[
\text{maxiter} = .0
\]

means that you do not want to exceed the default number of iterations during the main optimization process and that you want to suppress the feasible point algorithm for nonlinear constraints.

**MAXLEVERAGE**=\( n \)

**MAXLEV**=\( n \)
specifies the maximum number of leverage observations to display in the output, where \( n \) is between 1 and 9,999. The default MAXLEVERAGE= value is 30. The actual numbers of leverage observations in the output could be smaller than the maximum. In general, PROC CALIS finds the number leverage points \( m \) and then adds the next 5 most leveraged observations in the output. The actual number of leverage observations shown in the output is either \( m+5 \), the MAXLEVERAGE= value, or the number of observations in the data set, whichever is smaller.

**MAXMISSPAT**=\( n \)
specifies the maximum number of missing patterns to display in the output, where \( n \) is between 1 and 9,999. The default MAXMISSPAT= value is 10 or the number of missing patterns in the data, whichever is smaller. The number of missing patterns to display cannot exceed this MAXMISSPAT= value. This option is relevant only when there are incomplete observations (with some missing values in the analysis variables) in the input raw data set and when you use \( \text{METHOD=FIML} \) or \( \text{METHOD=LSFIML} \) for estimation.

Because the number of missing patterns could be quite large, PROC CALIS displays a limited number of the most frequent missing patterns in the output. The MAXMISSPAT= and the TMISSPAT= options are used in determining the number of missing patterns to display. The missing patterns are ordered according to the data proportions they account for, from the largest to the smallest. PROC CALIS displays a minimum number of the highest-frequency missing patterns. This minimum number is the smallest among five, the actual number of missing patterns, and the MAXMISSPAT= value. Then, PROC CALIS displays the subsequent high-frequency missing patterns if the data proportion accounted for by each of these patterns is at least as large as the proportion threshold set by the TMISSPAT= value (default at 0.05) until the total number of missing patterns displayed reaches the maximum set by the MAXMISSPAT= option.

**MAXOUTLIER**=\( n \)

**MAXOUT**=\( n \)
specifies the maximum number of outliers to display in the output, where \( n \) is between 1 and 9,999. The default MAXOUTLIER= value is 30. The actual numbers of outliers displayed in the output could be smaller than the maximum. In general, PROC CALIS finds the number outliers \( m \) and then adds the next 5 most outlying observations in the output. The actual number of outliers shown in the output is either \( m+5 \), the MAXOUTLIER= value, or the number of observations in the data set, whichever is smaller.
MEANPATTERN=name
MEANPAT=name

specifies one of the built-in mean structures for the data. The purpose of this option is to fit some commonly-used direct mean structures efficiently without the explicit use of the MSTRUCT model specifications. With this option, the mean structures are defined internally in PROC CALIS. The following names for the built-in mean structures are supported:

EQMEANVEC specifies the equality of mean vectors between multiple groups. That is, this option tests the null hypothesis that

\[ H_0 : \mu_1 = \mu_2 = \ldots = \mu_k = \mu \]

where \( \mu \) is a common mean vector for the \( k \mu_j \)'s \( (j = 1, \ldots, k) \). For example, if there are four variables in the analysis, the common \( \mu \) is defined as

\[
\begin{pmatrix}
m_1 \\
m_2 \\
m_3 \\
m_4 \\
\end{pmatrix}
\]

PROC CALIS denotes \( m_1, m_2, m_3 \), and \( m_4 \) by \_mean\_1, \_mean\_2, \_mean\_3, and \_mean\_4, respectively.

If you use the COVPATTERN=EQCOVMAT and MEANPATTERN=EQMEANVEC together in a maximum likelihood (METHOD=ML) analysis, you are testing a null hypothesis of the same multivariate normal distribution for the groups.

If you use the MEANPATTERN=EQMEANVEC option for a single-group analysis, the parameters for the single group are still created accordingly. However, the mean model for the single group contains only unconstrained parameters that would result in saturated mean structures for the model.

SATURATED specifies a saturated mean structure model. This is the default mean structure pattern when the covariance structures are specified by the COVPATTERN= pattern and the mean structure analysis is invoked by MEANSTR option. For example, if there are three variables in the analysis, \( \mu \) is defined as

\[
\begin{pmatrix}
m_1 \\
m_2 \\
m_3 \\
\end{pmatrix}
\]

PROC CALIS denotes \( m_1, m_2, \) and \( m_3 \) by \_mean\_1, \_mean\_2, and \_mean\_3, respectively.

UNIFORM specifies a mean vector with a uniform mean parameter. For example, if there are three variables in the analysis, the mean pattern is:

\[
\begin{pmatrix}
m \\
m \\
m \\
\end{pmatrix}
\]

PROC CALIS denotes the common mean parameter by \_meanparm.
specifies a zero vector for the mean structures. For example, if there are four variables in the analysis, the mean pattern generated by PROC CALIS is:

\[
\begin{pmatrix}
0 \\
0 \\
0 \\
0
\end{pmatrix}
\]

When you specify the mean structure model by means of the MEANPATTERN= option, you can define the set of variables in the analysis by the VAR statement (either within the scope of the PROC CALIS statement or the GROUP statements). If the VAR statement is not used, PROC CALIS uses all numerical variables in the data sets.

Except for the EQMEANVEC pattern, all other built-in mean patterns are primarily designed for single-group analysis. However, you can still use these mean pattern options for multiple-group situations. For example, consider the following three-group analysis:

```plaintext
proc calis meanpattern=uniform;
  group 1 / data=set1;
  group 2 / data=set2;
  group 3 / data=set3;
run;
```

In this specification, all three groups are fitted by the uniform mean pattern. However, there would be no constraints across these groups. PROC CALIS generates a distinct mean parameter for each group: _meanparm_mdl1 for Group 1, _meanparm_mdl2 for Group 2, and _meanparm_mdl3 for Group 3. Similarly, the _mdlxx suffix, where xx represents the model number, is applied to the parameters defined by the SATURATED mean pattern in multiple-group situations.

You can also apply the MEANPATTERN= option partially to the groups in the analysis. For example, the following statements apply the ZERO mean pattern to Group 1 and Group 2 only:

```plaintext
proc calis meanpattern=zero;
  group 1 / data=set1;
  group 2 / data=set2;
  group 3 / data=set3;
  model 3 / group=3;
    path x1 ===> y3;
    means x1 = mean_x1;
run;
```

Group 3 is fitted by Model 3, which is specified explicitly by a PATH model with a distinct mean parameter mean_x1.
If the EQMEANVEC pattern is specified instead, as shown in the following statements, the equality of mean vectors still holds for Groups 1 and 2:

```plaintext
proc calis meanpattern=eqmeanvec;
  group 1 / data=set1;
  group 2 / data=set2;
  group 3 / data=set3;
  model 3 / group=3;
    path x1 ===> y3;
    means x1 = mean_x1;
run;
```

However, Group 3 has its own mean structures as specified in Model 3.

Use the COVPATTERN= option if you also want to analyze some built-in covariance structures along with the mean structures. If you use the MEANPATTERN= option but do not specify the COVPATTERN= option, a saturated covariance structure model (that is, COVPATTERN=SATURATED) is assumed by default.

MEANSTR invokes the analysis of mean structures. By default, no mean structures are analyzed. You can specify the MEANSTR option in both the PROC CALIS and the MODEL statements. When this option is specified in the PROC CALIS statement, it propagates to all models. When this option is specified in the MODEL statement, it applies only to the local model. Except for the COSAN model, the MEANSTR option adds default mean parameters to the model. For the COSAN model, the MEANSTR option adds null mean vectors to the model. Instead of using the MEANSTR option to analyze the mean structures, you can specify the mean and the intercept parameters explicitly in the model by some model specification statements. That is, you can specify the intercepts in the LINEQS statement, the intercepts and means in the PATH or the MEAN statement, the _MEAN_ matrix in the MATRIX statement, or the mean structure formula in the COSAN statement. The explicit mean structure parameter specifications are useful when you need to constrain the mean parameters or to create your own references of the parameters.

METHOD=name
MET=name
M=name

specifies the method of parameter estimation. The default is METHOD=ML. Valid values for name are as follows:

FIML performs full information maximum-likelihood (FIML) or direct maximum likelihood parameter estimation for data with missing values. This method assumes raw input data sets. When there are no missing values in the analysis, the FIML method yields the same estimates as those by using the regular maximum likelihood (METHOD=ML) method with VARDEF=N. Because the FIML method recomputes the mean estimates iteratively during estimation, it must intrinsically analyze the mean structures of models. If you do not specify the MEANSTR option or any mean parameters for your models (which is not required for using the FIML method), PROC CALIS assumes saturated mean structures for models. However, when
computing fit statistics, these saturated mean structures would be ignored as if they were never modeled. If you do specify the MEANSTR option or any mean parameters for your models, these mean structures would be taken into account when computing fit statistics.

**ML | M | MAX** performs normal-theory maximum-likelihood parameter estimation. The ML method requires a nonsingular covariance or correlation matrix.

**GLS | G** performs generalized least squares parameter estimation. If no INWGT= data set is specified, the GLS method uses the inverse sample covariance or correlation matrix as the weight matrix $W$. Therefore, METHOD=GLS requires a nonsingular covariance or correlation matrix.

**WLS | W | ADF** performs weighted least squares parameter estimation. If no INWGT= data set is specified, the WLS method uses the inverse matrix of estimated asymptotic covariances of the sample covariance or correlation matrix as the weight matrix $W$. In this case, the WLS estimation method is equivalent to Browne’s asymptotically distribution-free estimation (Browne 1982, 1984). The WLS method requires a nonsingular weight matrix.

**DWLS | D** performs diagonally weighted least squares parameter estimation. If no INWGT= data set is specified, the DWLS method uses the inverse diagonal matrix of asymptotic variances of the input sample covariance or correlation matrix as the weight matrix $W$. The DWLS method requires a nonsingular diagonal weight matrix.

**ULS | LS | U** performs unweighted least squares parameter estimation.

**LSFIML** performs unweighted least squares followed by full information maximum-likelihood parameter estimation.

**LSML | LSM | LSMAX** performs unweighted least squares followed by normal-theory maximum-likelihood parameter estimation.

**LSGLS | LSG** performs unweighted least squares followed by generalized least squares parameter estimation.

**LSWLS | LSW | LSADF** performs unweighted least squares followed by weighted least squares parameter estimation.

**LSDWLS | LSD** performs unweighted least squares followed by diagonally weighted least squares parameter estimation.

**NONE | NO** uses no estimation method. This option is suitable for checking the validity of the input information and for displaying the model matrices and initial values.

**MODIFICATION**

**MOD** computes and displays Lagrange multiplier (LM) test indices for constant parameter constraints, equality parameter constraints, and active boundary constraints, as well as univariate and multivariate Wald test indices. The modification indices are not computed in the case of unweighted or diagonally weighted least squares estimation.

The Lagrange multiplier test (Bentler 1986; Lee 1985; Buse 1982) provides an estimate of the $\chi^2$ reduction that results from dropping the constraint. For constant parameter constraints and active
boundary constraints, the approximate change of the parameter value is displayed also. You can use this value to obtain an initial value if the parameter is allowed to vary in a modified model. See the section “Modification Indices” on page 1501 for more information.

Relying solely on the LM tests to modify your model can lead to unreliable models that capitalize purely on sampling errors. See MacCallum, Roznowski, and Necowitz (1992) for the use of LM tests.

**MSINGULAR=r**

**MSING=r**

specifies a relative singularity criterion \( r \) \((r > 0)\) for the inversion of the information matrix, which is needed to compute the covariance matrix. If you do not specify the **SINGULAR=** option, the default value for \( r \) or MSING= is 1E–12; otherwise, the default value is 1E–4 \( \times SING \), where **SING** is the specified **SINGULAR=** value.

When inverting the information matrix, the following singularity criterion is used for the diagonal pivot \( d_{j,j} \) of the matrix:

\[
|d_{j,j}| \leq \max(ASING, VSING \times |H_{j,j}|, MSING \times \max(|H_{1,1}|, \ldots, |H_{n,n}|))
\]

where **ASING** and **VSING** are the specified values of the **ASINGULAR=** and **VSINGULAR=** options, respectively, and \( H_{j,j} \) is the \( j \)th diagonal element of the information matrix. Note that in many cases a normalized matrix \( D^{-1}HD^{-1} \) is decomposed (where \( D^2 = \text{diag}(H) \)), and the singularity criteria are modified correspondingly.

**NOADJDF**

turns off the automatic adjustment of degrees of freedom when there are active constraints in the analysis. When the adjustment is in effect, most fit statistics and the associated probability levels will be affected. This option should be used when you believe that the active constraints observed in the current sample will have little chance to occur in repeated sampling. See the section “Adjustment of Degrees of Freedom” on page 1479 for more discussion on the issue.

**NOBS=nobs**

specifies the number of observations. If the **DATA=** input data set is a raw data set, **nobs** is defined by default to be the number of observations in the raw data set. The **NOBS=** and **EDF=** options override this default definition. You can use the **RDF=** option to modify the **nobs** specification. If the **DATA=** input data set contains a covariance, correlation, or scalar product matrix, you can specify the number of observations either by using the **NOBS=**, **EDF=**, and **RDF=** options in the **PROC CALIS** statement or by including a **_TYPE_**='N’ observation in the **DATA=** input data set.

**NOINDEXTYPE**

disables the display of index types in the fit summary table.

**NOMEANSTR**

deactivates the inherited **MEANSTR** option for the analysis of mean structures. You can specify the **NOMEANSTR** option in both the **PROC CALIS** and the **MODEL** statements. When this option is specified in the **PROC CALIS** statement, it does not have any apparent effect because by default the mean structures are not analyzed. When this option is specified in the **MODEL** statement, it deactivates the inherited **MEANSTR** option from the **PROC CALIS** statement. In other words, this option is mainly used for resetting the default behavior in the local model that is specified within the scope of a particular **MODEL** statement. If you specify both the **MEANSTR** and **NOMEANSTR** options in the same statement, the **NOMEANSTR** option is ignored.
\textbf{CAUTION:} This option does not remove the mean structure specifications from the model. It only deactivates the MEANSTR option inherited from the PROC CALIS statement. The mean structures of the model are analyzed as long as there are mean structure specifications in the model (for example, when you specify the means or intercepts in any of the \textit{main} or \textit{subsidiary} model specification statements).

\textbf{NOMISSPAT}

suppresses the display of the analytic results of the missing patterns. This option is relevant only when there are incomplete observations (with some missing values in the analysis variables) in the input raw data set and when you use \texttt{METHOD=FIML} or \texttt{METHOD=LSFIML} for estimation.

\textbf{NOMOD}

suppresses the computation of modification indices. The NOMOD option is useful in connection with the PALL option because it saves computing time.

\textbf{NOORDERSPEC}

prints the model results in the order they appear in the input specifications. This is the default printing behavior. In contrast, the ORDERSPEC option arranges the model results by the types of parameters. You can specify the NOORDERSPEC option in both the PROC CALIS and the MODEL statements. When this option is specified in the PROC CALIS statement, it does not have any apparent effect because by default the model results display in the same order as that in the input specifications. When this option is specified in the MODEL statement, it deactivates the inherited ORDERSPEC option from the PROC CALIS statement. In other words, this option is mainly used for resetting the default behavior in the local model that is specified within the scope of a particular MODEL statement. If you specify both the ORDERSPEC and NOORDERSPEC options in the same statement, the NOORDERSPEC option is ignored.

\textbf{NOPARMNAME}

suppresses the printing of parameter names in the model results. The default is to print the parameter names. You can specify the NOPARMNAME option in both the PROC CALIS and the MODEL statements. When this option is specified in the PROC CALIS statement, it propagates to all models. When this option is specified in the MODEL statement, it applies only to the local model.

\textbf{NOPRINT}

\textbf{NOP}

suppresses the displayed output. Note that this option temporarily disables the Output Delivery System (ODS). See Chapter 20, “Using the Output Delivery System,” for more information.

\textbf{NOSTAND}

suppresses the printing of standardized results. The default is to print the standardized results.

\textbf{NOSTDERR}

\textbf{NOSE}

suppresses the printing of the standard error estimates. Standard errors are not computed for unweighted least squares (ULS) or diagonally weighted least squares (DWLS) estimation. In general, standard errors are computed even if the STDERR display option is not used (for file output). You can specify the NOSTDERR option in both the PROC CALIS and the MODEL statements. When this option is specified in the PROC CALIS statement, it propagates to all models. When this option is specified in the MODEL statement, it applies only to the local model.
OMETHOD=name
OM=name
TECHNIQUE=name
TECH=name

specifies the optimization method or technique. Because there is no single nonlinear optimization algorithm available that is clearly superior (in terms of stability, speed, and memory) for all applications, different types of optimization methods or techniques are provided in the CALIS procedure. The optimization method or technique is specified by using one of the following names in the OMETHOD= option:

CONGRA | CG chooses one of four different conjugate-gradient optimization algorithms, which can be more precisely defined with the UPDATE= option and modified with the LINESEARCH= option. The conjugate-gradient techniques need only $O(t)$ memory compared to the $O(t^2)$ memory for the other three techniques, where $t$ is the number of parameters. On the other hand, the conjugate-gradient techniques are significantly slower than other optimization techniques and should be used only when memory is insufficient for more efficient techniques. When you choose this option, UPDATE=PB by default. This is the default optimization technique if there are more than 999 parameters to estimate.

DBLDOG | DD performs a version of double dogleg optimization, which uses the gradient to update an approximation of the Cholesky factor of the Hessian. This technique is, in many aspects, very similar to the dual quasi-Newton method, but it does not use line search. The implementation is based on Dennis and Mei (1979) and (Gay 1983).

LEVMAR | LM | MARQUARDT performs a highly stable (but for large problems, memory- and time-consuming) Levenberg-Marquardt optimization technique, a slightly improved variant of the (Moré 1978) implementation. This is the default optimization technique for estimation methods other than the FIML if there are fewer than 500 parameters to estimate.

NEWRAP | NRA performs a usually stable (but for large problems, memory- and time-consuming) Newton-Raphson optimization technique. The algorithm combines a line-search algorithm with ridging, and it can be modified with the LINESEARCH= option.

NRRIDG | NRR | NR | NEWTON performs a usually stable (but for large problems, memory- and time-consuming) Newton-Raphson optimization technique. This algorithm does not perform a line search. Since OMETHOD=NRRIDG uses an orthogonal decomposition of the approximate Hessian, each iteration of OMETHOD=NRRIDG can be slower than that of OMETHOD=NEWRAP, which works with Cholesky decomposition. However, usually OMETHOD=NRRIDG needs fewer iterations than OMETHOD=NEWRAP. The NRRIDG technique is the default optimization for the FIML estimation if there are fewer than 500 parameters to estimate.

QUANEW | QN chooses one of four different quasi-Newton optimization algorithms that can be more precisely defined with the UPDATE= option and modified with the LINESEARCH= option. If boundary constraints are used, these techniques sometimes converge slowly. When you choose this option, UPDATE=DBFGS
by default. If nonlinear constraints are specified in the NLINCON statement, a
modification of Powell’s VMCWD algorithm (Powell 1982a, b) is used, which
is a sequential quadratic programming (SQP) method. This algorithm can be
modified by specifying VERSION=1, which replaces the update of the Lagrange
multiplier estimate vector \( \mu \) to the original update of Powell (1978b, a) that
is used in the VF02AD algorithm. This can be helpful for applications with
linearly dependent active constraints. The QUANEW technique is the default
optimization technique if there are nonlinear constraints specified or if there are
more than 499 and fewer than 1,000 parameters to estimate. The QUANEW
algorithm uses only first-order derivatives of the objective function and, if
available, of the nonlinear constraint functions.

**TRUREG | TR**  performs a usually very stable (but for large problems, memory- and time-
consuming) trust-region optimization technique. The algorithm is implemented
similar to Gay (1983) and Moré and Sorensen (1983).

**NONE | NO**  does not perform any optimization. This option is similar to
METHOD=NONE, but OMETHOD=NONE also computes and displays
residuals and goodness-of-fit statistics. If you specify METHOD=ML,
METHOD=LSML, METHOD=GLS, METHOD=LSGLS, METHOD=WLS,
or METHOD=LSWLS, this option enables computing and displaying (if the
display options are specified) of the standard error estimates and modification
indices corresponding to the input parameter estimates.

For fewer than 500 parameters \( (t < 500) \), OMETHOD=NRRIDG (Newton-Raphson Ridge) is the
default optimization technique for the FIML estimation, and OMETHOD=LEVMAR (Levenberg-
Marquardt) is the default optimization technique for all other estimation methods. For \( 500 \leq
\leq 1,000 \), OMETHOD=QUANEW (quasi-Newton) is the default method, and for \( t \geq 1,000 \),
OMETHOD=CONGRA (conjugate gradient) is the default method. Each optimization method or
technique can be modified in various ways. See the section “Use of Optimization Techniques” on
page 1507 for more details.

**ORDERALL**  prints the model and group results in the order of the model or group numbers, starting from the
smallest number. It also arranges some model results by the parameter types. In effect, this option turns
on the ORDERGROUPS, ORDERMODELS, and ORDERSPEC options. The ORDERALL is not a
default option. By default, the printing of the results follow the order of the input specifications.

**ORDERGROUPS**

**ORDERG**  prints the group results in the order of the group numbers, starting from the smallest number. The default
behavior, however, is to print the group results in the order they appear in the input specifications.

**ORDERMODELS**

**ORDERMO**  prints the model results in the order of the model numbers, starting from the smallest number. The de-
fault behavior, however, is to print the model results in the order they appear in the input specifications.
ORDERSPEC
arranges some model results by the types of parameters. The default behavior, however, is to print the
results in the order they appear in the input specifications. You can specify the ORDERSPEC option
in both the PROC CALIS and the MODEL statements. When this option is specified in the PROC
CALIS statement, it propagates to all models. When this option is specified in the MODEL statement,
it applies only to the local model.

OUTEST=SAS-data-set
creates an output data set that contains the parameter estimates, their gradient, Hessian matrix, and
boundary and linear constraints. For METHOD=ML, METHOD=GLS, and METHOD=WLS, the
OUTEST= data set also contains the information matrix, the approximate covariance matrix of the
parameter estimates ((generalized) inverse of information matrix), and approximate standard errors.
If linear or nonlinear equality or active inequality constraints are present, the Lagrange multiplier
estimates of the active constraints, the projected Hessian, and the Hessian of the Lagrange function are
written to the data set.

See the section “OUTEST= Data Set” on page 1371 for a description of the OUTEST= data set. If you
want to create a SAS data set in a permanent library, you must specify a two-level name. For more
information about permanent libraries and SAS data sets, see SAS Language Reference: Concepts.

OUTFIT=SAS-data-set
creates an output data set that contains the values of the fit indices. See the section “OUTFIT= Data
Set” on page 1385 for details.

OUTMODEL=SAS-data-set
OUTRAM=SAS-data-set
creates an output data set that contains the model information for the analysis, the parameter estimates,
and their standard errors. An OUTMODEL= data set can be used as an input INMODEL= data set in a
subsequent analysis by PROC CALIS. If you want to create a SAS data set in a permanent library, you
must specify a two-level name. For more information about permanent libraries and SAS data sets, see

OUTSTAT=SAS-data-set
creates an output data set that contains the BY group variables, the analyzed covariance or correlation
matrices, and the predicted and residual covariance or correlation matrices of the analysis. You can
specify the correlation or covariance matrix in an OUTSTAT= data set as an input DATA= data set in a
subsequent analysis by PROC CALIS. See the section “OUTSTAT= Data Set” on page 1381 for
a description of the OUTSTAT= data set. If the model contains latent variables, this data set also
contains the predicted covariances between latent and manifest variables and the latent variable score
regression coefficients (see the PLATCOV option on page 1231). If the FACTOR statement is used, the
OUTSTAT= data set also contains the rotated and unrotated factor loadings, the unique variances, the
matrix of factor correlations, the transformation matrix of the rotation, and the matrix of standardized
factor loadings.

You can use the latent variable score regression coefficients with PROC SCORE to compute factor
scores.

If you want to create a SAS data set in a permanent library, you must specify a two-level name.
For more information about permanent libraries and SAS data sets, see SAS Language Reference:
Concepts.
OUTWGT=SAS-data-set
OUTWEIGHT=SAS-data-set

creates an output data set that contains the elements of the weight matrix \( W \) or its inverse \( W^{-1} \) used in the estimation process. The inverse of the weight matrix is output only when you specify an INWGT= data set with the INWGT= and INWGTINV options (or the INWGT(INV)= option alone) in the same analysis. As a result, the entries in the INWGT= and OUTWGT= data sets are consistent. In other situations where the weight matrix is computed by the procedure or obtained from the OUTWGT= data set without the INWGTINV option, the weight matrix is output in the OUTWGT= data set. Furthermore, if the weight matrix is computed by the procedure, the OUTWGT= data set contains the elements of the weight matrix on which the WRIDGE= and the WPENALTY= options are applied.

You cannot create an OUTWGT= data set with an unweighted least squares or maximum likelihood estimation. The weight matrix is defined only in the GLS, WLS (ADF), or DWLS fit function. An OUTWGT= data set can be used as an input INWGT= data set in a subsequent analysis by PROC CALIS. See the section “OUTWGT= Data Set” on page 1385 for the description of the OUTWGT= data set. If you want to create a SAS data set in a permanent library, you must specify a two-level name. For more information about permanent libraries and SAS data sets, see SAS Language Reference: Concepts.

PALL
ALL

displays all optional output except the output generated by the PCOVES and PDETERM options.

CAUTION: The PALL option includes the very expensive computation of the modification indices. If you do not really need modification indices, you can save computing time by specifying the NOMOD option in addition to the PALL option.

PARMNAME

prints the parameter names in the model results. This is the default printing behavior. In contrast, the NOPARMNAME option suppresses the printing of the parameter names in the model results. You can specify the PARMNAME option in both the PROC CALIS and the MODEL statements. When this option is specified in the PROC CALIS statement, it does not have any apparent effect because by default model results show the parameter names. When this option is specified in the MODEL statement, it deactivates the inherited NOPARMNAME option from the PROC CALIS statement. In other words, this option is mainly used for resetting the default behavior in the local model that is specified within the scope of a particular MODEL statement. If you specify both the PARMNAME and NOPARMNAME options in the same statement, the PARMNAME option is ignored.

PCORR
CORR

displays the covariance or correlation matrix that is analyzed and the predicted model covariance or correlation matrix.

PCOVES
PCE

displays the following:

- the information matrix
• the approximate covariance matrix of the parameter estimates (generalized inverse of the information matrix)
• the approximate correlation matrix of the parameter estimates

The covariance matrix of the parameter estimates is not computed for estimation methods ULS and DWLS. This displayed output is not included in the output generated by the PALL option.

PDETERM
PDE

displays three coefficients of determination: the determination of all equations (DETAE), the determination of the structural equations (DETSE), and the determination of the manifest variable equations (DETMV). These determination coefficients are intended to be global means of the squared multiple correlations for different subsets of model equations and variables. The coefficients are displayed only when you specify a FACTOR, LINEQS, LISMOD, PATH, or RAM model, but they are displayed for all five estimation methods: ULS, GLS, ML, WLS, and DWLS.

You can use the STRUCTEQ statement to define which equations are structural equations. If you do not use the STRUCTEQ statement, PROC CALIS uses its own default definition to identify structural equations.

The term “structural equation” is not defined in a unique way. The LISREL program defines the structural equations by the user-defined BETA matrix. In PROC CALIS, the default definition of a structural equation is an equation that has a dependent left-side variable that appears at least once on the right side of another equation, or an equation that has at least one right-side variable that appears at the left side of another equation. Therefore, PROC CALIS sometimes identifies more equations as structural equations than the LISREL program does.

PESTIM
PES

displays the parameter estimates. In some cases, this includes displaying the standard errors and t values.

PIN
PIN

displays the model specification with initial estimates and the vector of initial values.

PLATCOV
PLATMOM
PLC

displays the following:

• the estimates of the covariances among the latent variables
• the estimates of the covariances between latent and manifest variables
• the estimates of the latent variable means for mean structure analysis
• the latent variable score regression coefficients

The estimated covariances between latent and manifest variables and the latent variable score regression coefficients are written to the OUTSTAT= data set. You can use the score coefficients with PROC SCORE to compute factor scores.
Chapter 29: The CALIS Procedure

**PLOTS< =plot-request >**

**PLOT< =((plot-request < ...plot-request >))>**

specifies the ODS Graphics plots. When you specify only one plot-request, you can omit the parentheses around the plot-request. For example:

PLOTS=ALL
PLOTS=RESIDUALS
PLOTS=(PP RESBYPRED QQ)

ODS Graphics must be enabled before plots can be requested. For example:

```
ods graphics on;
proc calis plots;
    path y <= x,
        y <= z;
run;
ods graphics off;
```

For more information about enabling and disabling ODS Graphics, see the section “Enabling and Disabling ODS Graphics” on page 606 in Chapter 21, “Statistical Graphics Using ODS.”

You can specify the following plot-requests:

**ALL**

- displays all plots.

**CASERESID | CASERESIDUAL | CASERESIDUALS**

- displays all the case-level ODS Graphics plots enabled by the following plot-requests: CRESHIST, PP, QQ, RESBYLEV, and RESBYPRED. This option requires raw data input.

**CRESHIST | CRESIDUALHISTOGRAM**

- produces the ODS Graphics plot CaseResidualHistogram, which displays the distribution of the case-level (observation-level) residuals in the form of a histogram, where residuals are measured in terms of M-distances. This option requires raw data input.

**NONE**

- suppresses ODS Graphics plots.

**PATHDIAGRAM**

- produces the ODS Graphics plot PathDiagram, which display the path diagram for the unstandardized solution. For options that control and customize path diagrams, see the PATHDIAGRAM statement.

**PP | PP PLOT**

- produces the ODS Graphics plot ResPercentileByExpPercentile, which plots the observed percentiles of the residual M-distances against the theoretical percentiles. This plot is useful for showing departures from the theoretical distribution in terms of percentiles, and it is especially sensitive to departures in the middle region of the distribution. This option requires raw data input.

**QQ | QQ PLOT**

- produces the ODS Graphics plot ResidualByQuantile, which plots the residual M-distances (observed quantiles) against the theoretical quantiles. This plot is useful for showing departures from the theoretical distribution in terms of quantiles, and it is especially sensitive to departures at the upper tail of the distribution. This option requires raw data input.

**RESBYLEV | RESIDUAL BY LEVERAGE**

- produces the ODS Graphics plot ResidualByLeverage, which plots the residual M-distances against the leverage M-distances. This plot
is useful for showing outliers and leverage observations graphically. See the
\texttt{ALPHAOUT=} and \texttt{ALPHALEV=} options for detection criteria of outliers and
leverage observations. This option requires raw data input.

\texttt{RESBYPRED | RESONFIT | RESIDUALBYPREDICTED | RESIDUALONFIT}<\texttt{(VAR= }\texttt{var-list} )>
produces the ODS Graphics plot ResidualByPredicted, which plots the residuals
against the predicted values of the dependent observed variables in the model. You
can restrict the set of dependent variables to display by specifying \texttt{var-list} in the
\texttt{VAR=} option. If \texttt{var-list} is not specified in the \texttt{VAR=} option, plots for all dependent
observed variables are displayed. The residual on fit plots are useful for detecting
nonlinear relationships in the model. If the relationships are linear and the residual
variance is homoscedastic, the residuals should not show systematic pattern with
the predicted values. This option requires raw data input.

\texttt{RESIDUAL | RESIDUALS} produces the ODS Graphics plot for the histogram of residuals in covari-
ances and means rather than the case-level residuals. With this ODS Graphics plot,
the nongraphical (legacy) output for the bar chart of residual tallies is redundant and
therefore is suppressed. To display this bar chart together with the ODS Graphics
for residual histogram, you must use the \texttt{RESIDUAL(TALLY)} option in the PROC
CALIS statement. This option does not require raw data input.

\texttt{PRIMAT} \\
\texttt{PMAT}

\texttt{PRINT} \\
\texttt{PRI} \\
\texttt{PRINT} \\
\texttt{PRI}

\texttt{PSHORT} \\
\texttt{SHORT} \\
\texttt{PSH}

\texttt{PSUMMARY | SUMMARY | PSUM} \\
\texttt{SUMMARY} \\
\texttt{PSUM}

\texttt{PWEIGHT} \\
\texttt{PW}

\texttt{PRIMAT}

\texttt{PMAT}

\texttt{PRINT}

\texttt{PRI}

\texttt{PSHORT}

\texttt{SHORT}

\texttt{PSH}

\texttt{PSUM}

\texttt{PWEIGHT}

\texttt{PW}
RADIUS=r

is an alias for the INSTEP= option for Levenberg-Marquardt minimization.

RANDOM=i

specifies a positive integer as a seed value for the pseudo-random number generator to generate initial values for the parameter estimates for which no other initial value assignments in the model definitions are made. Except for the parameters in the diagonal locations of the central matrices in the model, the initial values are set to random numbers in the range $0 \leq r \leq 1$. The values for parameters in the diagonals of the central matrices are random numbers multiplied by 10 or 100. See the section “Initial Estimates” on page 1506 for more information.

RDF=n

DFR=n

makes the effective number of observations the actual number of observations minus the RDF= value. The degree of freedom for the intercept should not be included in the RDF= option. If you use PROC CALIS to compute a regression model, you can specify RDF= number-of-regressor-variables to get approximate standard errors equal to those computed by PROC REG.

READADDPARM

READADD

inputs the generated default parameters (for example, observations with _TYPE_=ADDPCOV, ADDMEAN, or ADDPVAR) in the INMODEL= data set as if they were part of the original model specification. Typically, these default parameters in the INMODEL= data set were generated automatically by PROC CALIS in a previous analysis and stored in an OUTMODEL= data set, which is then used as the INMODEL= data set in a new run of PROC CALIS. By default, PROC CALIS does not input the observations for default parameters in the INMODEL= data set. In most applications, you do not need to specify this option because PROC CALIS is able to generate a new set of default parameters that are appropriate to the new situation after it reads in the INMODEL= data set. Undistinguished uses of the READADDPARM option might lead to unintended constraints on the default parameters.

RESIDUAL <(TALLY | TALLIES)> <= NORM | VARSTAND | ASYSTAND>

RES <(TALLY | TALLIES)> <= NORM | VARSTAND | ASYSTAND>

displays the raw and normalized residual covariance matrix, the rank order of the largest residuals, and a bar chart of the residual tallies. If mean structures are modeled, mean residuals are also displayed and ranked.

For raw data input, this option also displays tables for case-level (observation-level) residual analysis, including outlier and leverage detections and departures of residuals from the theoretical residual distributions. To set the criterion for detecting outliers, use the ALPHAOOUT= option. To set the criterion for leverage observations, use the ALPHALEV= option. Case-level residual analysis is not available when you specify METHOD=FIML.

For the covariance and mean residuals, three types of normalized or standardized residual matrices can be chosen with the RESIDUAL= specification.

NORM normalized residuals

VARSTAND variance standardized residuals

ASYSTAND asymptotically standardized residuals
When ODS Graphics plots of covariance and mean residuals are also requested, the bar charts of residual tallies are suppressed. They are replaced with high quality graphical histograms showing residual distributions. If you still want to display the bar charts in this situation, use the RESIDUAL(TALLY) or RESIDUAL(TALLY)= option.

The RESIDUAL option is also enabled by the PRINT option. See the section “Assessment of Fit” on page 1481 for more details about the definitions of residuals.

RIDGE< =r > defines a ridge factor \( r \) for the diagonal of the covariance or correlation matrix \( S \) that is analyzed. The matrix \( S \) is transformed to:

\[
S \rightarrow \tilde{S} = S + r(\text{diag}(S))
\]

If you do not specify \( r \) in the RIDGE option, PROC CALIS tries to ridge the covariance or correlation matrix \( S \) so that the smallest eigenvalue is about \( 10^{-3} \). Because the weight matrix in the GLS method is the same as the observed covariance or correlation matrix, the RIDGE= option also applies to the weight matrix for the GLS estimation, unless you input the weight matrix by the INWGT= option.

\text{CAUTION:} The covariance or correlation matrix in the \text{OUTSTAT=} output data set does not contain the ridged diagonal.

ROBITER= \( i \) specifies the maximum number \( i \) of iterations for the iteratively reweighted least squares (IRLS) method to compute the robust mean and covariance matrices with the two-stage robust estimation. This option is relevant only with the use of the ROBUST= option and with raw data input. The default value is 5,000.

You can also specify this option in the GROUP statement so that different groups can use different ROBITER= values. Notice that the ROBITER= option does not specify the maximum number of iterations for the IRLS algorithm used in the direct robust estimation or in the second stage of the two-stage robust estimation. You can specify the MAXITER= option for this purpose.

ROBPHI= \( r \) sets the tuning parameter \( r \) (\( 0 < r < 1 \)) for the robust estimation method that you specify using the ROBUST= option. The ROBPHI= value controls the criterion for downweighting observations. This value indicates approximately the proportion of observations that would receive weights less than 1 (that is, would be downweighted) according to certain theoretical distributions. The larger the ROBPHI= value, the more observations are downweighted (that is, with weights less than 1). The default value is 0.05.

You can also specify this option in the GROUP statement so that different groups can use different ROBPHI= values for the tuning parameters.

ROBUST < <=name > invokes the robust estimation method that downweights the outliers in estimation. You can use the ROBUST option only in conjunction with the ML method (METHOD=ML). More accurately, the robust estimation is done by using the iteratively reweighted least squares (IRLS) method under
Chapter 29: The CALIS Procedure

the normal distribution assumption. The model fit of robust estimation is evaluated with the ML discrepancy function.

You must provide raw data input for the robust estimation method to work. With the robust method, the Huber weights are applied to the observations so that outliers are downweighted during estimation. See the section “Robust Estimation” on page 1470 for details.

You can request the three different types of robust methods by using one of the following names:

RESIDUAL | DIRECT | RESID | RES <(E)> specifies a direct robust method that downweights observations with large residuals during the iterative estimation of the model. This method treats the disturbances (the error terms of endogenous latent factors) as errors or residuals (hence the keyword E) in the associated factor model when computing residual M-distances and factor scores during the robust estimation. The (E) specification is irrelevant if there are no endogenous latent factors in the model. This is the default robust method.

RESIDUAL | DIRECT | RESID | RES (F) specifies a direct robust method that downweights observations with large estimated residuals during the iterative estimation of the model. Unlike the (E) method, this method treats the disturbances (the error terms of endogenous latent factors) as factors (hence the keyword F) in the associated factor model when computing residual M-distances and factor scores during the robust estimation. The (F) specification is irrelevant if there are no endogenous latent factors in the model.

SAT | TWOSTAGE | UNSTRUCT | UNS specifies a two-stage robust method that downweights the observations with large M-distances in all observed variable dimensions when computing the covariance matrix and mean vector from the input raw data. As a result, this option produces a robust covariance matrix and a mean vector for a subsequent model estimation where no reweighting would be applied at the observational level. Hence, this is a two-stage method that applies weights only in the first stage for computing the robust covariance and mean matrices. This is in contrast with the RES(E) or RES(F) option, where weighting and reweighting of observations are applied directly during model estimation.

For details about these robust methods, see the section “Robust Estimation” on page 1470.

To control the proportion of the observations that are downweighted during the robust estimation, you can specify the value of the tuning parameter \( \varphi \), which is between 0 and 1, by using the ROBPHI= option. Approximately, \( \varphi \times 100\% \) of observations would receive weights less than 1 according to certain theoretical distributions. By default, the value of the tuning parameter \( \varphi \) is set to 0.05 for all robust methods in PROC CALIS.

By default, the robust method uses a maximum of 5,000 iterations to obtain parameter convergence through the IRLS algorithm. You can override this default maximum number of iterations by specifying the ROBITER= option. The default relative parameter convergence criterion for the robust method is \( 1 \times 10^{-8} \). See the XCONV= option for the mathematical definition of this criterion and for information about overriding the default convergence criterion.

Because all robust methods reweight the observations iteratively, the observed variable means are always implicitly updated with the robust weights. Therefore, in a sense all robust methods intrinsically analyze the mean structures of models. If you do not specify the MEANSTR option or any mean
parameters for your models, PROC CALIS assumes appropriate saturated mean structures for the models. However, when you are computing fit statistics, these saturated mean structures are ignored as if they were never modeled. If you do specify the MEANSTR option or any mean parameters for your models, these mean structures are taken into account in computing fit statistics.

In this release, robust estimation with the IRLS method is not supported when you specify the BOUNDS, LINCON, or NLINCON statement. However, you can still set parameter constraints by using the same parameter names or by specifying the PARAMETERS statement and the SAS programming statements. See the section “Setting Constraints on Parameters” on page 1434 for techniques to set up implicit parameter constraints by using the PARAMETERS statement and SAS programming statements.

SALPHA=r
is an alias for the INSTEP= option for line-search algorithms.

SIMPLE
S

displays means, standard deviations, skewness, and univariate kurtosis if available. This information is displayed when you specify the PRINT option. If the KURTOSIS option is specified, the SIMPLE option is set by default.

SINGULAR=r
SING=r

specifies the singularity criterion \( r (0 < r < 1) \) used, for example, for matrix inversion. The default value is the square root of the relative machine precision or, equivalently, the square root of the largest double precision value that, when added to 1, results in 1.

SLMW=r

specifies the probability limit used for computing the stepwise multivariate Wald test. The process stops when the univariate probability is smaller than \( r \). The default value is \( r = 0.05 \).

SPRECISION=r
SP=r

is an alias for the LSPRECISION= option.

START=r

specifies initial estimates for parameters as multiples of the \( r \) value. In all CALIS models, you can supply initial estimates individually as parenthesized values after each parameter name. Unspecified initial estimates are usually computed by various reasonable initial estimation methods in PROC CALIS. If none of the initialization methods is able to compute all the unspecified initial estimates, then the remaining unspecified initial estimates are set to \( r \), \( 10 \sqrt{r} \), or \( 100 \sqrt{r} \). For variance parameters, \( 100 \sqrt{r} \) is used for covariance structure analyses and \( 100 \sqrt{r} \) is used for correlation structure analyses. For other types of parameters, \( r \) is used. The default value is \( r = 0.5 \). If the DEMPHAS= option is used, the initial values of the variance parameters are multiplied by the value specified in the DEMPHAS= option. See the section “Initial Estimates” on page 1506 for more information.

STDERR
SE

displayes approximate standard errors if estimation methods other than unweighted least squares (ULS) or diagonally weighted least squares (DWLS) are used (and the NOSTDERR option is not specified). In contrast, the NOSTDERR option suppresses the printing of the standard error estimates. If you
specify neither the STDERR nor the NOSTDERR option, the standard errors are computed for the OUTMODEL= data set. This information is displayed by default when you specify the PRINT option.

You can specify the STDERR option in both the PROC CALIS and the MODEL statements. When this option is specified in the PROC CALIS statement, it does not have any apparent effect because by default the model results display the standard error estimates (for estimation methods other than ULS and DWLS). When this option is specified in the MODEL statement, it deactivates the inherited NOSTDERR or NOSE option from the PROC CALIS statement. In other words, this option is mainly used for resetting the default behavior in the local model that is specified within the scope of a particular MODEL statement. If you specify both the STDERR and NOSTDERR options in the same statement, the STDERR option is ignored.

**TMISSPAT | THRESHOLDMISSPAT | THRESMISSPAT=\(n\)**

**THRESHOLDMISSPAT=\(n\)**

**THRESMISSPAT=\(n\)**

specifies the proportion threshold for the missing patterns to display in the output, where \(n\) is between 0 and 1. The default TMISSPAT= value is 0.05. This option is relevant only when there are incomplete observations (with some missing values in the analysis variables) in the input raw data set and when you use METHOD=FIML or METHOD=LSFIML for estimation.

Because the number of missing patterns could be quite large, PROC CALIS displays a limited number of the most frequent missing patterns in the output. Together with the MAXMISSPAT= option, this option controls the number of missing patterns to display in the output. See the MAXMISSPAT= option for a detailed description about how the number of missing patterns to display is determined.

**UPDATE=\(name\)**

**UPD=\(name\)**

specifies the update method for the quasi-Newton or conjugate-gradient optimization technique.

For OMETHOD=CONGRA, the following updates can be used:

**PB** performs the automatic restart update method of Powell (1977) and Beale (1972). This is the default.

**FR** performs the Fletcher-Reeves update (Fletcher 1980, p. 63).

**PR** performs the Polak-Ribiere update (Fletcher 1980, p. 66).

**CD** performs a conjugate-descent update of Fletcher (1987).

For OMETHOD=DBLDOG, the following updates (Fletcher 1987) can be used:

**DBFGS** performs the dual Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update of the Cholesky factor of the Hessian matrix. This is the default.

**DDFP** performs the dual Davidon, Fletcher, and Powell (DFP) update of the Cholesky factor of the Hessian matrix.

For OMETHOD=QUANEW, the following updates (Fletcher 1987) can be used:

**BFGS** performs original BFGS update of the inverse Hessian matrix. This is the default for earlier releases.
**PROC CALIS Statement**

DFP  performs the original DFP update of the inverse Hessian matrix.

DBFGS  performs the dual BFGS update of the Cholesky factor of the Hessian matrix. This is the default.

DDFP  performs the dual DFP update of the Cholesky factor of the Hessian matrix.

**VARDEF= DF | N | WDF | WEIGHT | WGT**

specifies the divisor used in the calculation of covariances and standard deviations. The default value is VARDEF=N for the METHOD=FIML, and VARDEF=DF for other estimation methods. The values and associated divisors are displayed in the following table, where \( k \) is the number of partial variables specified in the PARTIAL statement. When a WEIGHT statement is used, \( w_j \) is the value of the WEIGHT variable in the \( j \)th observation, and the summation is performed only over observations with positive weight.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Divisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>Degrees of freedom</td>
<td>( N - k - 1 )</td>
</tr>
<tr>
<td>N</td>
<td>Number of observations</td>
<td>( N )</td>
</tr>
<tr>
<td>WDF</td>
<td>Sum of weights DF</td>
<td>( \sum_{j}^{N} w_j - k - 1 )</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>Sum of weights</td>
<td>( \sum_{j}^{N} w_j )</td>
</tr>
<tr>
<td>WGT</td>
<td>Sum of weights</td>
<td></td>
</tr>
</tbody>
</table>

**VSINGULAR=r**

**VSING=r** specifies a relative singularity criterion \( r \) (\( r > 0 \)) for the inversion of the information matrix, which is needed to compute the covariance matrix. If you do not specify the SINGULAR= option, the default value for \( r \) or VSING= is 1E-8; otherwise, the default value is SING, which is the specified SINGULAR= value.

When inverting the information matrix, the following singularity criterion is used for the diagonal pivot \( d_{j,j} \) of the matrix:

\[
|d_{j,j}| \leq \max(ASING, VSING \times |H_{j,j}|, MSING \times \max(|H_{1,1}|, \ldots, |H_{n,n}|))
\]

where ASING and MSING are the specified values of the ASINGULAR= and MSINGULAR= options, respectively, and \( H_{j,j} \) is the \( j \)th diagonal element of the information matrix. Note that in many cases a normalized matrix \( D^{-1}HD^{-1} \) is decomposed (where \( D^2 = \text{diag}(H) \)), and the singularity criteria are modified correspondingly.

**WPENALTY=r**

**WPEN=r** specifies the penalty weight \( r \geq 0 \) for the WLS and DWLS fit of the diagonal elements of a correlation matrix (constant 1s). The criterion for weighted least squares estimation of a correlation structure is

\[
F_{WLS} = \sum_{i=2}^{n} \sum_{j=1}^{n-k} \sum_{k=2}^{n-1} \sum_{l=1}^{k-1} w_{ij,kl} (s_{ij} - c_{ij}) (s_{kl} - c_{kl}) + r \sum_{i}^{n} (s_{ii} - c_{ii})^2
\]

where \( r \) is the penalty weight specified by the WPENALTY=r option and the \( w_{ij,kl} \) are the elements of the inverse of the reduced \( (n(n-1)/2) \times (n(n-1)/2) \) weight matrix that contains only the nonzero rows and columns of the full weight matrix \( W \). The second term is a penalty term to fit the diagonal
elements of the correlation matrix. The default value is 100. The reciprocal of this value replaces
the asymptotic variance corresponding to the diagonal elements of a correlation matrix in the weight
matrix $W_0$, and it is effective only with the ASYCOV=CORR option, which is the default for correlation
analyses. The often used value $r = 1$ seems to be too small in many cases to fit the diagonal elements
of a correlation matrix properly. The default WPENALTY= value emphasizes the importance of the fit
of the diagonal elements in the correlation matrix. You can decrease or increase the value of $r$ if you
want to decrease or increase the importance of the diagonal elements fit. This option is effective only
with the WLS or DWLS estimation method and the analysis of a correlation matrix.

See the section “Estimation Criteria” on page 1463 for more details.

**CAUTION:** If you input the weight matrix by the INWGT= option, the WPENALTY= option is
ignored.

WRIDGE=$r$

defines a ridge factor $r$ for the diagonal of the weight matrix $W$ used in GLS, WLS, or DWLS
estimation. The weight matrix $W$ is transformed to

$$W \rightarrow \tilde{W} = W + r(\text{diag}(W))$$

The WRIDGE= option is applied on the weight matrix before the following actions occur:

- the WPENALTY= option is applied on it
- the weight matrix is written to the OUTWGT= data set
- the weight matrix is displayed

**CAUTION:** If you input the weight matrix by the INWGT= option, the OUTWGT= data set will
contain the same weight matrix without the ridging requested by the WRIDGE= option. This ensures
that the entries in the INWGT= and OUTWGT= data sets are consistent. The WRIDGE= option is
ignored if you input the inverse of the weight matrix by the INWGT= and INWGTINV options (or the
INWGT(INV)= option alone).

XCONV=$r$

XTOL=$r$

specifies the relative parameter convergence criterion. Termination requires a small relative parameter
($x$) change in subsequent iterations, that is,

$$\max_j \frac{|x_j^{(k)} - x_j^{(k-1)}|}{\max(|x_j^{(k)}|, |x_j^{(k-1)}|, XSIZE)} \leq r$$

The default value for $r$ is 1E–8 for robust estimation (see the ROBUST option) with the iteratively
rewighted least squares method, and it is 0 for other estimation methods. The default value for
XSIZE is 0. You can change this default value by specifying the XSIZE= option in the NLOPTIONS
statement.
**BOUNDS Statement**

```
BOUNDS constraint <, constraint . . .> ;
```

where `constraint` represents

```
< number operator > parameter-list < operator number >
```

You can use the BOUNDS statement to define boundary constraints for any independent parameter that has its name specified in the main or subsidiary model specification statements, the PARAMETERS statement, or the INMODEL= data set. You cannot define boundary constraints for dependent parameters created in SAS programming statements or elsewhere.

Valid operators are `<`, `<=`, `>=`, `>`, and `=` (or, equivalently, LE, LT, GE, GT, and EQ). The following is an example of the BOUNDS statement:

```
bounds 0. <= a1-a9 x <= 1. ,
       -1. <= c2-c5 ,
       b1-b10 y >= 0. ;
```

You must separate boundary constraints with a comma, and you can specify more than one BOUNDS statement. The feasible region for a parameter is the intersection of all boundary constraints specified for that parameter; if a parameter has a maximum lower boundary constraint greater than its minimum upper bound, the parameter is set equal to the minimum of the upper bounds.

The active set strategies made available in PROC CALIS treat strict inequality constraints `<` or `>` as if they were just inequality constraints `<=` or `>=`. For example, if you require `x` be strictly greater than zero so as to prevent an undefined value for `y = log(x)`, specifying the following statement is insufficient:

```
BOUNDS x > 0;
```

Specify the following statement instead:

```
BOUNDS x > 1E-8;
```

If the CALIS procedure encounters negative variance estimates during the minimization process, serious convergence problems can occur. You can use the BOUNDS statement to constrain these parameters to nonnegative values. Although allowing negative values for variances might lead to a better model fit with smaller $\chi^2$ value, it adds difficulties in interpretation.

**BY Statement**

```
BY variables ;
```

You can specify a BY statement with PROC CALIS to obtain separate analyses of observations in groups that are defined by the BY variables. When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables. If you specify more than one BY statement, only the last one specified is used.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data by using the SORT procedure with a similar BY statement.
• Specify the NOTSORTED or DESCENDING option in the BY statement for the CALIS procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.

• Create an index on the BY variables by using the DATASETS procedure (in Base SAS software).

The BY statement is not supported if you define more than one group by using the GROUP statements.

For more information about BY-group processing, see the discussion in SAS Language Reference: Concepts. For more information about the DATASETS procedure, see the discussion in the Base SAS Procedures Guide.

COSAN Statement

COSAN < VAR=variable-list, > term < + term . . . > ;

where variable-list is a list of observed variables and term represents either one of the following forms:

matrix-definition < * matrix-definition . . . > < mean-definition>

or

mean-definition

where matrix-definition is of the following form:

matrix-name < ( number-of-columns < , matrix-type < , transformation >> ) >

and mean-definition is one of the following forms:

[ / mean-vector]

or

[ MEAN=mean-vector]

where mean-vector is a vector name.

COSAN stands for covariance structure analysis (McDonald 1978, 1980). The COSAN model in PROC CALIS is a generalized version of the original COSAN model. See the section “The COSAN Model” on page 1390 for details of the generalized COSAN model. You can analyze a very wide class of mean and covariance structures with the COSAN modeling language, which consists of the COSAN statement as the main model specification statement and the MATRIX statement as the subsidiary model specification statement. Use the following syntax to specify a COSAN model:

COSAN < VAR=variable-list, > term < + term . . . > ;

MATRIX matrix-name parameters-in-matrix ;
//” Repeat the MATRIX statement as needed */ ;

VARNAMES name-assignments ;

The PROC CALIS statement invokes the COSAN modeling language. You can specify at most one COSAN statement in a model within the scope of either the PROC CALIS statement or a MODEL statement. To complete the COSAN model specification, you might need to add as many MATRIX statements as needed. Optionally, you can provide the variable names for the COSAN model matrices in the VARNAMES statement.
In the COSAN statement, you specify the list of observed variables for analysis in the VAR= option and the formulas for covariance and mean structures in the terms. If specified at all, the VAR= option must be specified at the very beginning of the COSAN statement. The order of the variables in the VAR= option is important. It is the same order assumed for the row and column variables in the mean and covariance structures defined in the terms. If you do not specify the VAR= option, PROC CALIS selects all the numerical variables in the associated groups for analysis. To avoid confusion about the variables being analyzed in the model, it is recommended that you set the VAR= list explicitly in the COSAN statement.

To define the matrix formulas for the covariance and mean structures, you specify the terms, matrix-definitions, and mean-vector in the COSAN statement. The forms of the covariance and mean structures that are supported in PROC CALIS are mentioned in the section “The COSAN Model” on page 1390. In each term, you specify the covariance structures by listing the matrices in the matrix-definitions. These matrices must be in the proper order such that their matrix product produces the intended covariance structures. If you want to analyze the corresponding mean structures, specify the trailing mean-vectors in the terms whenever needed.

To illustrate the COSAN statement syntax, consider a factor-analytic model with six variables (var1–var6) and two factors. The covariance structures of the six variables are described by the matrix formula

\[ \Sigma = FPF' + U \]

where \( \Sigma \) is a 6 \times 6 symmetric matrix for the covariance matrix, \( F \) is a 6 \times 2 factor loading matrix, \( P \) is a 2 \times 2 (symmetric) factor covariance matrix, and \( U \) is a 6 \times 6 diagonal matrix of unique variances. You can use the following COSAN statement to specify the covariance structures of this factor model:

```
  cosan var = var1-var6,  
           F(2,GEN) * P(2,SYM) + U(6,DIA);
```

In the VAR= option of the COSAN statement, you define a list of six observed variables in the covariance structures. The order of the variables in the VAR= list determines the order of the row variables in the first matrix of each term in the model. That is, both matrices \( F \) and \( U \) have these six observed variables as their row variables, which are ordered the same way as in the VAR= list.

Next, you define the formula for the covariance structures by listing the matrices in the desired order up to the central covariance matrix in each term. In the first term of this example, you need to specify only \( FP \) instead of the complete covariance structure formula \( FPF' \). The reason is that the latter part of the term (that is, after the central covariance matrix) contains only the transpose of the matrices that have already been defined. Hence, PROC CALIS can easily generate the complete term with the nonredundant information given.

In each of the matrix-definitions, you can provide the number of columns in the first argument (that is, the number-of-columns field) inside a pair of parentheses. You do not need to provide the number of rows because this information can be deduced from the given covariance structure formula. By using some keywords, you can optionally provide the matrix type in the second argument (that is, the matrix-type field) and the matrix transformation in the third argument (that is, the transformation field).

In the current example, \( F(2, GEN) \) represents a general rectangular (GEN) matrix \( F \) with two columns. Implicitly, it has six rows because it is the first matrix of the first term in the covariance structure formula. \( P(2, SYM) \) represents a symmetric (SYM) matrix \( P \) with two columns. Implicitly, it has two rows because it is premultiplied with \( F \), which has two columns. In the second term, \( U(6, DIA) \) represents a diagonal (DIA) matrix \( U \) with six rows and six columns. Because you do not specify the third argument in these matrix-definitions, no transformation is applied to any of the matrices in the covariance structure formula.
Chapter 29: The CALIS Procedure

PROC CALIS supports the following keywords for matrix-type:

IDE specifies an identity matrix. If the matrix is not square, this specification describes an identity submatrix followed by a rectangular zero submatrix.

ZID specifies an identity matrix. If the matrix is not square, this specification describes a rectangular zero submatrix followed by an identity submatrix.

DIA specifies a diagonal matrix. If the matrix is not square, this specification describes a diagonal submatrix followed by a rectangular zero submatrix.

ZDI specifies a diagonal matrix. If the matrix is not square, this specification describes a rectangular zero submatrix followed by a diagonal submatrix.

LOW specifies a lower triangular matrix. The matrix can be rectangular.

UPP specifies an upper triangular matrix. The matrix can be rectangular.

SYM specifies a symmetric matrix. The matrix cannot be rectangular.

GEN specifies a general rectangular matrix (default).

If you omit the matrix-type argument, PROC CALIS sets the type of matrix by default. For central covariance matrices, the default for matrix-type is SYM. For all other matrices, the default for matrix-type is GEN. For example, if A is not a central covariance matrix in the covariance structure formula, the following specifications are equivalent for a general matrix A with three columns:

A(3, GEN)

A(3)

A(3,

A(3, ,)

A(3, ,)

PROC CALIS supports the following two keywords for transformation:

INV uses the inverse of the matrix.

IMI uses the inverse of the difference between the identity and the matrix. For example, A(3, GEN, IMI) represents \((I - A)^{-1}\).

Both INV or IMI require square (but not necessarily symmetric) matrices to transform. If you omit the transformation argument, no transformation is applied.

CAUTION: You can specify the same matrix by using the same matrix-name in different locations of the matrix formula in the COSAN statement. The number-of-columns and the matrix-type fields for matrices with identical matrix-names must be consistent. This consistency can be maintained easily by specifying each of these two fields only once in any of the locations of the same matrix. However, there is no restriction on the transformation for the same matrix in different locations. For example, while R must be the same 3 × 3 symmetric matrix throughout the formula in the following specification, the INV transformation of R applies only to the R matrix in the second term, but not to the same R matrix in the first term:

\[
\text{cosan var = var1-var6,}
\]

\[
B(3, \text{GEN}) \ast R(3, \text{SYM}) + H(3, \text{DIA}) \ast R(3, \text{SYM, INV});
\]

Mean and Covariance Structures

Suppose now you want to analyze the mean structures in addition to the covariance structures of the preceding factor model. The mean structure formula for \(\mu\) of the observed variables is

\[
\mu = Fv + a
\]
where $\mu$ is a $6 \times 1$ vector for the observed variable means, $v$ is a $2 \times 1$ vector for the factor means, and $a$ is a $6 \times 1$ vector for the intercepts of the observed variables. To include the mean structures in the COSAN model, you need to specify the mean vector at the end of the terms, as shown in the following statement:

```cosan var = var1-var6,  
F(2,GEN) * P(2,SYM) [/ v] + U(6,DIA) [/ a];```

If you take the mean vectors within the brackets away from each of the terms, the formula for the covariance structures is generated as

$$\Sigma = FPF' + U$$

which is exactly the same covariance structure as described in a preceding example. Now, with the mean vectors specified at the end of each term, you analyze the corresponding mean structures simultaneously with the covariance structures.

To generate the mean structure formula, PROC CALIS replaces the central covariance matrices with the mean vectors in the terms. In the current example the mean structure formula is formed by replacing $P$ and $U$ with $v$ and $a$, respectively. Hence, the first term of the mean structure formula is $F * v$, and the second term of the mean structure formula is simply $a$. Adding these two terms yields the desired mean structure formula for the model.

To make the mean vector specification more explicit, you can use the following equivalent syntax with the MEAN= option:

```cosan var = var1-var6,  
F(2,GEN) * P(2,SYM) [mean=v] + U(6,DIA) [mean=a];```

If a term in the specification does not have a mean vector (covariance matrix) specification, a zero mean vector (null covariance matrix) is assumed. For example, the following specification generates the same mean and covariance structures as the preceding example:

```cosan var = var1-var6,  
F(2,GEN) * P(2,SYM) [/ v] + U(6,DIA) + [/ a];```

The covariance structure formula for this specification is

$$\Sigma = FPF' + U + 0$$

where $0$ in the last term represents a null matrix. The corresponding mean structure formula is

$$\mu = Fv + 0 + a$$

where $0$ in the second term represents a zero vector.

### Specifying Models with No Explicit Central Covariance Matrices

In some situations, the central covariance matrices in the covariance structure formula are not defined explicitly. For example, the covariance structure formula for an orthogonal factor model is:

$$\Sigma = FF' + U$$

Again, assuming that $F$ is a $6 \times 2$ factor loading matrix and $U$ is a $6 \times 6$ diagonal matrix for unique variances, you can specify the covariance structure formula as in the following COSAN statement:
In determining the proper formula for the covariance structures, PROC CALIS detects whether the last matrix specified in each term is symmetric. If you specify this last matrix explicitly with the SYM, IDE (with the same number of rows and columns), or DIA type, it is certainly a symmetric matrix. If you specify this last matrix without an explicit matrix-type and it has the same number of rows and columns, it is also treated as a symmetric matrix for the central covariance matrix of the term. Otherwise, this last matrix is not symmetric and PROC CALIS treats the term as if an identity matrix has been inserted for the central covariance matrix. For example, for the orthogonal factor model specified in the preceding statement, PROC CALIS correctly generates the first term as $FF' = FIF'$ and the second term as $U$.

Certainly, you might also specify your own central covariance matrix explicitly for the orthogonal factor model. That is, you add an identity matrix into the COSAN model specification as shown in the following statement:

$$\text{cosan var} = \text{var1-var6},\quad F(2,\text{GEN}) \ast I(2,\text{IDE}) + U(6,\text{DIA});$$

### Specifying Mean Structures for Models with No Central Covariance Matrices

When you specify covariance structures with central covariance matrices explicitly defined in the terms, the corresponding mean structure formula is formed by replacing the central covariance matrices with the mean-vectors that are specified in the brackets. However, when there is no central covariance matrix explicitly specified in a term, the last matrix of the term in the covariance structure formula is replaced with the mean-vector to generate the mean structure formula. Consider the following specification where there is no central covariance matrix defined explicitly in the first term of the COSAN model:

$$\text{cosan var} = \text{var1-var6},\quad A(6,\text{GEN}) \begin{bmatrix} v \end{bmatrix};$$

The generated formulas for the covariance and mean structures are

$$\Sigma = AA'$$
$$\mu = v$$

If, instead, you intend to fit the following covariance and mean structures

$$\Sigma = AA'$$
$$\mu = Av$$

you must put an explicit identity matrix for the central covariance matrix in the first term. That is, you can use the following specification:

$$\text{cosan var} = \text{var1-var6},\quad A(6,\text{GEN}) \ast I(6,\text{IDE}) \begin{bmatrix} v \end{bmatrix};$$

### Specifying Parameters in Matrices

By specifying the COSAN statement, you define the covariance and mean structures in matrix formulas for the observed variables. To specify the parameters in the model matrices, you need to use the MATRIX statements.
For example, for an orthogonal factor model with six variables (var1–var6) and two factors, the $6 \times 2$ factor loading matrix $F$ might take the following form:

$$
F = \begin{pmatrix}
    x & 0 \\
    x & 0 \\
    x & 0 \\
    0 & x \\
    0 & x \\
    0 & x \\
\end{pmatrix}
$$

The $6 \times 6$ unique variance matrix $U$ might take the following form:

$$
U = \begin{pmatrix}
    x & 0 & 0 & 0 & 0 & 0 \\
    0 & x & 0 & 0 & 0 & 0 \\
    0 & 0 & x & 0 & 0 & 0 \\
    0 & 0 & 0 & x & 0 & 0 \\
    0 & 0 & 0 & 0 & x & 0 \\
    0 & 0 & 0 & 0 & 0 & x \\
\end{pmatrix}
$$

where each $x$ in the matrices represents a free parameter to estimate and 0 represents a fixed zero value. The covariance structures for the observed variables are described by the following formula:

$$
\Sigma = FF' + U
$$

To specify the entire model, you use the following statements to define the covariance structure formula and the free parameters in the model matrices:

```sas
    cosan var = var1-var6,
    F(2,GEN) + U(6,DIA);
    matrix F [1 to 3,@1],[4 to 6,@2];
    matrix U [1,1],[2,2],[3,3],[4,4],[5,5],[6,6];
```

In the MATRIX statements, you specify the free parameters in the matrices. For the factor loading matrix $F$, you specify that rows 1, 2, and 3 in column 1 and rows 4, 5, and 6 in column 2 are free parameters. For the unique variance matrix $U$, you specify that all diagonal elements are free parameters. All other unspecified entries in the matrices are fixed zeros by default. Certainly, you can also specify fixed zeros explicitly. For the current example, you can specify matrix $F$ equivalently as:

```sas
    matrix F [1 to 3,@1],[4 to 6,@2],
        [4 to 6,@1] = 0. 0. 0.,
        [1 to 3,@2] = 0. 0. 0.;
```

See the MATRIX statement on page 1295 for various ways to specify the parameters in matrices.

**Matrix Names versus Parameter Names**

Although parameter names and matrix names in PROC CALIS are both arbitrary SAS names for denoting mathematical entities in the model, their usages are very different in one aspect. That is, parameter names are globally defined in the procedure, while matrix names are only locally defined in models.

Consider the following two-group analysis example:
proc calis;
  group 1 / data=g1;
  group 2 / data=g2;
  model 1 / group=1;
    cosan var = var1-var6,
           F(2,GEN) * I(2,IDE) + U(6,DIA);
    matrix F [1 to 3,@1],[4 to 6,@2];
    matrix U [1,1] = u1-u6;
  model 2 / group=2;
    cosan var = var1-var6,
           F(1,GEN) * I(1,IDE) + D(6,DIA);
    matrix F [1 to 6,@1];
    matrix D [1,1] = u1-u6;
run;

In this example, you fit Model 1 to Group 1 and Model 2 to Group 2. You specify a matrix called F in each of the models. However, the two models are not constrained by this “same” matrix F. In fact, matrix F in Model 1 is a 6 x 2 matrix but matrix F in Model 2 is a 6 x 1 matrix. In addition, none of the parameters in the F matrices are constrained by the parameter names (simply because no parameter names are used). This illustrates that matrix names in PROC CALIS are defined only locally within models.

In contrast, in this example you use different matrix names for the second terms of the two models. In Model 1, you define a 6 x 6 diagonal matrix U for the second term; and in Model 2, you define a 6 x 6 diagonal matrix D for the second term. Are these two matrices necessarily different? The answer depends on how you define the parameters in these matrices. In the MATRIX statement for U, all diagonal elements of U are specified as free parameters u1–u6. Similarly, in the MATRIX statement for D, all diagonal elements of D are also specified as free parameters u1–u6. Because you use the same sets of parameter names in both of these MATRIX statements, matrices U and D are essentially constrained to be the same even though their names are different. This illustrates that parameter names are defined globally in PROC CALIS.

The following points summarize how PROC CALIS treats matrix and parameter names differently:

- Matrices with the same name in the same model are treated as identical.
- Matrices with the same name in different models are not treated as identical.
- Parameters with the same name are identical throughout the entire PROC CALIS specification.
- Cross-model constraints on matrix elements are set by using the same parameter names, but not the same matrix names.

**Row and Column Variable Names for Matrices**

You can use the VARNAMES statement to define the column variable names for the model matrices of a COSAN model. However, you do not specify the row variable names for the model matrices directly because they are determined by the column variable names of the related matrices in the covariance and mean structure formulas. For example, the following specification names the column variables of matrices F and I:
cosan var = var1-var6,
    F(2, GEN) * I(2, IDE) + U(6, DIA);
varnames
    F = [Factor1 Factor2],
    I = F;

The column names for matrix F are Factor1 and Factor2. The row names of matrix F are var1–var6 because it is the first matrix in the first term. Matrix I has the same column variable names as those for matrix F, as specified in the last specification of the VARNAMES statement. Because matrix I is a central covariance matrix, its row variable names are the same as its column variable names: Factor1 and Factor2. You do not specify the column variables names for matrix U in the VARNAMES statement. However, because it is the first matrix in the second term, its row variable names are the same as that of the VAR= list in the COSAN statement. Because matrix U is also the central covariance matrix in the second term, its column variable names are the same its row variable names, which has been determined to be var1–var6. See the VARNAMES statement for more details.

Default Parameters

Unlike other modeling languages in PROC CALIS, the COSAN modeling language does not set any default free parameters for the model matrices. There is only one type of default parameters in the COSAN model: fixed values for matrix elements. These fixed values can be 0 or 1. For matrices with the IDE or ZID type, all elements are predetermined with either 0 or 1. They are fixed matrices in the sense that you cannot override these default fixed values. For all other matrix types, PROC CALIS sets their elements to fixed zeros by default. You can override these default zeros by specifying them explicitly in the MATRIX statements.

Modifying a COSAN Model from a Reference Model

In this section, it is assumed that you use a REFMODEL statement within the scope of a MODEL statement and the reference model (or base model) is also a COSAN model. The reference model is referred to as the old model, while the model that makes reference to this old model is referred to as the new model. If the new model is not intended to be an exact copy of the old model, you can use the following extended COSAN modeling language to make modifications within the scope of the MODEL statement for the new model. The syntax is similar to, but not exactly the same as, the ordinary COSAN modeling language. (See the section “COSAN Statement” on page 1242.) The respecification syntax for a COSAN model is as follows:

```
COSAN ;
    MATRIX matrix-name parameters-in-matrix ;
    /* Repeat the MATRIX statement as needed */ ;
    VARNAMES name-assignments ;
```

In the respecification, the COSAN statement is optional. In fact, the purpose of using the COSAN statement at all is to remind yourself that a COSAN model is used in the model definition. If you use the COSAN statement, you cannot specify the VAR= option or the covariance and mean structure formula. This means that the model form and the observed variable references of the new model must be the same as the old (reference) model. The reason for enforcing these model structures is to ensure that the MATRIX statement respecifications are consistently interpreted.

You can optionally use the VARNAMES statement in the respecification. If the variable names for a COSAN matrix are defined in the old model but not redefined the new model, all variable names for that matrix are duplicated in the new model. However, specification of variable names for a COSAN matrix in the new model overrides the corresponding specification in the old model.
You can respecify or modify the elements of the COSAN model matrices by using the `MATRIX matrix-name` statements. The syntax of the MATRIX statements for respecifications is the same as that in the ordinary COSAN modeling language, but with one more feature. In the respecification syntax, you can use the missing value `.` to drop a parameter specification from the old model.

The new model is formed by integrating with the old model in the following ways:

- **Duplication:** If you do not specify in the new model a parameter location that exists in the old model, the old parameter specification is duplicated in the new model.
- **Addition:** If you specify in the new model a parameter location that does not exist in the old model, the new parameter specification is used in the new model.
- **Deletion:** If you specify in the new model a parameter location that also exists in the old model and the new parameter is denoted by the missing value `.` , the old parameter specification is not copied into the new model.
- **Replacement:** If you specify in the new model a parameter location that also exists in the old model and the new parameter is not denoted by the missing value `.` , the new parameter specification replaces the old one in the new model.

For example, the following two-group analysis specifies Model 2 by referring to Model 1 in the `REFMODEL` statement:

```plaintext
proc calis;
  group 1 / data=d1;
  group 2 / data=d2;
  model 1 / group=1;
    cosan
      var = x1-x6,
      F(2,GEN) * PHI(2,SYM) + PSI(6,SYM);
    matrix F
      [1,1] = 1.,
      [2,1] = load2,
      [3,1] = load3,
      [4,2] = 1.,
      [5,2] = load5,
      [6,2] = load6;
    matrix PHI
      [1,1] = phi1,
      [2,2] = phi2,
      [2,1] = phi21;
    matrix PSI
      [1,1] = psi1,
      [2,2] = psi2,
      [3,3] = psi3,
      [4,4] = psi4,
      [5,5] = psi5,
      [6,6] = psi6;
    varnames F = [Factor1 Factor2],
                PHI = F;
  model 2 / group=2;
    refmodel 1;
    matrix F
      [3,1] = load2; /* replacement */
    matrix PHI
      [2,1] = .; /* deletion */
    matrix PSI
      [3,1] = psi31; /* addition */
    varnames F = [FF1 FF2],
run;
```
In this example, Model 2 is the new model which refers to the old model, Model 1. It illustrates the four types of model integration by using the MATRIX statements:

- Duplication: Except for the $F[3,1]$ and $\Phi[2,1]$ elements, all parameter specifications in the old model are duplicated in the new model.

- Addition: The $\Psi[3,1]$ element is added with a new parameter $\psi_{31}$ in the new model. This indicates the presence of a correlated error in Model 2, but not in Model 1.

- Deletion: The $\Phi[2,1]$ element is no longer a free parameter in the new model. This means that the two latent factors are correlated in Model 1, but not in Model 2.

- Replacement: The $F[3,1]$ element defined in Model 2 replaces the definition in the old model. This element is now a free parameter named $load2$. Because the $F[2,1]$ element (via duplication from the old model) is also a free parameter with this same name, $F[3,1]$ and $F[2,1]$ are constrained to be the same in Model 2, but not in Model 1.

With the VARNAMES statement specification in Model 1, the two columns of matrix $F$ are labeled with $Factor1$ and $Factor2$, respectively. In addition, because PHI=F is specified in the VARNAMES statement of Model 1, the row and column of matrix $\Phi$ in Model 1 also contain $Factor1$ and $Factor2$ as the variable names. In Model 2, with the explicit VARNAMES specifications the two columns of matrix $F$ are labeled with $FF1$ and $FF2$, respectively. These names are not the same as those for matrix $F$ in the old (reference) model. However, because PHI=F is not specified in the VARNAMES statement of Model 2, the row and column of matrix $\Phi$ in Model 2 contain $Factor1$ and $Factor2$ as the variable names, which are duplicated from the old (reference) model.

**COSAN Models and Other Models**

Because the COSAN model is a more general model than any other model considered in PROC CALIS, you can virtually fit any other type of model in PROC CALIS by using the COSAN modeling language. See the section “Special Cases of the Generalized COSAN Model” on page 1392, Example 29.29, and Example 29.30 for illustrations and discussions.

In general, it is recommended that you use the more specific modeling languages such as FACTOR, LINEQS, LISMOD, MSTRUCT, PATH, and RAM. Because the COSAN model is very general in its formulation, PROC CALIS cannot exploit the specific model structures to generate reasonable initial estimates the way it does with other specific models such as FACTOR and PATH. If you do not provide initial estimates for a COSAN model, PROC CALIS uses some default starting values such as 0.5. See the START= option for controlling the starting value. See the RANDOM= option for setting random starting values. There are other reasons for preferring specific modeling languages whenever possible. The section “Which Modeling Language?” on page 1194 discusses these various reasons. However, when the covariance structures are complicated and are difficult to specify otherwise, the COSAN modeling language is a very useful tool. See Example 29.31 and Example 29.33 for illustrations.
COV Statement

\[
\text{COV } \text{assignment } <, \text{assignment } . . . > ;
\]

where \text{assignment} represents

\[
\text{var-list } < * \text{var-list2 } > < = \text{parameter-spec}>
\]

The COV statement is a subsidiary model specification statement for the confirmatory FACTOR and LINEQS models. In the LINEQS model, the COV statement defines the covariances among the exogenous variables, including errors and disturbances. In the confirmatory FACTOR model, the COV statement defines the factor covariances. In each \text{assignment} of the COV statement, you specify variables in the \text{var-list} and the \text{var-list2} lists, followed by the covariance parameter specification in the \text{parameter-spec} list. The latter two specifications are optional.

You can specify the following five types of the parameters for the covariances:

- an unnamed free parameter
- an initial value
- a fixed value
- a free parameter with a name provided
- a free parameter with a name and initial value provided

Consider a LINEQS model with exogenous variables \text{V1}, \text{V2}, \text{V3}, and \text{V4}. The following COV statement shows the five types of specifications in five \text{assignments}:

\[
\text{cov } \text{V2 V1 ,}
\text{V3 V1 } = (0.3),
\text{V3 V2 } = 1.0,
\text{V4 V1 } = \text{phi1,}
\text{V4 V2 } = \text{phi2(0.2)};
\]

In this statement, \text{cov (V2, V1)} is specified as an unnamed free parameter. For this covariance, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1). \text{cov (V3, V1)} is an unnamed free parameter but with an initial value of 0.3. PROC CALIS also generates a parameter name for this covariance. \text{cov (V3, V2)} is a fixed value of 1.0. This value stays the same in the estimation. \text{cov (V4, V1)} is a free parameter named phi1. \text{cov (V4, V2)} is a free parameter named phi2 with an initial value of 0.2.

Note that the \text{var-list} and \text{var-list2} lists on the left-hand side of the equal sign of the COV statement should contain only names of exogenous variables. Hence, the COV statement is different from the PCOV statement in which you can list both exogenous and endogenous variables, although the COV and PCOV statements share the same syntax.

You can use the COV statement for specifying covariance parameters in the FACTOR and LINEQS models. In the FACTOR model, the COV statement specifies the covariances among latent factors. In the LINEQS model, the COV statement specifies the covariances among all observed or latent exogenous variables, including error and disturbance terms.
If you specify only the var-list list, then you are specifying the so-called within-list covariances. If you specify both of the var-list and var-list2 lists, then you are specifying the so-called between-list covariances. An asterisk is used to separate the two variable lists. You can use one of these two alternatives to specify the covariance parameters. Figure 29.2 illustrates the within-list and between-list covariance specifications.

**Figure 29.2** Within-List and Between-List Covariances

**Within-List Covariances**

The left panel of the figure shows that the same set of four variables are used in both the rows and columns. This yields six nonredundant covariances to specify. In general, with a var-list list with \( k \) variables in the COV statement, there are \( k(k - 1)/2 \) distinct covariance parameters you can specify. The variable order of the var-list list is important. For example, the left panel of Figure 29.2 corresponds to the following COV statement specification:

```
cov E1-E4 = phi1-phi6;
```

This specification is equivalent to the following specification:

```
cov E2 E1 = phi1, 
E3 E1 = phi2, E3 E2 = phi3, 
E4 E1 = phi4, E4 E2 = phi5, E4 E3 = phi6;
```

Another way to assign distinct parameter names with the same prefix is to use the so-called prefix-name. For example, the following COV statement specification is exactly the same as the preceding specification:

```
cov E1-E4 = 6*phi__; /* phi with two trailing underscores */
```

In the COV statement, \( phi__ \) is a prefix-name with the root \( phi \). The notation \( 6* \) means this prefix-name is applied six times, resulting in a generation of the six parameter names \( phi1, phi2, \ldots, phi6 \) for the six covariance parameters.

The root of the prefix-name should have few characters so that the generated parameter name is not longer than 32 characters. To avoid unintentional equality constraints, the prefix-names should not coincide with other parameter names.
You can also specify the within-list covariances as unnamed free parameters, as shown in the following statement:

\[ \text{cov E1-E4;} \]

This specification is equivalent to the following specification:

\[ \text{cov E2 E1,} \]
\[ \quad \text{E3 E1, E3 E2,} \]
\[ \quad \text{E4 E1, E4 E2, E4 E3;} \]

**Between-List Covariances**

The right panel of Figure 29.2 illustrates the application of the between-list covariance specification. The set of row variables is different from the set of column variables. You intend to specify the cross covariances of the two sets of variables. There are four of these covariances in the figure. In general, with \( k_1 \) and \( k_2 \) variable names in the two variable lists (separated by an asterisk) in a COV statement, there are \( k_1 \times k_2 \) distinct covariances to specify. Again, variable order is very important. For example, the right panel of Figure 29.2 corresponds to the following between-list covariance specification:

\[ \text{cov E1 E2 * E3 E4 = phi1-phi4;} \]

This is equivalent to the following specification:

\[ \text{cov E1 E3 = phi1, E1 E4 = phi2,} \]
\[ \quad \text{E2 E3 = phi3, E2 E4 = phi4;} \]

You can also use the prefix-name specification for the same specification, as shown in the following statement:

\[ \text{cov E1 E2 * E3 E4 = 4*phi__;} /* phi with two trailing underscores */ \]

**Mixed Parameter Lists**

You can specify different types of parameters for the list of covariances. For example, you use a list of parameters with mixed types in the following statement:

\[ \text{cov E1-E4 = phi1(0.1) 0.2 phi3 phi4(0.4) (0.5) phi6;} \]

This specification is equivalent to the following specification:

\[ \text{cov E2 E1 = phi1(0.1),} \]
\[ \quad \text{E3 E1 = 0.2, E3 E2 = phi3,} \]
\[ \quad \text{E4 E1 = phi4(0.4), E4 E2 = (0.5), E4 E3 = phi6;} \]

Notice that an initial value that follows a parameter name is associated with the free parameter. Therefore, in the original mixed list specification, 0.1 is interpreted as the initial value for the parameter \( \phi_1 \), but not as the initial estimate for the covariance between \( E_3 \) and \( E_1 \). Similarly, 0.4 is the initial value for the parameter \( \phi_4 \), but not the initial estimate for the covariance between \( E_4 \) and \( E_2 \).

However, if you indeed want to specify that \( \phi_1 \) is a free parameter without an initial value and 0.1 is an initial estimate for the covariance between \( E_3 \) and \( E_1 \) (while keeping all other things the same), you can use a null initial value specification for the parameter \( \phi_1 \), as shown in the following statement:
\[ \text{cov } E_1-E_4 = \phi_1() (0.1) \phi_3 \phi_4(0.4) (0.5) \phi_6; \]

This way 0.1 becomes the initial estimate for the covariance between \(E_3\) and \(E_1\). Because a parameter list with mixed types might be confusing, you can break down the specifications into separate assignments to remove ambiguities. For example, you can use the following equivalent specification:

\[
\begin{align*}
\text{cov } E_2 & \ E_1 = \phi_1, \\
E_3 & \ E_1 = (0.1), \ E_3 \ E_2 = \phi_3, \\
E_4 & \ E_1 = \phi_4(0.4), \ E_4 \ E_2 = (0.5), \ E_4 \ E_3 = \phi_6;
\end{align*}
\]

**Shorter and Longer Parameter Lists**

If you provide fewer parameters than the number of covariances in the variable lists, all the remaining parameters are treated as unnamed free parameters. For example, the following specification assigns a fixed value to \(\text{cov}(E_1,E_3)\) while treating all the other three covariances as unnamed free parameters:

\[
\text{cov } E_1 \ E_2 \ast E_3 \ E_4 = 1.0;
\]

This specification is equivalent to the following specification:

\[
\text{cov } E_1 \ E_3 = 1.0, \ E_1 \ E_4, \ E_2 \ E_3, \ E_2 \ E_4;
\]

If you intend to fill up all values by the last parameter specification in the list, you can use the continuation syntax \([...], [...]\), or \([..]\), as shown in the following example:

\[
\text{cov } E_1 \ E_2 \ast E_3 \ E_4 = 1.0 \ \phi_1 \ [..];
\]

This means that \(\text{cov}(E_1,E_3)\) is a fixed value of 1 and all the remaining three covariances are free parameter named \(\phi_1\). The last three covariances are thus constrained to be equal by using the same parameter name.

However, you must be careful not to provide too many parameters. For example, the following specification results in an error:

\[
\text{cov } E_1 \ E_2 \ast E_3 \ E_4 = 1.0 \ \phi_2(2.0) \ \phi_3 \ \phi_4 \ \phi_5 \ \phi_6;
\]

The parameters after \(\phi_4\) are excessive.

**Default Covariance Parameters**

In the confirmatory FACTOR model, by default all factor covariances are free parameters. In the LINEQS model, by default all covariances among exogenous manifest and latent variables (excluding error or disturbance variables) are also free parameters. For these default free parameters, PROC CALIS generate the parameter names with the \(_\text{Add}\) prefix and appended with unique integer suffixes. You can also use the COV statement specification to override these default covariance parameters in situations where you want to set parameter constraints, provide initial or fixed values, or make parameter references.

Another type of default covariances are fixed zeros. In the LINEQS model, covariances among errors or disturbances are all fixed zeros by default. Again, you can override the default fixed values by providing explicit specification of these covariances in the COV statement.
Modifying a Covariance Parameter Specification from a Reference Model

If you define a new model by using a reference (old) model in the REFMODEL statement, you might want to modify some parameter specifications from the COV statement of the reference model before transferring the specifications to the new model. To change a particular covariance specification from the reference model, you can simply respecify the same covariance with the desired parameter specification in the COV statement of the new model. To delete a particular covariance parameter from the reference model, you can specify the desired covariance with a missing value specification in the COV statement of the new model.

For example, suppose that the covariance between variables \( V1 \) and \( V2 \) is specified in the reference model but you do not want this covariance specification be transferred to the new model. You can use the following COV statement specification in the new model:

\[
\text{cov } \text{V1 V2 } = .;
\]

Note that the missing value syntax is valid only when you use it with the REFMODEL statement. See the section “Modifying a LINEQS Model from a Reference Model” on page 1280 for a more detailed example of the LINEQS model respecification with the REFMODEL statement. See the section “Modifying a FACTOR Model from a Reference Model” on page 1266 for a more detailed example of the FACTOR model respecification with the REFMODEL statement.

As discussed in a preceding section, PROC CALIS generates some default free covariance parameters for the LINEQS and FACTOR models if you do not specify them explicitly in the COV statement. When you use the REFMODEL statement for defining a reference model, these default free covariance parameters in the old (reference) model are not transferred to the new model. Instead, the new model generates its own set of default free covariance parameters after it is resolved from the reference model, the REFMODEL statement options, the RENAMEPARM statement, and the COV statement specifications in the new model. This also implies that if you want any of the covariance parameters to be constrained across the models by means of the REFMODEL specification, you must specify them explicitly in the COV statement of the reference model so that the same covariance specification is transferred to the new model.

DETERM Statement

DETERM | STRUCTEQ variables </option> ;

where option represents:

LABEL | NAME= name

The DETERM statement is used to compute the determination coefficient of the listed dependent variables in the model. The precursor of the DETERM statement is the STRUCTEQ statement, which enables you to define the list of the dependent variables of the structural equations. Because the term structural equation is not defined in a unique way, a more generic concept of determination coefficients is revealed by the DETERM statement.

You can specify the DETERM statement as many times as you want for computing determination coefficients for the sets of dependent variables of interest. You can label each set of dependent variables by using the LABEL= option. Note that you cannot use the DETERM statement in an MSTRUCT model because there are no dependent variables in this type of model.
**EFFPART Statement**

**EFFPART** `effect <, effect . . . > ;`

where `effect` represents:

`var-list < direction var-list2>`

and `direction` is the direction of the effect, as indicated by one of the following:

```===>, <->>, ==>, -->, >, <=>, <===, <===, <=-, <=-```

In the **EFFPART** statement, you select those effects you want to analyze by partitioning the total effects into direct and indirect effects, with estimated standard errors. The **EFFPART** or **TOTEFF** option of the PROC CALIS statement also enables you to analyze effects. The difference is that the **EFFPART** or **TOTEFF** option displays effects on *all* endogenous variables, while the **EFFPART** statement shows only the effects of interest. In addition, the **EFFPART** statement enables you to arrange the effects in any way you like. Hence, the **EFFPART** statement offers a more precise and organized way to present various results of effects.

The **EFFPART** statement supports the following three types of effect specifications:

- **>**, `=>`, `<->`, `==>`, `<--`, `====>`, or `------` direction
  
  Example:

  ```effpart X1 X3-X5 ===> Y1 Y2;```  
  
  This will display *four* separate tables, respectively for the effects of `X1`, `X3`, `X4`, and `X5` on `Y1` and `Y2`. Each table contains the total, direct, and indirect effects of an `X`-variable on the two `Y`-variables.

- `<`, `<=`, `<->`, `<==`, `<---`, `<===`, or `<-----` direction
  
  Example:

  ```effpart Y1 Y2 <=== X1 X3-X5;```  
  
  This will display *two* separate tables, respectively for the effects on `Y1` and `Y2`, by `X1`, `X3`, `X4`, and `X5`. Each table contains the total, direct, and indirect effects of the four `X`-variables on a `Y`-variable. Certainly, the results produced from this statement are essentially the same as the previous statement. The difference is about the organization of the effects in the tables.

- no direction
  
  Example:

  ```effpart Y1 Y2 X1-X3;```  
  
  In this case, variables on the list are analyzed one by one to determine the nature of the effects. If a variable has nonzero effects on any other variables in the model, a table of the total, direct, and indirect effects of the variable on those variables is displayed. If a variable is endogenous, a table of total, direct, and indirect effects of those variables that have nonzero effects on the variable is displayed.
Note that an endogenous variable in a model might also have effects on other endogenous variables. Therefore, the two cases mentioned are not mutually exclusive—a variable listed in the EFFPART statement might yield two tables for effect analysis.

## FACTOR Statement

FACTOR <EFA-options | CFA-spec> ;

where *EFA-options* are options for the exploratory factor analysis that are described in the section “Exploratory Factor Analysis” on page 1258 and *CFA-spec* is a specification of confirmatory factor analysis that is described in the section “Confirmatory Factor Analysis” on page 1262.

In the FACTOR statement, you can specify either *EFA-options*, *CFA-spec*, or neither of these. However, you cannot specify both *EFA-options* and *CFA-spec* at the same time. If no option is specified or there is at least one *EFA-option* (exploratory factor analysis option) specified in the FACTOR statement, an exploratory factor model is analyzed. Otherwise, a confirmatory factor model is analyzed with the *CFA-spec*. These two types of models are discussed in the next two sections.

## Exploratory Factor Analysis

FACTOR <EFA-options> ;

For the exploratory factor model with orthogonal factors, PROC CALIS assumes the following model structures for the population covariance or correlation matrix $\Sigma$:

$$\Sigma = FF^\prime + U$$

where $F$ is the factor loading matrix and $U$ is a diagonal matrix of error variances. In this section, $p$ denotes the number of manifest variables corresponding to the rows and columns of matrix $\Sigma$, and $n$ denotes the number of factors (or components, if the COMPONENT option is specified in the FACTOR statement) corresponding to the columns of the factor loading matrix $F$. While the number of manifest variables is set automatically by the number of variables in the VAR statement or in the input data set, the number of factors can be set by the N= option in the FACTOR statement.

The unrestricted exploratory factor model is not identified because any orthogonal rotated factor loading matrix $\tilde{F} = F\Theta$ satisfies the same model structures as $F$ does, where $\Theta$ is any orthogonal matrix so that $\Theta^\prime\Theta = \Theta\Theta^\prime = I$. Mathematically, the covariance or correlation structures can be expressed as:

$$\Sigma = \tilde{F}\tilde{F}^\prime + U = F\Theta\Theta^\prime F^\prime + U = FF^\prime + U$$

To obtain an identified orthogonal factor solution as a starting point, the $n(n - 1)/2$ elements in the upper triangle of $F$ are constrained to zeros in PROC CALIS. Initial estimates for factor loadings and unique variances are computed by an algebraic method of approximate factor analysis. Given the initial estimates, final estimates are obtained through the iterative optimization of an objective function, which depends on the estimation method specified in the METHOD= option (default with ML—maximum likelihood) of the PROC CALIS statement.

To make the factor solution more interpretable, you can use the ROTATE= option in the FACTOR statement to obtain a rotated factor loading matrix with a “simple” pattern. Rotation can be orthogonal or oblique.
The rotated factors remain uncorrelated after an orthogonal rotation but would be correlated after an oblique rotation. The model structures of an oblique solution are expressed in the following equation:

\[ \Sigma = \tilde{\mathbf{F}} \mathbf{P} \tilde{\mathbf{F}}' + \mathbf{U} \]

where \( \tilde{\mathbf{F}} \) is the rotated factor loading matrix and \( \mathbf{P} \) is a symmetric matrix for factor correlations. See the sections “The FACTOR Model” on page 1394 and “Exploratory Factor Analysis Models” on page 1395 for more details about exploratory factor models.

You can also do exploratory factor analysis by the more dedicated FACTOR procedure. Even though extensive comparisons of the factor analysis capabilities between the FACTOR and CALIS procedures are not attempted here, some general points can be made here. In general, the FACTOR procedure provides more factor analysis options than the CALIS procedure does, although both procedures have some unique factor analysis features that are not shared by the other. PROC CALIS requires more computing time and memory than PROC FACTOR because it is designed for more general structural estimation problems and is not able to exploit all the special properties of the unconstrained factor analysis model. For maximum likelihood analysis, you can use either PROC FACTOR (with METHOD=ML, which is not the default method in PROC FACTOR) or PROC CALIS. Because the initial unrotated factor solution obtained by PROC FACTOR uses a different set of identification constraints than that of PROC CALIS, you would observe different initial ML factor solutions for the procedures. Nonetheless, the initial solutions by both procedures are statistically equivalent.

The following \textit{EFA-options} are available in the FACTOR statement:

\textbf{COMPONENT}

\textbf{COMP} computes a component analysis instead of a factor analysis (the diagonal matrix \( \mathbf{U} \) in the model is set to 0). Note that the rank of \( \mathbf{FF}' \) is equal to the number \( n \) of components in \( \mathbf{F} \). If \( n \) is smaller than the number of variables in the moment matrix \( \Sigma \), the matrix of predicted model values is singular and maximum likelihood estimates for \( \mathbf{F} \) cannot be computed. You should compute ULS estimates in this case.

\textbf{HEYWOOD}

\textbf{HEY} constrains the diagonal elements of \( \mathbf{U} \) to be nonnegative. Equivalently, you can constrain these elements to positive values by the \textit{BOUNDS} statement.

\textbf{GAMMA=}\( p \)

specifies the orthomax weight used with the option \textit{ROTATE=ORTHOMAX}. Alternatively, you can use \textit{ROTATE=ORTHOMAX}\( (p) \) with \( p \) representing the orthomax weight. There is no restriction on valid values for the orthomax weight, although the most common values are between 0 and the number of variables. The default \textit{GAMMA=} value is one, resulting in the varimax rotation.

\textbf{N=}\( n \)

specifies the number of first-order factors or components. The number of factors \( (n) \) should not exceed the number of manifest variables \( (p) \) in the analysis. For the saturated model with \( n = p \), the COMP option should generally be specified for \( \mathbf{U} = 0 \); otherwise, \( df < 0 \). For \( n = 0 \) no factor loadings are estimated, and the model is \( \Sigma = \mathbf{U} \), with a diagonal \( \mathbf{U} \) matrix. By default, \( n = 1 \).
NORM< = KAISER | NONE >  
Kaiser-normalizes the rows of the factor pattern for rotation. NORM=KAISER, which is the default, has exactly the same effect as NORM. You can turn off the normalization by NORM=NONE.

RCONVERGE=\( p \)
RCONV=\( p \)
specifies the convergence criterion for rotation cycles. Rotation stops when the scaled change of the simplicity function value is less than the RCONVERGE= value. The default convergence criterion is:

\[
\frac{|f_{\text{new}} - f_{\text{old}}|}{K} < \epsilon
\]

where \( f_{\text{new}} \) and \( f_{\text{old}} \) are simplicity function values of the current cycle and the previous cycle, respectively, \( K = \max(1, |f_{\text{old}}|) \) is a scaling factor, and \( \epsilon \) is 1E–9 by default and is modified by the RCONVERGE= value.

\( RITER= i \)

specifies the maximum number of cycles \( i \) for factor rotation. The default \( i \) is the greater of 10 times the number of variables and 100.

\( \text{ROTATE=} \, \text{name} \)

\( R= \, \text{name} \)
specifies an orthogonal or oblique rotation of the initial factor solution. Although ROTATE=PRINCIPAL is actually not a rotation method, it is put here for convenience. By default, ROTATE=NONE.

Valid names for orthogonal rotations are as follows:

- **BIQUARTIMAX | BIQMAX** specifies orthogonal biquartimax rotation. This corresponds to the specification ROTATE=ORTHOMAX(0.5).
- **EQUAMAX | E** specifies orthogonal equamax rotation. This corresponds to the specification ROTATE=ORTHOMAX with GAMMA=\( n/2 \).
- **FACTORPARSIMAX | FPA** specifies orthogonal factor parsimax rotation. This corresponds to the specification ROTATE=ORTHOMAX with GAMMA=\( n \).
- **NONE | N** specifies that no rotation be performed, leaving the original orthogonal solution.
- **ORTHCF(\( p_1, p_2 \)) | ORCF(\( p_1, p_2 \))** specifies the orthogonal Crawford-Ferguson rotation (Crawford and Ferguson 1970) with the weights \( p_1 \) and \( p_2 \) for variable parsimony and factor parsimony, respectively. See the definitions of weights in Chapter 37, “The FACTOR Procedure.”
- **ORTHGENCF(\( p_1, p_2, p_3, p_4 \)) | ORGENCF(\( p_1, p_2, p_3, p_4 \))** specifies the orthogonal generalized Crawford-Ferguson rotation (Jennrich 1973), with the four weights \( p_1, p_2, p_3, \) and \( p_4 \). For the definitions of these weights, see the section “Simplicity Functions for Rotations” on page 2337 in Chapter 37, “The FACTOR Procedure.”
- **ORTHOMAX(\( p_1 \)) | ORMAX(\( p_1 \))** specifies the orthomax rotation (see Harman 1976) with orthomax weight \( p_1 \). If ROTATE=ORTHOMAX is used, the default \( p_1 \) value is 1 unless specified otherwise in the GAMMA= option. Alternatively, ROTATE=ORTHOMAX(\( p_1 \)) specifies \( p_1 \) as the orthomax weight or the GAMMA= value. For the definitions of the orthomax weight, see the section “Simplicity Functions for Rotations” on page 2337 in Chapter 37, “The FACTOR Procedure.”
PARSIMAX | PA specifies orthogonal parsimax rotation. This corresponds to the specification
\[
\text{ROTATE=ORTHOMAX with}
\]
\[
\text{GAMMA} = \frac{p \times (n-1)}{p + n - 2}
\]
PRINCIPAL | PC specifies a principal axis rotation. If \text{ROTATE=PRINCIPAL} is used with a factor
rather than a component model, the following rotation is performed:
\[
F_{\text{new}} = F_{\text{old}}T, \quad \text{with} \quad F'_{\text{old}}F_{\text{old}} = TT'
\]
where the columns of matrix \(T\) contain the eigenvectors of \(F'_{\text{old}}F_{\text{old}}\).
QUARTIMAX | QMAX | Q specifies orthogonal quartimax rotation. This corresponds to the specification
\text{ROTATE=ORTHOMAX(0)}.
VARIMAX | V specifies orthogonal varimax rotation. This corresponds to the specification
\text{ROTATE=ORTHOMAX with GAMMA=1}.
Valid \textit{names} for oblique rotations are as follows:

BIQUARTIMIN | BIQMIN specifies biquartimin rotation. It corresponds to the specification
\text{ROTATE=OBLIMIN(.5) or ROTATE=OBLIMIN with TAU=.5}.
COVARIMIN | CVMIN specifies covarimin rotation. It corresponds to the specification
\text{ROTATE=OBLIMIN(1) or ROTATE=OBLIMIN with TAU=1}.
OBBIQUARTIMAX | OBIQMAX specifies oblique biquartimax rotation.
OBEQUAMAX | OE specifies oblique equamax rotation.
OBFACTORPARSIMAX | OFPA specifies oblique factor parsimax rotation.
OBLICF\((p1,p2)\) | OBCF\((p1,p2)\) specifies the oblique Crawford-Ferguson rotation (Crawford and
Ferguson 1970) with the weights \(p1\) and \(p2\) for variable parsimony and factor
parsimony, respectively. For the definitions of these weights, see the section
“Simplicity Functions for Rotations” on page 2337 in Chapter 37, “The FACTOR
Procedure.”
OBLIGENCF\((p1,p2,p3,p4)\) | OBGENCF\((p1,p2,p3,p4)\) specifies the oblique generalized Crawford-
Ferguson rotation (Jennrich 1973) with the four weights \(p1\), \(p2\), \(p3\), and \(p4\). For the
definitions of these weights, see the section “Simplicity Functions for Rotations”
on page 2337 in Chapter 37, “The FACTOR Procedure.”
OBLIMIN\((p)\) | OBLMIN\((p)\) specifies the oblimin rotation with oblimin weight \(p\). If
\text{ROTATE=OBLIMIN} is used, the default \(p\) value is zero unless specified oth-
erwise in the TAU= option. Alternatively, \text{ROTATE=OBLIMIN(p1)} specifies \(p1\) as
the oblimin weight or the TAU= value. For the definitions of the oblimin weight,
see the section “Simplicity Functions for Rotations” on page 2337 in Chapter 37,
“The FACTOR Procedure.”
OBPARSIMAX | OPA specifies oblique parsimax rotation.
OBQUARTIMAX | OQMAX specifies oblique quartimax rotation. This is the same as the QUAR-
TIMIN method.
OBVARIMAX | OV specifies oblique varimax rotation.
QUARTIMIN | QMIN specifies quartimin rotation. It is the same as the oblique quartimax method. It also corresponds to the specification ROTATE=OBLIMIN(0) or ROTATE=OBLIMIN with TAU=0.

TAU=\(p\)
specifies the oblimin weight used with the option ROTATE=OBLIMIN. Alternatively, you can use ROTATE=OBLIMIN(\(p\)) with \(p\) representing the oblimin weight. There is no restriction on valid values for the oblimin weight, although for practical purposes a negative or zero value is recommended. The default TAU= value is 0, resulting in the quartimin rotation.

Confirmatory Factor Analysis

FACTOR factor-variables-relation <, factor-variables-relation . . . >;
where each factor-variables-relation is defined as

\[ \text{factor right-arrow var-list} < = \text{parameter-spec}> \]

where right-arrow is one of the following: \(\rightarrow\), \(--->\), \(==->\), \(==>\), \(=>\), \(-->\), or \(>\).

To complete the specification of a confirmatory factor model, you might need to use the PVAR, COV, and MEAN statements to specify the variance, partial variance, covariance, and mean parameters in the model, as shown in the following syntax:

FACTOR factor-variable-relation <, factor-variables-relation . . . >;
    PVAR partial-variance-parameters;
    COV covariance-parameters;
    MEAN mean-parameters;

The model structures for the covariance matrix \(\Sigma\) of the confirmatory factor model are described in the equation

\[ \Sigma = FPF' + U \]

where \(F\) is the factor loading matrix, \(P\) is a symmetric matrix for factor correlations, and \(U\) is a diagonal matrix of error variances.

If the mean structures are also analyzed, the model structures for the mean vector \(\mu\) of the confirmatory factor model are described in the equation

\[ \mu = \alpha + Fv \]

where \(\alpha\) is the intercept vector for the observed variables and \(v\) is the vector for factor means. See the sections “The FACTOR Model” on page 1394 and “Confirmatory Factor Analysis Models” on page 1397 for more details about confirmatory factor models.

The FACTOR statement is the main model specification statement for the confirmatory factor model. The specifications in the FACTOR statement concern the factor loading pattern in the \(F\) matrix. More details follow after a brief description of the subsidiary model specification statements: PVAR, COV, and MEAN.

By default, the factor variance parameters in the diagonal of matrix \(P\) and the error variances in the diagonal of matrix \(U\) are free parameters in the confirmatory factor model. However, you can override these default parameters by specifying them explicitly in the PVAR statement. For example, in some confirmatory factor
models, you might want to set some of these variances to fixed constants, or you might want to set equality constraints by using the same parameter name at different parameter locations in your model.

By default, factor covariances, which are the off-diagonal elements of matrix $P$, are free parameters in the confirmatory factor model. However, you can override these default covariance parameters by specifying them explicitly in the COV statement. Note that you cannot use the COV statement to specify the error covariances—they are always fixed zeros in the confirmatory factor analysis model.

By default, all factor means are fixed zeros and all intercepts are free parameters if the mean structures are analyzed. You can override these defaults by explicitly specifying the means of the factors in vector $\alpha$ and the intercepts of the manifest variables in vector $\beta$ in the MEAN statement.

Because the default parameterization of the confirmatory FACTOR model already covers most commonly used parameters in matrices $P$, $U$, $\alpha$, and $\beta$, the specifications in the PVAR, COV, and MEAN statements are secondary to the specifications in the FACTOR statement, which specifies the factor pattern of the $F$ matrix. The following example statement introduces the syntax of the confirmatory FACTOR statement. Suppose that there are nine manifest variables $V_1$–$V_9$ in your sample and you want to fit a model with four factors, as shown in the following FACTOR statement:

```
factor
g_factor ===> V1-V9 ,
factor_a ===> V1-V3 ,
factor_b ===> V4-V6 ,
factor_c ===> V7-V9 ;
```

In this factor model, you assume a general factor $g_{factor}$ and three group-factors: $factor_a$, $factor_b$, and $factor_c$. The general factor $g_{factor}$ is related to all manifest variables in the sample, while each group-factor is related only to three manifest variables. This example fits the following pattern of factor pattern of $F$:

<table>
<thead>
<tr>
<th>fact</th>
<th>$g_{factor}$</th>
<th>$factor_a$</th>
<th>$factor_b$</th>
<th>$factor_c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_1$</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_2$</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_3$</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_4$</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_5$</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>$V_6$</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>$V_7$</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>$V_8$</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_9$</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where an $x$ represents an unnamed free parameter and all other cells that are blank are fixed zeros. For each of these unnamed parameters, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1, _Parm2 and so on).

An unnamed free parameter is only one of the following five types of parameters (parameter-spec) you can specify at the end of each factor-variables-relation:

- an unnamed free parameter
- an initial value
• a fixed value
• a free parameter with a name provided
• a free parameter with a name and initial value provided

To illustrate these different types of parameter specifications, consider the following factor pattern for $F$:

\[
\begin{array}{llll}
g\_factor & factor\_a & factor\_b & factor\_c \\
V1 & g\_load1 & 1. \\
V2 & g\_load2 & x \\
V3 & g\_load3 & x \\
V4 & g\_load4 & 1. \\
V5 & g\_load5 & load\_a \\
V6 & g\_load6 & load\_b \\
V7 & g\_load7 & 1. \\
V8 & g\_load8 & load\_c \\
V9 & g\_load9 & load\_c \\
\end{array}
\]

where an $x$ represents an unnamed free parameter, a constant 1 represents a fixed value, and each name in a cell represents a name for a free parameter. You can specify this factor pattern by using the following FACTOR statement:

\[
\begin{align*}
\text{factor} & \quad \text{g\_factor} \quad \Rightarrow \quad \text{V1-V9} = \text{g\_load1-g\_load9} (9*0.6), \\
\text{factor\_a} & \quad \Rightarrow \quad \text{V1-V3} = 1. \ (0.7 \ 0.8), \\
\text{factor\_b} & \quad \Rightarrow \quad \text{V4-V6} = 1. \ \text{load\_a} \ (0.9) \ \text{load\_b}, \\
\text{factor\_c} & \quad \Rightarrow \quad \text{V7-V9} = 1. \ 2*\text{load\_c} ; \\
\end{align*}
\]

In the first entry of the FACTOR statement, you specify that the loadings of $V1$–$V9$ on $g\_factor$ are free parameters $g\_load1$–$g\_load9$ with all given an initial estimate of 0.6. The syntax $9*0.6$ means that 0.6 is repeated nine times. Because they are enclosed in a pair parentheses, all these values are treated as initial estimates, but not fixed values.

The second entry of the FACTOR statement can be split into the following specification:

\[
\begin{align*}
\text{factor\_a} & \quad \Rightarrow \quad \text{V1} = 1. , \\
\text{factor\_a} & \quad \Rightarrow \quad \text{V2} = (0.7), \\
\text{factor\_a} & \quad \Rightarrow \quad \text{V3} = (0.8), \\
\end{align*}
\]

This means that the first loading is a fixed value of 1, while the other loadings are unnamed free parameters with initial estimates 0.7 and 0.8, respectively. For each of these unnamed parameters with initial values, PROC CALIS also generates a parameter name with the _Parm prefix and appended with a unique integer.

The third entry of the FACTOR statement can be split into the following specification:

\[
\begin{align*}
\text{factor\_b} & \quad \Rightarrow \quad \text{V4} = 1. , \\
\text{factor\_b} & \quad \Rightarrow \quad \text{V5} = \text{load\_a} \ (0.9), \\
\text{factor\_b} & \quad \Rightarrow \quad \text{V6} = \text{load\_b}, \\
\end{align*}
\]

This means that the first loading is a fixed value of 1, the second loading is a free parameter named $\text{load\_a}$ with an initial estimate of 0.9, and the third loading is a free parameter named $\text{load\_b}$ without an initial estimate. PROC CALIS generates the initial value for this free parameter.
The fourth entry of the FACTOR statement states that the first loading is a fixed 1 and the remaining two loadings are free parameters named load_c. No initial estimate is given. But because the two loadings have the same parameter name, they are constrained to be equal in the estimation.

Notice that an initial value that follows after a parameter name is associated with the free parameter. For example, in the third entry of the FACTOR statement, the specification (.9) after load_a is interpreted as the initial value for the parameter load_a, but not as the initial estimate for the next loading for V6.

However, if you indeed want to specify that load_a is a free parameter without an initial value and (0.9) is an initial estimate for the loading for V6, you can use a null initial value specification for the parameter load_a, as shown in the following specification:

\[
\text{factor_b} \rightarrow V4-V6 = 1. \text{ load}_a() \ (0.9),
\]

This way 0.9 becomes the initial estimate of the loading for V6. Because a parameter list with mixed parameter types might be confusing, you can split the specification into separate entries to remove ambiguities. For example, you can use the following equivalent specification:

\[
\begin{align*}
\text{factor_b} \rightarrow V4 &= 1., \\
\text{factor_b} \rightarrow V5 &= \text{load}_a, \\
\text{factor_b} \rightarrow V6 &= (0.9),
\end{align*}
\]

**Shorter and Longer Parameter Lists**

If you provide fewer parameters than the number of loadings that are specified in the corresponding factor-variable-relation, all the remaining parameters are treated as unnamed free parameters. For example, the following specification assigns a fixed value of 1.0 to the first loading, while treating the remaining two loadings as unnamed free parameters:

\[
\text{factor} \\
\text{factor_a} \rightarrow V1-V3 = 1.;
\]

This specification is equivalent to the following specification:

\[
\begin{align*}
\text{factor} \\
\text{factor_a} \rightarrow V1 &= 1., \\
\text{factor_a} \rightarrow V2 \ V3 &;
\end{align*}
\]

If you intend to fill up all values with the last parameter specification in the list, you can use the continuation syntax [ . . ], [ . . . ], or [ . ], as shown in the following example:

\[
\begin{align*}
\text{factor} \\
\text{g_factor} \rightarrow V1-V30 &= 1. \ (0.5) \ [ . . ];
\end{align*}
\]

This means that the loading of V1 on g_factor is a fixed value of 1.0, while the remaining 29 loadings are unnamed free parameters with all given an initial estimate of 0.5.

However, you must be careful not to provide too many parameters. For example, the following specification results in an error:

\[
\begin{align*}
\text{factor} \\
\text{g_factor} \rightarrow V1-V3 &= \text{load1-load6};
\end{align*}
\]

The parameter list has six parameters for three loadings. Parameters after load3 are excessive.
**Default Parameters**

It is important to understand the default parameters in the FACTOR model. First, if you know which parameters are default free parameters, you can make your specification more efficient by omitting the specifications of those parameters that can be set by default. For example, because all error variances in the confirmatory FACTOR model are free parameters by default, you do not need to specify them with the PVAR statement if these error variances are not constrained. Second, if you know which parameters are default free parameters, you can specify your model accurately. For example, because all factor variance and covariances in the confirmatory FACTOR model are free parameters by default, you must use the COV statement to restrict the covariances among the factors if you want to fit an orthogonal factor model. See the section “Default Parameters in the FACTOR Model” on page 1401 for details about the default parameters of the FACTOR model.

**Modifying a FACTOR Model from a Reference Model**

This section assumes that you use a REFMODEL statement within the scope of a MODEL statement and that the reference model (or base model) is a factor model, either exploratory or confirmatory. The reference model is called the old model, and the model that refers to the old model is called the new model. If the new model is not intended to be an exact copy of the old FACTOR model, you can use the extended FACTOR modeling language described in this section to make modifications from the old model before transferring the specifications to the new model.

Using the REFMODEL statement for defining new factor models is not recommended in the following cases:

- If your old model is an exploratory factor analysis model, then specification by using the FACTOR modeling language in the new model replaces the old model completely. In this case, the use of the REFMODEL statement is superfluous and should be avoided.

- If your old model is a confirmatory factor analysis model, then specification of an exploratory factor model by using the FACTOR statement in the new model also replaces the old model completely. Again, the use of the REFMODEL statement is superfluous and should be avoided.

The nontrivial case where you might find the REFMODEL statement useful is when you modify an old confirmatory factor model to form a new confirmatory factor model. This nontrivial case is the focus of discussion in the remaining of the section.

The extended FACTOR modeling language for modifying model specification bears the same syntax as that of the ordinary FACTOR modeling language (see the section “Confirmatory Factor Analysis” on page 1262). The syntax is:

```
FACTOR factor-variable-relation ;
PVAR partial-variance-parameters ;
COV covariance-parameters ;
MEAN mean-parameters ;
```

The new model is formed by integrating with the old model in the following ways:

**Duplication:** If you do not specify in the new model a parameter location that exists in the old model, the old parameter specification is duplicated in the new model.

**Addition:** If you specify in the new model a parameter location that does not exist in the old model, the new parameter specification is added in the new model.
Deletion: If you specify in the new model a parameter location that also exists in the old model and the new parameter is denoted by the missing value ‘.’, the old parameter specification is not copied into the new model.

Replacement: If you specify in the new model a parameter location that also exists in the old model and the new parameter is not denoted by the missing value ‘.’, the new parameter specification replaces the old one in the new model.

For example, consider the following two-group analysis:

```plaintext
proc calis;
  group 1 / data=d1;
  group 2 / data=d2;
  model 1 / group=1;
    factor
      F1 ===> V1-V3 = 1. load1 load2,
      F2 ===> V4-V6 = 1. load3 load4,
      F3 ===> V7-V9 = 1. load5 load6;
    cov
      F1 F2 = c12,
      F2 F3 = c23;
    pvar
      F1-F3 = c1-c3,
      V1-V9 = ev1-ev9;
  model 2 / group=2;
    refmodel 1;
    factor
      F1 ===> V1   = loada,
      F2 ===> V4   = loadb,
      F3 ===> V7   = loadc;
    cov
      F1 F2 = .,
      F1 F3 = c13;
run;
```

In this specification, you specify Model 2 by referring to Model 1 in the REFMODEL statement; Model 2 is the new model which refers to the old model, Model 1. Because the PVAR statement is not used in new model, all variance and partial variance parameter specifications in the PVAR statement of the old model are duplicated in the new model. The covariance parameter c23 for covariance between F2 and F3 in the COV statement of the old model is also duplicated in the new model. Similarly, loading parameters load1–load6 for some specific factor matrix locations are duplicated from the old model to the new model.

The new model has an additional parameter specification that the old model does not have. In the COV statement of the new model, covariance parameter c13 for the covariance between F1 and F3 is added.

In the same statement, the covariance between F1 and F2 is denoted by the missing value ‘.’. The missing value indicates that this parameter location in the old model should not be included in the new model. The consequence of this deletion from the old model is that the covariance between F1 and F2 is a fixed zero in the new model.

Finally, the three new loading specifications in the FACTOR statement of the new model replace the fixed ones in the old model. They are now free parameters loada, loadb, and loadc in the new model.
FITINDEX Statement

FITINDEX option < option . . . > ;

You can use the FITINDEX statement to set the options for computing and displaying the fit indices, or to output the fit indices. All but the OFF= and ON= options of the FITINDEX statement are also available in the PROC CALIS statement. The options set in the FITINDEX statement will overwrite those set in the PROC CALIS statement.

For the listing of fit indices and their definitions, see the section “Overall Model Fit Indices” on page 1484. Note that not all fit indices are available with all estimation methods, which is specified by the METHOD= option of the PROC CALIS statement. See the section “Fit Indices and Estimation Methods” on page 1491 for more details.

The options of the FITINDEX statement are as follows:

ALPHAECV=\alpha

specifies a (1 - \alpha)100\% confidence interval (0 \leq \alpha \leq 1) for the Browne and Cudeck (1993) expected cross validation index (ECVI). See the ALPHAECV= option of the PROC CALIS statement on page 1205.

ALPHARMS=\alpha

specifies a (1 - \alpha)100\% confidence interval (0 \leq \alpha \leq 1) for the Steiger and Lind (1980) root mean square error of approximation (RMSEA) coefficient. See the ALPHARMS= option of the PROC CALIS statement on page 1205.

BASEFIT=SAS-data-set

INBASEFIT=SAS-data-set

inputs the SAS-data-set that contains the fit information of the baseline model of your choice. See the BASEFIT= option of the PROC CALIS statement on page 1206.

BASEFUNC=r(< DF=>i)

BASEFUNC(< DF=>i)=r

inputs the fit function value \( r \) and the degrees of freedom \( i \) of the baseline model of your choice. See the BASEFUNC= option of the PROC CALIS statement on page 1207.

CHICORRECT= name | c

CHICORR= name | c

specifies a correction factor \( c \) for the chi-square statistics for model fit. See the CHICORRECT= option of the PROC CALIS statement on page 1209.

CLOSEFIT=\rho

defines the criterion value \( \rho \) for indicating a close fit. See the CLOSEFIT= option of the PROC CALIS statement on page 1211.

DFREDUCE=i

reduces the degrees of freedom of the \( \chi^2 \) test by \( i \). See the DFREDUCE= option of the PROC CALIS statement on page 1215.
NOADJDF
  turns off the automatic adjustment of degrees of freedom when there are active constraints in the analysis. See the NOADJDF option of the PROC CALIS statement on page 1225.

NOINDEXTYPE
  disables the display of index types in the fit summary table. See the NOINDEXTYPE option of the PROC CALIS statement on page 1225.

OFF= [names] | {names}
OFFLIST= [names] | {names}
  turns off the printing of one or more fit indices or modeling information as indicated by names, where a name represents a fit index, a group of fit indices, or modeling information. Names must be specified inside a pair of parentheses and separated by spaces. By default, all fit indices are printed. See the ON= option for the value of names.

ON < (ONLY) > = [names] | {names}
ONLIST < (ONLY) > = [names] | {names}
  turns on the printing of one or more fit indices or modeling information as indicated by names, where a name represents a fit index, a group of fit indices, or modeling information. Names must be specified inside a pair of parentheses and separated by spaces. Because all fit indices and modeling information are printed by default, using an ON= list alone is redundant. When both ON= and OFF= lists are specified, the ON= list will override the OFF= list for those fit indices or modeling information that appear on both lists. If an ON(ONLY)= list is used, only those fit indices or modeling information specified in the list will be printed. Effectively, an ON(ONLY)= list is the same as the specification with an ON= list with the same selections and an OFF=ALL list in the FITINDEX statement.

Output Control of Fit Index Groups and Modeling Information Group
You can use the following names to refer to the groups of fit indices or modeling information available in PROC CALIS:

  ABSOLUTE    Absolute or stand-alone fit indices that measures the model fit without using a baseline model.
  ALL         All fit indices available in PROC CALIS.
  INCREMENTAL Incremental fit indices that measure model fit by comparing with a baseline model.
  MODELINFO   General modeling information including sample size, number of variables, number of variables, and so on.
  PAR SIMONY   Fit indices that take model parsimony into account.

Output Control of Modeling Information
You can use the following names to refer to the individual modeling information available in PROC CALIS:

  BASECHISQ   Chi-square statistic for the baseline model.
  BASEDF      Degrees of freedom of the chi-square statistic for the baseline model.
  BASEFUNC    Baseline model function value.
  BASELOGLIKE Baseline model –2 log-likelihood function value for METHOD=FIML.
BASEPROBCHI  \( p \)-value of the chi-square statistic for the baseline model fit.

BASESTATUS Status of the baseline model fitting for METHOD=FIML.

NACTCON Number of active constraints.

NIOBS Number of incomplete observations for METHOD=FIML.

NMOMENTS Number of elements in the moment matrices being modeled.

NOBS Number of observations assumed in the analysis.

NPARM | NPARMS Number of independent parameters.

NVAR Number of variables.

SATFUNC Saturated model function value for METHOD=FIML.

SATLOGLIKE Saturated model –2 log-likelihood function value for METHOD=FIML.

SATSTATUS Status of the saturated model fitting for METHOD=FIML.

Output Control of Absolute Fit Indices
You can use the following names to refer to the individual absolute fit indices available in PROC CALIS:

CHISQ Chi-square statistic for model fit.

CN | CRITICAL_N Hoelter’s critical N.

CONTLIKE Percentage contribution to the Log-likelihood function value of each group in multiple-group analyses with METHOD=FIML.

CONTRIBUTION | CONTCHI Percentage contribution to the chi-square value for multiple-group analyses.

DF Degrees of freedom for the chi-square test for model fit.

ELLiptic Elliptical chi-square statistic for ML and GLS methods in single-group analyses without mean structures. This index is computed only when you input the raw data with the KURTOSIS option specified.

FUNCVAL Optimized function value.

GFI Goodness-of-fit index by Jöreskog and Sörbom.

LOGLIKE Fitted model –2 log-likelihood function value for METHOD=FIML.

PROBCHI \( p \)-value of the chi-square statistic for model fit.

PROBELLiptic \( p \)-value of the elliptical chi-square statistic.

RMR Root mean square residual.

SRMR Standardized root mean square residual.

ZTEST Z-test of Wilson and Hilferty.

Output Control of Parsimonious Fit Indices
You can use the following names to refer to the individual parsimonious fit indices available in PROC CALIS:
AGFI  Adjusted GFI.
AIC   Akaike information criterion.
CAIC  Bozdogan corrected AIC.
CENTRALITY  McDonald centrality measure.
ECVI  Expected cross-validation index.
ECVI_LL | LL_ECVI  Lower confidence limit for ECVI.
ECVI_UL | UL_ECVI  Upper confidence limit for ECVI.
PGFI  Parsimonious GFI.
PROBCLFIT  Probability of close fit.
RMSEA  Root mean square error of approximation.
RMSEA_LL | LL_RMSEA  Lower confidence limit for RMSEA.
RMSEA_UL | UL_RMSEA  Upper confidence limit for RMSEA.
SBC  Schwarz Bayesian criterion.

Output Control of Incremental Fit Indices
You can use the following names to refer to the individual incremental fit indices available in PROC CALIS:

BENTLERCFI | CFI  Bentler comparative fit index.
BENTLERNFI  Bentler-Bonett normed fit index.
BENTLERNNFI  Bentler-Bonett nonnormed fit index.
BOLLENNFI  Bollen normed fit index (Rho1).
BOLLENNNFI  Bollen nonnormed fit index (Delta2).
PNFI  James et al. parsimonious normed fit index.

OUTFIT=SAS-data-set
creates an output data set containing the values of the fit indices. This is the same as the OUTFIT= option of the PROC CALIS statement on page 1229. See the section “OUTFIT= Data Set” on page 1385 for details.

FREQ Statement

FREQ variable ;

If one variable in your data set represents the frequency of occurrence for the other values in the observation, specify the variable’s name in a FREQ statement. PROC CALIS then treats the data set as if each observation appears \( n_i \) times, where \( n_i \) is the value of the FREQ variable for observation \( i \). Only the integer portion of the value is used. If the value of the FREQ variable is less than 1 or is missing, that observation is not included in the analysis. The total number of observations is considered to be the sum of the FREQ values. You can use only one FREQ statement within the scope of each GROUP or the PROC CALIS statement.
GROUP Statement

GROUP i <= options> ;

where i is an assigned group number between 1 and 9999, inclusively.

The GROUP statement signifies the beginning of a group specification block and designates a group number for the group. All subsidiary group specification statements after a GROUP statement belong in that group until another MODEL or GROUP statement is used. The subsidiary group specification statements refer to one of the following four statements:

- FREQ statement on page 1271
- PARTIAL statement on page 1322
- VAR statement on page 1359
- WEIGHT statement on page 1367

For example, consider the following statements:

```
proc calis;
  var X1-X4;
  group 1 / label='Women' data=women_data;
    freq Z;
  group 2 / label='Men' data=men_data;
    partial P;
  model 1 / group = 1-2;
    factor N=1; /* One factor exploratory factor analysis */
run;
```

In the GROUP statements, two groups are defined. Group 1, labeled as ‘Women’, refers to the data set women_data. Group 2, labeled as ‘Men’, refers to the data set men_data. Both groups are fitted by an exploratory factor model defined in Model 1, as indicated in the GROUP= option of the MODEL statement. While the frequency variable Z defined in the FREQ statement is applicable only to Group 1, the partial variable P defined in the PARTIAL statement is applicable only to Group 2. However, the VAR statement, which appears before the definitions of both groups, applies globally to both Group 1 and Group 2. Therefore, variables X1–X4 are the analysis variables in the two groups.

You can set group-specific options in each GROUP statement. All but one (that is, the LABEL= option) of these options are also available in the MODEL and PROC CALIS statements. If you set these group-specific options in the PROC CALIS statement, they will apply to all groups unless you respecify them in the GROUP statement. If you set these group-specific options in the MODEL statement, they will apply to all groups that are fitted by the associated model. In general, the group-specific options are transferred from the PROC CALIS statement to the MODEL statements (if present) and then to the fitted groups. In the transferring process, options are overwritten by the newer ones. If you want to apply some group-specific options to a particular group only, you should set those options in the GROUP statement corresponding to that group.
Option Available in the GROUP Statement Only

**LABEL=** name

**NAME=** name

specifies a label for the current group. You can use any valid SAS names up to 256 characters for labels. You can also use quote strings for the labels. This option can be specified only in the GROUP statement, not the PROC CALIS statement.

Options Available in the GROUP and PROC CALIS Statements

These *options* are available in the GROUP and PROC CALIS statements:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA=</td>
<td>on page 1214 Specifies the input data set</td>
</tr>
<tr>
<td>INWGT=</td>
<td>on page 1217 Specifies the data set that contains the weight matrix</td>
</tr>
<tr>
<td>OUTSTAT=</td>
<td>on page 1229 Specifies the data set for storing the statistical results</td>
</tr>
<tr>
<td>OUTWGT=</td>
<td>on page 1230 Specifies the data set for storing the weight matrix</td>
</tr>
<tr>
<td>ROBITER=</td>
<td>on page 1235 Specifies the maximum number of iterations for estimating robust covariance and mean matrices</td>
</tr>
<tr>
<td>ROBPHI=</td>
<td>on page 1235 Specifies the tuning parameter for robust methods</td>
</tr>
</tbody>
</table>

See the section “PROC CALIS Statement Options” on page 1205 for more details about these *options*. If you specify these *options* in the PROC CALIS statement, they are transferred to all GROUP statements. They might be overwritten by the respecifications in the individual GROUP statements.

Options Available in GROUP, MODEL, and PROC CALIS Statements

These *options* are available in the GROUP, MODEL, and PROC CALIS statements:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHALEV=</td>
<td>on page 1205 Specifies the ( \alpha )-level criterion for detecting leverage points</td>
</tr>
<tr>
<td>ALPHAOUT=</td>
<td>on page 1205 Specifies the ( \alpha )-level criterion for detecting outliers</td>
</tr>
<tr>
<td>BIASKUR</td>
<td>on page 1209 Computes the skewness and kurtosis without bias corrections</td>
</tr>
<tr>
<td>EDF=</td>
<td>on page 1215 Defines nobs by the number of error df</td>
</tr>
<tr>
<td>INWGTINV</td>
<td>on page 1218 Specifies that the INWGT= data set contains the inverse of the weight matrix</td>
</tr>
<tr>
<td>KURTOSIS</td>
<td>on page 1218 Computes and displays kurtosis</td>
</tr>
<tr>
<td>MAXMISSPAT=</td>
<td>on page 1220 Specifies the maximum number of missing patterns to display</td>
</tr>
<tr>
<td>NOBS=</td>
<td>on page 1225 Defines the number of observations (nobs)</td>
</tr>
<tr>
<td>NOMISSPAT</td>
<td>on page 1226 Suppresses the display of missing pattern analysis</td>
</tr>
<tr>
<td>PCORR</td>
<td>on page 1230 Displays analyzed and estimated moment matrix</td>
</tr>
<tr>
<td>PLOTS=</td>
<td>on page 1232 Specifies ODS Graphics selection</td>
</tr>
<tr>
<td>PWEIGHT</td>
<td>on page 1367 Displays the weight matrix</td>
</tr>
<tr>
<td>RDF</td>
<td>DFR=</td>
</tr>
<tr>
<td>RESIDUAL</td>
<td>RES=</td>
</tr>
<tr>
<td>RESIDUAL</td>
<td>RES=</td>
</tr>
</tbody>
</table>
Chapter 29: The CALIS Procedure

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIDGE on page 1235</td>
<td>Specifies the ridge factor for covariance matrix</td>
</tr>
<tr>
<td>SIMPLE on page 1237</td>
<td>Prints univariate statistics</td>
</tr>
<tr>
<td>TMISSPAT= on page 1238</td>
<td>Specifies the data proportion threshold for displaying the missing patterns</td>
</tr>
<tr>
<td>VARDEF= on page 1239</td>
<td>Specifies variance divisor</td>
</tr>
<tr>
<td>WPENALTY= on page 1239</td>
<td>Specifies the penalty weight to fit correlations</td>
</tr>
<tr>
<td>WRIDGE= on page 1240</td>
<td>Specifies the ridge factor for the weight matrix</td>
</tr>
</tbody>
</table>

If you specify these options in the PROC CALIS statement, they are transferred to all MODEL statements. These options are overwritten by the respecifications in the individual MODEL statements. After these options are resolved in a given MODEL statement, they are transferred further to the GROUP statements of which the associated groups are fitted by the model. Again, these options might be overwritten by the respecifications in the individual GROUP statements.

LINCON Statement

```
LINCON constraint < , constraint . . . > ;
```

where `constraint` represents one of the following:

- `number operator linear-term`
- `linear-term operator number`

and `linear-term` is

```
< +|- > < coefficient * > parameter < +|- > < coefficient * > parameter . . . >
```

The LINCON statement specifies a set of linear equality or inequality constraints of the following form:

\[
\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, \ldots, m
\]

The constraints must be separated by commas. Each linear constraint \(i\) in the statement consists of a linear combination \(\sum_j a_{ij} x_j\) of a subset of the \(n\) parameters \(x_j, j = 1, \ldots, n\), and a constant value \(b_j\) separated by a comparison operator. Valid operators are \(<=, <, >=, >, and =\), or, equivalently, LE, LT, GE, GT, and EQ. PROC CALIS cannot enforce the strict inequalities \(<\) or \(>\). Note that the coefficients \(a_{ij}\) in the linear combination must be constant numbers and must be followed by an asterisk and the name of a parameter (that is, listed in the PARMS, main, or subsidiary model specification statements). The following is an example of the LINCON statement that sets a linear constraint on parameters \(x1\) and \(x2\):

```
lincon     x1 + 3 * x2 <= 1;
```

Although you can express boundary constraints of individual parameters in LINCON statements, it is much more convenient to specify these boundary constraints with the BOUNDS statement. For example, the following LINCON statement essentially specifies boundary constraints on four parameters:
lincon
  x1 <= 1,
x2 <= 1,
v1 > 0,
v2 > 0;

Instead of using the LINCON statement, you can specify the following BOUNDS statement:

bounds
  x1 x2 <= 1,
v1 v2 > 0;

Another advantage of using the BOUNDS statement for setting bounds for individual parameters is that when a boundary constraint becomes active in the solution (that is, when the final estimate is on the boundary), a Lagrange multiplier test on releasing the boundary constraint would be conducted when you also specify the MOD option. However, no Lagrange multiplier test would be conducted if you specified the boundary constraint with the LINCON statement.

**LINEQS Statement**

**LINEQS** < equation <, equation ... >> ;

where equation represents:

\[
\text{dependent} = \text{term} < + \text{term} \ldots >
\]

and each term represents one of the following:

- \(\text{coefficient-name} < (\text{number}) > < \ast > \text{variable-name}\)
- \(\text{prefix-name} < (\text{number}) > < \ast > \text{variable-name}\)
- \(< \text{number} > < \ast > \text{variable-name}\)

The LINEQS statement is a *main model specification statement* that invokes the LINEQS modeling language. You can specify at most one LINEQS statement in a model, within the scope of either the PROC CALIS statement or a MODEL statement. To completely specify a LINEQS model, you might need to add some *subsidiary model specification statements* such as the VARIANCE, COV, and MEAN statements. The syntax for the LINEQS modeling language is as follows:

**LINEQS** < equation <, equation ... >> ;

  **VARIANCE** partial-variance-parameters ;
  **COV** covariance-parameters ;
  **MEAN** mean-parameters ;

In the LINEQS statement, you use equations to specify the linear functional relations among manifest and latent variables. Equations in the LINEQS statement are separated by commas.

In the **VARIANCE** statement, you specify the variance parameters. In the **COV** statement, you specify the covariance parameters. In the **MEAN** statement, you specify the mean parameters. For details of these subsidiary model specification statements, see the syntax of these statements.
In the LINEQS statement, in addition to the functional relations among variables, you specify the coefficient parameters of interest in the equations. There are five types of parameters you can specify in equations, as shown in the following example:

\[
\text{lineqs} \\
V1 = * F1 + E1, \\
V2 = (.5) * F1 + E2, \\
V3 = 1. * F1 + E3, \\
V4 = b4 * F1 + E4; \\
V5 = b5 (.4) * F1 + E5;
\]

In this example, you have manifest variables \(V1\)–\(V5\), which are related to a latent factor, denoted by \(F1\), as specified in the equations. In each equation, you have one outcome variable (\(V\)-variable), one predictor variable (\(F1\), which is assumed to be a latent factor, the so-called \(F\)-variable), and one error variable (\(E\)-variable). The following four types of parameters have been specified:

- an unnamed free parameter
  The effect of \(F1\) on \(V1\) in the first equation is an unnamed free parameter. Although you specify nothing before the asterisk sign, the effect parameter is effectively specified. For an unnamed free parameter, PROC CALIS generates a parameter name with the \(_\text{Parm}\) prefix and appended with a unique integer (for example, \(_\text{Parm1}\), \(_\text{Parm2}\), and so on).

- an initial value
  The effect of \(F1\) on \(V2\) in the second equation is an unnamed free parameter with an initial estimate of 0.5. PROC CALIS also generates a parameter name for this specification. Notice that you must use a pair of parentheses for the initial value specification because it is interpreted as a fixed value otherwise, as described in the next case.

- a fixed value
  The effect of \(F1\) on \(V3\) in the third equation is an unnamed free parameter with a fixed value of 0.5. A fixed value remains the same in the estimation. There is no parameter name for a fixed constant in the model.

- a free parameter with a name
  The effect of \(F1\) on \(V4\) in the fourth equation is a free parameter named \(b4\). You do not provide an initial estimate for this free parameter.

- a free parameter with a name and an initial estimate
  The effect of \(F1\) on \(V5\) in the fifth equation is a free parameter named \(b5\) with an initial estimate of 0.4. Parameters with no starting values are initialized by various heuristic and effective methods in PROC CALIS. See the section “Initial Estimates” on page 1506 for details.

Notice that there must be an error term in each equation. The error terms in equation must start with the prefix ‘\(E\)’, ‘\(e\)’, ‘\(D\)’, or ‘\(d\)’. See the section “Representing Latent Variables in the LINEQS Model” on page 1278 for details about naming the factors and error terms. The effect or the path coefficient attached to an error term must be 1.0. This is implicitly specified as in the preceding example. For example, there is no parameter specification nor an asterisk sign before the error term \(E1\) in the first equation, as shown in the following:
V1 = * F1 + E1,

This specification is the same as the following explicit specification with a fixed constant 1.0 for the effect of the error term E1:

V1 = * F1 + 1. * E1,

The equivalence shown here implies that you can also specify the third equation in the following equivalent way:

V3 = F1 + E3,

This implicitly specifies a constant 1.0 for the effect of F1 on V3. You must be very careful about the distinction between this specification and the following one with an asterisk before F1:

V3 = * F1 + E3,

With an asterisk sign, the effect of F1 on V3 becomes an unnamed free parameter in the current specification. This interpretation is very different from the preceding one without an asterisk sign before F1, which assumes a fixed constant of 1.0.

Except for the unnamed free parameter specification, you can omit the asterisk signs in all other types of parameter specifications. That is, you can use the following equivalent statement for the preceding LINEQS specification:

\begin{verbatim}
lineqs
  V1 = * F1 + E1,
  V2 = (.5) F1 + E2,
  V3 = 1. F1 + E3,
  V4 = b4 F1 + E4;
  V5 = b5 (.4) F1 + E5;
\end{verbatim}

Again, you cannot omit the asterisk in the first equation because it is intended to denote an unnamed free parameter.

If your model contains many unconstrained parameters and it is too cumbersome to find different parameter names, you can specify all those parameters by the same prefix-name. A prefix name is a short name called “root” followed by two underscores __. Whenever a prefix-name is encountered, the CALIS procedure generates a parameter name by appending a unique integer to the root. Hence, the prefix-name should have few characters so that the generated parameter name is not longer than thirty-two characters. To avoid unintentional equality constraints, the prefix names should not coincide with explicitly defined parameter names. The following statement illustrates the uses prefix-names:

\begin{verbatim}
lineqs
  V1 = 1. * F1 + E1,
  V2 = b__ * F1 + E2,
  V3 = b__ * F1 + E3,
  V4 = b__ * F1 + E4;
  V5 = b__ * F1 + E5;
\end{verbatim}

In the five equations, only the first equation has a fixed constant 1.0 for the effect of F1 on V1. For all the remaining equations, the effects of F1 on the variables are all free parameters with the prefix b. The generated parameter names for these effects have unique integers appended to this prefix. For example, b1, b2, b3, and b4 are the parameter names for these effects.
Representing Latent Variables in the LINEQS Model

Because latent variables are widely used in structural equation modeling, PROC CALIS needs a way to identify different types of latent variables that are specified in the LINEQS model. This is accomplished by following some naming conventions for the latent variables. See the section “Naming Variables in the LINEQS Model” on page 1403 for details about these naming rules. Essentially, latent factors (systematic sources) must start with the letter ‘F’ or ‘f’. Error terms must start with the letter ‘E’, ‘e’, ‘D’, or ‘d’. Prefix ‘E’ or ‘e’ represents the error term of an endogenous manifest variable. Prefix ‘D’ or ‘d’ represents the disturbance (or error) term of an endogenous latent variable. Although D- and E- variables are conceptually different, for modeling purposes ‘D’ and ‘E’ prefixes are interchangeable in the LINEQS modeling language. Essentially, only the distinction between latent factors (systematic sources) and errors or disturbances (unsystematic sources) is critical in specifying a proper LINEQS model. Manifest variables in the LINEQS model do not need to follow additional naming rules beyond those required by the general SAS System—they are recognized by PROC CALIS by referring to the variables in the input data sets.

Types of Variables and Semantic Rules of Equations

Depending on their roles in the system of equations, variables in a LINEQS model can be classified into endogenous or exogenous. An endogenous variable is a variable that serves as an outcome variable (left-hand side of an equation) in one of the equations. All other variables are exogenous variables, including those manifest variables that do not appear in any equations but are included in the model because they are specified in the VAR statement for the analysis.

Merely following the syntactic rules described so far is not sufficient to define a proper system of equations that PROC CALIS can analyze. You also need to observe the following semantic rules:

- Only manifest or latent variables can be endogenous. This means that you cannot specify any error or disturbances variables on the left-hand side of the equations. This also means that error and disturbance variables are always exogenous in the LINEQS model.

- An endogenous variable that appears on the left-hand side of an equation cannot appear on the left-hand side of another equation. In other words, you have to specify all the predictors for an endogenous variable in a single equation.

- An endogenous variable that appears on the left-hand side of an equation cannot appear on the right-hand side of the same equation. This prevents a variable to have a direct effect on itself (but indirect effect on itself is possible).

- Each equation must contain one and only one unique error term, be it an E-variable for a manifest outcome variable or a D-variable for a latent outcome variable. If, indeed, you want to specify an equation without an error term, you can equivalently set the variance of the error term to a fixed zero in the VARIANCE statement.

Mean Structures in Equations

To fit a LINEQS model with mean structures, you can specify the MEANSTR option in the PROC CALIS or the associated MODEL statement. This generates the default mean and intercept parameters for the model (see the section “Default Parameters” on page 1279). Alternatively, you can specify the intercept parameters with the Intercept variable in the equations or the mean parameters in the MEAN statement. The Intercept
variable in the LINEQS model is a special “variable” that contains the value 1 for each observation. You do not need to have this variable in your input data set, nor do you need to generate it in the DATA step. It serves as a notational convenience in the LINEQS modeling language. The actual intercept parameter is expressed as a coefficient parameter with the intercept variable. For example, consider the following LINEQS model specification:

```
lineqs
  V1 = a1 (10) * Intercept + 1.0 * F1 + E1,
  V2 = * Intercept + * F1 + E2,
  V3 = + b2 * F1 + E3,
  V4 = a2 * Intercept + b2 * F1 + E4,
  V5 = a2 * Intercept + b4 (.4) * F1 + E5;
```

In the first equation, \( a1 \), with a starting value at 10, is the intercept parameter of \( V1 \). In the second equation, the intercept parameter of \( V2 \) is an unnamed free parameter. In the third equation, although you do not specify the \( \text{Intercept} \) variable, the intercept parameter of manifest variable \( V3 \) is assumed to be a free parameter by default. See the section “Default Parameters” on page 1279 for more details about default parameters. In the fourth and the fifth equations, the intercept parameters are both named \( a2 \). This means that these intercepts are constrained to be the same in the estimation.

In some cases, you might need to set the intercepts to fixed constants such as zeros. You can use the following syntax:

```
lineqs
  V1 = 0 * Intercept + F_intercept + a2 * F_slope + E1;
```

This sets the intercept parameter of \( V1 \) to a fixed zero. An example of this application is the analysis of latent growth curve model in which you define the intercept as a random variable represented by a latent factor (for example, \( F_{\text{intercept}} \) in the specification). See Example 29.25 for a detailed example.

To complete the specification of the mean structures in the LINEQS model, you might want to use the MEAN statement to specify the mean parameters. For example, the following statements specify the means of \( F_{\text{intercept}} \) and \( F_{\text{slope}} \) as unnamed free parameters in the LINEQS model:

```
lineqs
  V1 = 0 * Intercept + F_intercept + 1 * F_slope + E1;
mean
  F_intercept F_slope;
```

See the MEAN statement for details.

**Default Parameters**

It is important to understand the default parameters in the LINEQS model. First, if you know which parameters are default free parameters, you can make your specification more efficient by omitting the specifications of those parameters that can be set by default. For example, because all variances and covariances among exogenous variables (excluding error terms) are free parameters by default, you do not need to specify them with the COV and VARIANCE statements if these variances and covariances are not constrained. Second, if you know which parameters are default fixed zero parameters, you can specify your model accurately. For example, because all error covariances in the LINEQS model are fixed zeros by default, you must use the COV statement to specify the covariances among the errors if you want to fit a model with correlated errors. See the section “Default Parameters in the LINEQS Model” on page 1407 for details about the default parameters of the LINEQS model.
Modifying a LINEQS Model from a Reference Model

This section assumes that you use a REFMODEL statement within the scope of a MODEL statement and that the reference model (or base model) is a LINEQS model. The reference model is called the old model, and the model being defined is called the new model. If the new model is not intended to be an exact copy of the old model, you can use the extended LINEQS modeling language described in this section to make modifications within the scope of the MODEL statement for the new model.

The syntax of the extended LINEQS modeling language is the same as that of the ordinary LINEQS modeling language (see the section “LINEQS Statement” on page 1275):

```
LINEQS < equation < , equation . . . > > ;
VARIANCE partial-variance-parameters ;
COV covariance-parameters ;
MEAN mean-parameters ;
```

The new model is formed by integrating with the old model in the following ways:

**Duplication:** If you do not specify in the new model an equation with an outcome variable (that is, a variable on the left side of the equal sign) that exists in the old model, the equation with that outcome variable in the old model is duplicated in the new model. For specifications other than the LINEQS statement, if you do not specify in the new model a parameter location that exists in the old model, the old parameter specification is duplicated in the new model.

**Addition:** If you specify in the new model an equation with an outcome variable that does not exist as an outcome variable in the equations of the old model, the equation is added in the new model. For specifications other than the LINEQS statement, if you specify in the new model a parameter location that does not exist in the old model, the new parameter specification is added in the new model.

**Deletion:** If you specify in the new model an equation with an outcome variable that also exists as an outcome variable in an equation of the model and you specify the missing value '.' as the only term on the right-hand side of the equation in the new model, the equation with the same outcome variable in the old model is not copied into the new model. For specifications other than the LINEQS statement, if you specify in the new model a parameter location that also exists in the old model and the new parameter is denoted by the missing value '.', the old parameter specification is not copied into the new model.

**Replacement:** If you specify in the new model an equation with an outcome variable that also exists as an outcome variable in an equation of the model and the right-hand side of the equation in the new model is not denoted by the missing value '.', the new equation replaces the old equation with the same outcome variable in the new model. For specifications other than the LINEQS statement, if you specify in the new model a parameter location that also exists in the old model and the new parameter is not denoted by the missing value '.', the new parameter specification replaces the old one in the new model.
For example, the following two-group analysis specifies Model 2 by referring to Model 1 in the `REFMODEL` statement:

```r
proc calis;
  group 1 / data=d1;
  group 2 / data=d2;
  model 1 / group=1;
    lineqs
      V1 = 1 * F1  + E1,
      V2 = load1 * F1  + E2,
      V3 = load2 * F1  + E3,
      F1 = b1 * V4  + b2 * V5  + b3 * V6  + D1;
    variance
      E1-E3 = ve1-ve3,
      D1 = vd1,
      V4-V6 = phi4-phi6;
    cov
      E1 E2 = cve12;
  model 2 / group=2;
    refmodel 1;
    lineqs
      V3 = load1 * F1  + E3;
    cov
      E1 E2 = .,
      E2 E3 = cve23;
  run;
```

Model 2 is the new model which refers to the old model, Model 1. This example illustrates the four types of model integration:

- **Duplication:** All equations, except the one with outcome variable $V_3$, in the old model are duplicated in the new model. All specifications in the VARIANCE and COV statements, except the covariance between $E_1$ and $E_2$, in the old model are also duplicated in the new model.

- **Addition:** The parameter $cve23$ for the covariance between $E_2$ and $E_3$ is added in the new model.

- **Deletion:** The specification of covariance between $E_1$ and $E_2$ in the old model is not copied into the new model, as indicated by the missing value `.` specified in the new model.

- **Replacement:** The equation with $V_3$ as the outcome variable in the old model is replaced with a new equation in the model. The new equation uses parameter `load1` so that it is now constrained to be the same as the regression coefficient in the equation with $V_2$ as the outcome variable.
**LISMOD Statement**

```
LISMOD < var-lists > ;
```

where *var-lists* represent one or more of the following:

- **YVAR | YV | Y = var-list**
- **XVAR | XV | X = var-list**
- **ETAVAR | ETAV | ETA = var-list**
- **XIVAR | XIV | XI | KSIVAR | KSIV | KSI = var-list**

LISMOD stands for LISREL modeling, where LISREL is the program developed by Jöreskog and Sörbom (1988). Like the original implementation of LISREL, LISMOD uses a matrix specification interface. To complete the LISMOD specification, you might need to add as many **MATRIX** statements as needed, as shown in the following statement structure for the LISMOD model:

```
LISMOD var-lists ;
MATRIX matrix-name parameters-in-matrix ;
Repeat the MATRIX statement as needed ;
```

The *matrix-name* in the **MATRIX** statement should be one of the twelve model matrices in LISMOD, as listed in the following:

- Matrices in the structural model: _ALPHA_, _KAPPA_, _BETA_, _GAMMA_, _PHI_, or _PSI_
- Matrices in the measurement model for *y*-variables: _NUY_, _LAMBDAY_, or _THETAY_
- Matrices in the measurement model for *x*-variables: _NUX_, _LAMBDAX_, or _THETAX_

See the section “Model Matrices in the LISMOD Model” on page 1410 for definitions of these matrices and their roles in the LISMOD modeling language. See the **MATRIX statement** on page 1295 for the details of parameter specification.

In the LISMOD statement, you can specify the following four lists of variables:

- **YVAR=** list is for manifest variables *y* that are directly related to the endogenous latent variables *η* (eta). Variables in the list are called *y*-variables.
- **XVAR=** list is for manifest variables *x* that are directly related to the exogenous latent variables *ξ* (xi or ksi). Variables in the list are called *x*-variables.
- **ETAVAR=** list is for endogenous latent variables *η*. Variables in the list are called *η*-variables.
- **XIVAR=** list is for exogenous latent variables *ξ*. Variables in the list are called *ξ*-variables.

The order of variables in the lists of the LISMOD statement is used to define the variable order in rows and columns of the LISMOD model matrices.
Depending on the model of interest, you might not need to specify all the lists of variables. When some variable lists are not specified, the full model reduces to specialized submodels. However, to be a proper submodel in the LISMOD modeling language, it is necessary (but not sufficient) that at least one of the \texttt{YVAR=} or \texttt{XVAR=} lists is defined. See the section “LISMOD Submodels” on page 1413 for the details about LISMOD submodels that PROC CALIS can handle.

An example of a LISMOD model specification is shown as follows:

```
proc calis;
  lismod xvar=x1-x3 yvar=y1-y6 xivar=xi etavar=eta1-eta2;
  matrix _LAMBDAY_ [,1] = 1. load3 load4,
                  [,2] = 0. 0. 0. 1. load5 load6;
  matrix _THETAY_ [1,1] = ey1-ey3,
                   [2,1] = cey;
  matrix _LAMBDAX_ [,] = 1. load1 load2;
  matrix _THETAX_ [1,1] = 3*ex;
  matrix _GAMMA_ [,1] = beta1 beta2;
  matrix _PHI_ [1,1] = phi;
  matrix _PSI_ [1,1] = psi1-psi2;
run;
```

In this example, you have three x-variables $x_{1-3}$, six y-variables $y_{1-6}$, one $\xi$-variable $x_i$, and two $\eta$-variables $\eta_{1-2}$. The numbers of variables in these lists define the dimensions of the LISMOD model matrices. For example, matrix \texttt{_LAMBDAY_} is $6 \times 2$, with $y_{1-6}$ as the row variables and $\eta_{1-2}$ as the column variables. Matrix \texttt{_THETAY_} is $3 \times 3$, with $x_{1-3}$ as the row and column variables. In the \texttt{MATRIX} statements, you specify parameters in the elements of the matrices. After the matrix name, you specify in square brackets ‘[’ and ‘]’ the starting row and column numbers of the first element to be parameterized. After the equal sign, you specify fixed or free parameters for the matrix elements.

Depending on how you specify the starting row and column numbers, the parameter specification might proceed differently. See the \texttt{MATRIX} statement on page 1295 for a detailed description. In this example, the first specification of the parameters in the \texttt{_LAMBDAY_} matrix starts from $[.,1]$—meaning that it starts from the first column and proceeds downwards. As a result, the $[1,1]$ element is a fixed constant 1.0, the $[2,1]$ element is a free parameter called \texttt{load3}, and the $[3,1]$ element is a free parameter called \texttt{load4}. Similarly, in the second specification in the \texttt{_LAMBDAY_} matrix, the $[1,2]$, $[2,2]$, $[3,2]$, and $[4,2]$ elements take constant values 0, 0, 0, and 1, respectively, and the $[5,2]$ and $[6,2]$ elements are free parameters \texttt{load5} and \texttt{load6}, respectively.

You can also use similar notation to specify the parameters of a row. For example, with the notation $[2,]$ for the starting row and column numbers, specification proceeds to the left with the same second row in the matrix.

If you have specified both starting row and column numbers, such as those in the first specification in matrix \texttt{_THETAY_}, the parameter specification starts from $[1,1]$ and proceeds to the next row and column numbers—that is $[2,2]$, $[3,3]$, and so on. This results in specifying the diagonal elements of matrix \texttt{_THETAY_} as free parameters \texttt{ey1}, \texttt{ey2}, and \texttt{ey3}.

With the notation $[.,]$, no starting row and column numbers are specified. Specification starts from the first valid element in the matrix and proceeds row-wise for all valid elements in the matrix. For example, in the matrix \texttt{_LAMBDAX_} statement, the $[1,1]$ element of matrix \texttt{_LAMBDAX_} is a fixed constant 1, and the $[1,2]$ and $[1,3]$ elements are free parameters \texttt{load1} and \texttt{load2}, respectively.
Default Parameters

It is important to understand the default parameters in the LISMOD model. First, if you know which parameters are default free parameters, you can make your specification more efficient by omitting some specifications. For example, because all variances and covariances among the exogenous \( \xi \)-variables (excluding error terms) are free parameters by default, you do not need to specify them with MATRIX statement if these variances and covariances are not constrained. Second, if you know which parameters are default fixed zero parameters, you can specify your model accurately. For example, because all measurement errors in the LISMOD model are fixed zeros by default, you must use the MATRIX statement to specify the covariances among the errors in the \( \Theta_x \) (_THETAX_) or \( \Theta_y \) (_THETAY_) matrices if you want to fit a model with some correlated measurement errors. See the section “Default Parameters in the LISMOD Model” on page 1416 for details about the default parameters of the LISMOD model.

Modifying a LISMOD Model from a Reference Model

This section assumes that you use a REFMODEL statement within the scope of a MODEL statement and that the reference model (or base model) is also a LISMOD model. The reference model is called the old model, and the model that refers to this old model is called the new model. If the new model is not intended to be an exact copy of the old model, you can use the extended LISMOD modeling language described in this section to make modifications within the scope of the MODEL statement for the new model. The syntax is similar to, but not exactly the same as, the ordinary LISMOD modeling language (see the section “LISMOD Statement” on page 1282). The respecification syntax for a LISMOD model is shown as follows:

```
LISMOD;
  MATRIX matrix-name parameters-in-matrix;
  Repeat the MATRIX statement as needed;
```

First, in the respecification you should not put any variable lists in the LISMOD statement. The reason is that the parameter respecifications in the new model refer to the variable lists of the old model. Therefore, the variable lists in the new model are implicitly assumed to be exactly the same as those in the old model. Because of this, the LISMOD statement is entirely optional for the respecification in the new model.

Second, you can use MATRIX matrix-name statements to modify the old model by using the same syntax as in the LISMOD modeling language. The matrix-name can be one of the twelve possible LISMOD matrices. In addition, in the respecification syntax you can use the missing value ‘.’ to drop a parameter specification from the old model.

The new model is formed by integrating with the old model in the following ways:

- **Duplication:** If you do not specify in the new model a parameter location that exists in the old model, the old parameter specification is duplicated in the new model.

- **Addition:** If you specify in the new model a parameter location that does not exist in the old model, the new parameter specification is used in the new model.

- **Deletion:** If you specify in the new model a parameter location that also exists in the old model and the new parameter is denoted by the missing value ‘.’, the old parameter specification is not copied into the new model.

- **Replacement:** If you specify in the new model a parameter location that also exists in the old model and the new parameter is not denoted by the missing value ‘.’, the new parameter specification replaces the old one in the new model.
For example, the following two-group analysis specifies Model 2 by referring to Model 1 in the `REFMODEL` statement:

```plaintext
proc calis;
  group 1 / data=d1;
  group 2 / data=d2;
  model 1 / group=1;
    lsimod xvar=X1-X3 yvar=Y1-Y6 xivar=xi etavar=eta1-eta2;
    matrix _LAMBDAY_ [1,1] = 1. load3 load4,
                  [2,1] = 0. 0. 0. 1. load5 load6;
    matrix _THETAY_ [1,1] = ey1-ey3,
                   [2,1] = cey;
    matrix _LAMBDAX_ [,1] = 1. load1 load2;
    matrix _THETAX_ [1,1] = 3*ex;
    matrix _GAMMA_ [1,1] = beta1 beta2;
    matrix _PHI_ [1,1] = phi;
    matrix _PSI_ [1,1] = psi1-psi2;
  model 2 / group=2;
    refmodel 1;
    matrix _THETAY_ [2,1] = .;
    matrix _THETAX_ [1,1] = ex1-ex3;
    matrix _BETA_ [2,1] = beta;
run;
```

In this example, Model 2 is the new model which refers to the old model, Model 1. It illustrates the four types of model integration:

- **Duplication**: All parameter locations and specifications in the old model are duplicated in the new model, except for the [2,1] element in matrix `_THETAY_` and the diagonal of matrix `_THETAX_`, which are modified in the new model.

- **Addition**: The `_BETA_[2,1]` parameter location is added with a new parameter `beta` in the new model. This indicates that `eta1` is a predictor variable of `eta2` in the new model, but not in the old model.

- **Deletion**: Because the missing value `.` is used for the parameter value, the `_THETAY_[2,1]` parameter location is no longer defined as a free parameter in the new model. In the old model, the same location is defined by the free parameter `cey`.

- **Replacement**: The diagonal elements of the `_THETAX_` matrix in the new model are now defined by three distinct parameters `ex1–ex3`. This replaces the old specification where a single constrained parameter `ex` is applied to all the diagonal elements in the `_THETAX_` matrix.

---

**LMTESTS Statement**

```plaintext
LMTESTS | LMTEST option < option ... >;
```

where `option` represents one of the following:

- `display-option`
- `test-set`
Chapter 29: The CALIS Procedure

and test-set represents one of the following:

- set-name = [ regions ]
- set-name = { regions }

where set-name is the name of the set of Lagrange multiplier (LM) tests defined by the regions that follow after the equal sign and regions are keywords denoting specific sets of parameters in the model.

You can use the LMTESTS statement to set display-options or to customize the test-sets for the LM tests. The LMTESTS statement is one of the model analysis statements. It can be used within the scope of the CALIS statement so that the options will apply to all models. It can also be used within the scope of each MODEL statement so that the options will apply only locally. Therefore, different models within a CALIS run can have very different LMTESTS options.

The LM Tests Display Options

The following are the display-options for the LM tests:

**DEFAULT**
conducts the default sets of LM tests for freeing fixed parameters in the model. This option is used when you need to reset the default sets of LM tests in the local model. For example, you might have turned off the default LM tests by using the NODEFAULT option in the LMTESTS statement within the scope of PROC CALIS statement. However, for the model under the scope of a particular MODEL statement, you can use this DEFAULT option in the local LMTESTS statement to turn on the default LM tests again.

**MAXRANK**
sets the maximum number of rankings within a set of LM tests. The actual number of test rankings might be smaller because the number of possible LM tests within a set might be smaller than the maximum number requested.

**NODEFAULT**
turns off the default sets of LM tests for freeing fixed parameters in the model. As a result, only the customized LM tests defined in the test-sets of the LMTESTS statement are conducted and displayed. Note that the LM tests for equality and active boundary constraints are not turned off by this option. If you specify this option in the LMTESTS statement within the scope of the PROC CALIS statement, it will propagate to all models.

**NORANK**
turns off the ranking of the LM tests. Ranking of the LM tests is done automatically when the model modification indices are requested. The NORANK option is ignored if you also set the MAXRANK option.

**LMMAT**
prints the sets of LM tests in matrix form, in addition to the normal LM test results.
The Customized Sets of LM Tests: Syntax of the Test-sets

In addition to the display-options, you can define customized sets of LM tests as test-sets in the LMTESTS statement. You can define as many test-sets as you like. Ranking of the LM tests will be done individually for each test-set. For example, the following LMTESTS statement requests that the default sets of LM tests not be conducted by the NODEFAULT option. Instead, two customized test-sets are defined.

```
lmtests nodefault MyFirstSet=[ALL] MySecondSet=[COVEXOG COVERR];
```

The first customized set MyFirstSet pulls all possible parameter locations together for the LM test ranking (ALL keyword). The second customized set MySecondSet pulls only the covariances among exogenous variables (COVEXOG keyword) and among errors (COVERR keyword) together for the LM test ranking.

Two different kinds of regions for LM tests are supported in PROC CALIS: matrix-based or non-matrix-based.

The matrix-based regions can be used if you are familiar with the matrix representations of various types of models. Note that defining test-sets by using matrix-based regions does not mean that LM tests are printed in matrix format. It means only that the parameter locations within the specified matrices are included into the specific test-sets for LM test ranking. For matrix output of LM tests, use the LMMAT option in the LMTESTS statement.

Non-matrix-based regions do not assume the knowledge of the model matrices. They are easier to use in most situations. In addition, non-matrix-based regions can cover special subsets of parameter locations that cannot be defined by model matrices and submatrices. For example, because of the compartmentalization according to independent and dependent variables in the LINEQS model matrices, the sets of LM tests defined by the LINEQS matrix-based regions are limited. For example, you cannot use any matrix-based regions to request LM tests for new paths to existing independent variables in the LINEQS model. Such a matrix does not exist in the original specification. However, you can use the non-matrix based region NEWENDO to refer to these new paths.

The regions for parameter locations are specified by keywords in the LMTESTS statement. Because the regions are specific to the types of models, they are described separately for each model type in the following.

The LM Test Regions for COSAN Models

ALLMAT

specifies all parameter locations in all matrices.

CENTRAL

specifies all parameter locations in the central covariance matrices in all terms.

MATRIX= [set-of-matrices] | {set-of-matrices}

MAT= [set-of-matrices] | {set-of-matrices}

MATSET= [set-of-matrices] | {set-of-matrices}

specifies the parameter locations in the matrices specified in set-of-matrices.

MEANVEC

specifies all parameter locations in the central mean vectors in all terms.

OUTER

specifies all parameter locations in all matrices except for the central covariance matrices and central mean vectors in all terms.
The LM Test Regions for FACTOR Models

The keywords for the matrix-based regions are associated with the FACTOR model matrices. See the section “Summary of Matrices in the FACTOR Model” on page 1399 for the definitions and properties of these matrices.

**Keywords for Matrix-Based Regions**

_FACTERRV_ | FACTERRV
 specifies the error variances.

_FACTFCOV_ | FACTFCOV
 specifies the covariances among factors.

_FACTINTE_ | FACTINTE
 specifies the intercepts.

_FACTLOAD_ | FACTLOAD
 specifies the factor loadings.

_FACTMEAN_ | FACTMEAN
 specifies the factor means.

**Keywords for Non-Matrix-Based Regions**

ALL
 specifies all parameter locations.

COV
 specifies the covariances among factors.

COVERR
 specifies the covariances among errors.

COVF | COVLV
 specifies the covariances among factors.

FIRSTMOMENTS
 specifies the means of factors and the intercepts.

INTERCEPTS
 specifies the intercepts.

LOADINGS
 specifies the factor loadings.

MEANS | MEAN
 specifies the means of factors.
The LM Test Regions for LINEQS Models

**Keywords for Matrix-Based Regions**
The *keywords* for the matrix-based regions are associated with the LINEQS model matrices. See the section “Matrix Representation of the LINEQS Model” on page 1403 for definitions of these model matrices and see the section “Summary of Matrices and Submatrices in the LINEQS Model” on page 1405 for the names and properties and the model matrices and submatrices.

_EQSALPHA_ | EQSALPHA
specifies the intercepts of dependent variables.

_EQSBETA_ | EQSBETA
specifies effects of dependent variables on dependent variables.

_EQSGAMMA_ | _EQSGAMMA_SUB_ | EQSGAMMA | EQSGAMMASUB
specifies the effects of independent variables (excluding errors) on dependent variables. Because effects of errors on dependent variables are restricted to ones in the LINEQS model, LM tests on _EQSGAMMA_ and _EQSGAMMA_SUB_ (submatrix of _EQSGAMMA_) are the same.

_EQSNU_ | _EQSNU_SUB_ | EQSNU | EQSNUSUB
specifies the means of independent variables (excluding errors). Because means of errors are restricted to zero in the LINEQS model, LM tests on _EQSNU_ and _EQSNU_SUB_ (submatrix of _EQSNU_) are the same.

_EQSPHI_ | EQSPHI
specifies variances and covariances among all independent variables, including errors.

_EQSPHI11_ | EQSPHI11
specifies variances and covariances among independent variables, excluding errors.

_EQSPHI21_ | EQSPHI21
specifies covariances between errors and disturbances with other independent variables.

_EQSPHI22_ | EQSPHI22
specifies variances and covariances among errors and disturbances.

**Keywords for Non-Matrix-Based Regions**

ALL
specifies all possible parameter locations.

COV
specifies all covariances among independent variables, including errors and disturbances.

COVERR
specifies covariances among errors or disturbances.

COVEXOG
specifies covariances among independent variables, excluding errors and disturbances.
**Chapter 29: The CALIS Procedure**

**COVEXOGERR**
- Specifies covariances of errors and disturbances with other independent variables.

**COVLV | COVFECT**
- Specifies covariances among latent variables (excluding errors and disturbances).

**COVMV | COVOV**
- Specifies covariance among independent manifest variables.

**EQUATION | EQUATIONS**
- Specifies all possible linear relationships among variables.

**FIRSTMOMENTS**
- Specifies means and intercepts.

**INTERCEPTS | INTERCEPT**
- Specifies intercepts of dependent variables.

**LV===>LV | LV-->LV**
- Specifies all possible effects of latent factors on latent factors.

**LV===>MV | LV-->MV | MV<=LV | MV<-LV**
- Specifies all possible effects of latent factors on manifest variables.

**LV<=MV | LV<--MV | MV==>LV | MV<--LV**
- Specifies all possible effects of manifest variables on latent factors.

**MEANS | MEAN**
- Specifies the means of independent factors.

**MV===>MV | MV-->MV**
- Specifies all possible effects of manifest variables on manifest variables.

**NEWDEP | NEWENDO**
- Specifies effects of other variables on the independent variables in the original model.

**PATHS | PATH**
- Specifies all possible linear relationships among variables.

**The LM Test Regions for LISMOD Models**

The *keywords* for the matrix-based regions are associated with the LISMOD model matrices. See the section “Model Matrices in the LISMOD Model” on page 1410 for the definitions and properties of these matrices.

**Keywords for Matrix-Based Regions**

**_ALPHA_ | ALPHA**
- Specifies the _ALPHA_ matrix.

**_BETA_ | BETA**
- Specifies the _BETA_ matrix.
_GAMMA_ | GAMMA
    specifies the _GAMMA_ matrix.

_KAPPA_ | KAPPA
    specifies the _KAPPA_ matrix.

_LAMBDA_ | LAMBDA
    specifies the _LAMBDAX_ and _LAMBDAY_ matrices.

_LAMBDAX_ | LAMBDAX
    specifies the _LAMBDAX_ matrix.

_LAMBDAY_ | LAMBDAY
    specifies the _LAMBDAY_ matrix.

_NU_ | NU
    specifies the _NUX_ and _NUY_ matrices.

_NUX_ | NUX
    specifies the _NUX_ matrix.

_NUY_ | NUY
    specifies the _NUY_ matrix.

_PHI_ | PHI
    specifies the _PHI_ matrix.

_PSI_ | PSI
    specifies the _PSI_ matrix.

_THETA_ | THETA
    specifies the _THETAX_ and _THETAY_ matrices.

_THETAX_ | THETAX
    specifies the _THETAX_ matrix.

_THETAY_ | THETAY
    specifies the _THETAY_ matrix.

*Keywords for Non-Matrix-Based Regions*

ALL
    specifies all model matrices.

COV
    specifies all covariance parameters in _THETAY_, _THETAX_, _PHI_, and _PSI_.

COVERR
    specifies all covariances for errors or disturbances in _THETAY_, _THETAX_, and _PSI_.

The LM Test Regions for MSTRUCT Models

The **keywords** for the matrix-based regions are associated with the MSTRUCT model matrices. See the section “Model Matrices in the MSTRUCT Model” on page 1417 for the definitions and properties of these matrices.

**Keywords for Matrix-Based Regions**

- `_MSTRUCTCOV_ | _COV_ | MSTRUCTCOV`
  
  specifies the `_MSTRUCTCOV_` or `_COV_` matrix.

- `_MSTRUCTMEAN_ | _MEAN_ | MSTRUCTMEAN`
  
  specifies the `_MSTRUCTMEAN_` or `_MEAN_` vector.
Keywords for Non-Matrix-Based Regions

ALL
specifies the _MSTRUCTCOV_ (or _COV_) and _MSTRUCTMEAN_ (or _MEAN_) matrices.

COV
specifies the _MSTRUCTCOV_ or _COV_ matrix.

MEANS | MEAN
specifies the _MSTRUCTMEAN_ or _MEAN_ matrix.

The LM Test Regions for PATH and RAM Models

The keywords for the matrix-based regions are associated with the submatrices of the RAM model matrices. See the section “Partitions of the RAM Model Matrices and Some Restrictions” on page 1427 for the definitions of these submatrices and the section “Summary of Matrices and Submatrices in the RAM Model” on page 1428 for the summary of the names and properties of these submatrices.

Keywords for Matrix-Based Regions

_RAM_ | _A_ | RAMA
specifies the _RAMA_ matrix.

_RAMALPHA_ | RAMALPHA
specifies the _RAMALPHA_ matrix.

_RAMBETA_ | RAMBETA
specifies the _RAMBETA_ matrix.

_RAMGAMMA_ | RAMGAMMA
specifies the _RAMGAMMA_ matrix.
Chapter 29: The CALIS Procedure

_RAMNU_ | RAMNU
  specifies the _RAMNU_ matrix.

_RAMP_ | _P_ | RAMP
  specifies the _RAMP_ matrix.

_RAMP11_ | RAMP11
  specifies the _RAMP11_ matrix.

_RAMP21_ | RAMP21
  specifies the _RAMP21_ matrix.

_RAMP22_ | RAMP22
  specifies the _RAMP22_ matrix.

_RAMW_ | _W_ | RAMW
  specifies the _RAMW_ vector.

**Keywords for Non-Matrix-Based Regions**

ALL
  specifies all possible parameter locations.

ARROWS | ARROW
  specifies all possible paths (that is, the entries in the _RAMA_ matrix).

COV
  specifies all covariances and partial covariances (that is, the entries in the _RAMP_ matrix).

COVERR
  specifies partial covariances among endogenous variables (that is, the entries in the _RAMP11_ matrix).

COVEXOG
  specifies covariances among exogenous variables (that is, the entries in the _RAMP22_ matrix).

COVEXOGERR
  specifies partial covariances of endogenous variables with exogenous variables (that is, the entries in the _RAMP21_ matrix).

COVLV | COVFACT
  specifies covariance among latent factors (that is, entries in _RAMP11_ pertaining to latent variables).

COVMV | COVOV
  specifies covariance among manifest variables (that is, entries in _RAMP11_ pertaining to manifest variables).

FIRSTMOMENTS
  specifies means or intercepts (that is, entries in _RAMW_ vector).

INTERCEPTS | INTERCEPT
  specifies intercepts for endogenous variables (that is, entries in _RAMALPHA_ vector).
LV==>LV | LV->LV
specifies effects of latent variables on latent variables.

LV==>MV | LV->MV | MV<==LV | MV<–LV
specifies effects of latent variables on manifest variables.

LV<==MV | LV<–MV | MV==>LV | MV–>LV
specifies effects of manifest variables on latent variables.

MEANS | MEAN
specifies the means of exogenous variables (that is, entries in the _RAMNU_ vector).

MV==>MV | MV–>MV
specifies effects of manifest variables on manifest variables.

NEWENDO
specifies new paths to the exogenous variables in the original model.

PATHS | PATH
specifies all possible paths (that is, the entries in the _RAMA_ matrix).

---

**MATRIX Statement**

```
MATRIX matrix-name < location < = parameter-spec > , location < = parameter-spec . . . >> ;
```

MATRIX statement specifies the matrix elements (locations) and their parameters. Parameters can be fixed or free, with or without initial estimates. The `matrix-name` indicates the matrix to specify in the MATRIX statement. The `location` indicates the starting row and column numbers of the matrix being specified and the `parameter-spec` is a list of free or fixed parameters for the elements that are indicated by the `location`.

The MATRIX statement is a subsidiary model specification statement of the COSAN, LISMOD, and MSTRUCT modeling languages. You might need to use the MATRIX statements as many times as needed for specifying your model. However, you can use the MATRIX statement at most once for each distinct model matrix.

**Valid Matrix Names for the COSAN Model**

The valid `matrix-names` depend on the your specification in the COSAN statement in which you define the COSAN model matrices and their properties. Except for those fixed matrices with the IDE or ZID type, you can use the MATRIX statement to specify any COSAN model matrices you define in the COSAN statement.

**Valid Matrix Names for the LISMOD Model**

There are 12 model matrices in the LISMOD model, and they correspond to the following valid `matrix-names`:

- matrices and their types in the measurement model for the y-variables
  
  _LAMBDA_y_ the matrix of regression coefficients of the y-variables on the \( \eta \)-variables (general, GEN)
Chapter 29: The CALIS Procedure

• matrices and their types in the measurement model for the \( x \)-variables
  \[ \_LAMBDAX_ \] the matrix of regression coefficients of the \( x \)-variables on the \( \xi \)-variables (general, GEN)
  \[ \_NUX_ \] the vector of intercept terms of the \( x \)-variables (general, GEN)
  \[ \_THETAX_ \] the error covariance matrix for the \( x \)-variables (symmetric, SYM)

• matrices and their types in the structural model
  \[ \_ALPHA_ \] the vector of intercept terms of the \( \eta \)-variables (general, GEN)
  \[ \_BETA_ \] the matrix of regression coefficients of the \( \eta \)-variables on the \( \eta \)-variables (general, GEN)
  \[ \_GAMMA_ \] the matrix of regression coefficients of the \( \eta \)-variables on the \( \xi \)-variables (general, GEN)
  \[ \_KAPPA_ \] the mean vector for the \( \xi \)-variables (general, GEN)
  \[ \_PHI_ \] the covariance matrix for the \( \xi \)-variables (symmetric, SYM)
  \[ \_PSI_ \] the error covariance matrix for the \( \xi \)-variables (symmetric, SYM)

Valid Matrix Names for the MSTRUCT Modeling Language

The following matrix-names are valid for the MSTRUCT modeling language:

- \[ \_COV_ \] the covariance matrix (symmetric, SYM)
- \[ \_MEAN_ \] the mean vector (general, GEN)

Specifying Locations in Model Matrices

The five main types of matrix locations (elements) specification in the MATRIX statement are briefly described in the following:

- **Unspecified location**: Blank or \([ , ]\)
  Use this notation to specify the \([1,1]\) element of the matrix, and to specify the remaining valid elements of the matrix in a prescribed order until all the parameters in the parameter-spec list are assigned.

- **Row-and-column location**: \([i,j]\), \([@i,j]\), \([i, @j]\), or \([@i, @j]\)
  Use this notation to specify the \([i,j]\) element of a matrix, and to specify the remaining elements of the matrix in the order indicated by the location notation until all the parameters in the parameter-spec list are assigned.
• Row location only: \([i, i_1, j, j_1, \) or \([iset, ]\)
  Use this notation to specify the first valid matrix element in the \([i]\)-th row (for the first two notations)
  or the \([i_1]\)-th row (for the \([iset, ]\) notation, where \(iset=(i_1, i_2, \ldots)\) is a set of row numbers),
  and to specify the remaining elements of the matrix in the order indicated by the location notation until all
  the parameters in the parameter-spec list are assigned.

• Column location only: \([i, j, \) or \([, jset]\)
  Use this notation to specify the first valid matrix element in the \([j]\)-th column (for the first two
  notations) or the \([j_1]\)-th column (for the \([, jset]\) notation, where \(jset=(j_1, j_2, \ldots)\) is a set
  of columns), and to specify the remaining elements of the matrix in the order indicated by the location
  notation until all the parameters in the parameter-spec list are assigned.

• Row-and-column-sets location: \([iset, jset, ]\), \([iset, j, ]\), or \([i, jset]\)
  Use this notation to specify the \([i_1, j_1]\) element of the matrix, where \(i_1\) is either the same as \(i\)
  or the first row number specified in \(iset\), and \(j_1\) is either the same as \(j\) or the first column number
  specified in \(jset\), and to specify the remaining elements of the matrix in the order indicated by the location
  notation until all the parameters in the parameter-spec list are assigned.

Consider the following points about the location specifications:

• In the description of the various location specifications, the starting matrix element for parameter
  assignment is relatively well-defined. However, if the parameter-spec list has more than one parameter,
  there are more matrix elements to assign with the parameters in the parameter-spec list. If there is
  no parameter-spec list, a set of matrix elements are specified as unnamed free parameters. Hence,
  the actual number of elements specified by these location specifications depends on the length of the
  parameter-spec list.

• Because more than one matrix element could be specified in any of these location specifications, it is
  important to understand the order that PROC CALIS uses to assign the matrix elements.

• In some of the location specifications, either the row or column is unspecified and the assignment of
  the matrix element starts with the first valid element given the column or the row number. This first
  valid element depends on the type of the matrix in question.

The next few sections describe the parameter assignments in more detail for each of these location
specifications in the MATRIX statement.

Unspecified Location: Blank or \([, ]\)

This notation means that all valid elements started with the \([1, 1]\) element of the matrix specified in the
model. If no parameter-spec list is specified, all valid elements in the matrix are unnamed free parameters.
For these elements, PROC CALIS generates parameter names with the _Parm prefix followed by a unique
integer (for example, _Parm1, _Parm2, and so on). If a parameter-spec list is specified, the assignment
of parameters starts with the \([1, 1]\) element and proceeds to the next valid elements in the same row.
If the entire row of valid elements is assigned with parameters, it proceeds to the next row and so on, until all
the parameters in the parameter-spec list are assigned. The valid element given the row or column number
depends on the type of matrix in question. The following examples illustrate the usage of the unspecified
location notation.

Suppose that _GAMMA_ is a general \(3 \times 3\) matrix. The following statement specifies four elements of this
matrix:
matrix _GAMMA_ [,] = gg1-gg4;

Equivalently, you can use the following blank location specification:

matrix _GAMMA_ = gg1-gg4;

Both specifications are equivalent to the following elementwise specification:

\[
\begin{align*}
\text{matrix } & \text{ _GAMMA_ } [1,1] = gg1, \\
& [1,2] = gg2, \\
& [1,3] = gg3, \\
& [2,1] = gg4;
\end{align*}
\]

With the unspecified location for the matrix _GAMMA_, the first row is filled up with the parameters first. Then it proceeds to the next row and so on until all parameters in the parameter-spec list are assigned. Because there are four parameters and _GAMMA_ has three columns, the parameter gg4 is assigned to the _GAMMA_ [2, 1] element.

However, if the preceding specification is for a 3 × 3 matrix symmetric matrix _PHI_, the parameters are assigned differently. That is, the following specification has different matrix elements assigned with the parameters:

matrix _PHI_ = gg1-gg4;

Because symmetric matrices contain redundant elements, parameters are assigned only to the lower triangular elements (including the diagonal elements). As a result, the following elementwise specification reflects the preceding specification of matrix _PHI_:

\[
\begin{align*}
\text{matrix } & \text{ _PHI_ } [1,1] = gg1, \\
& [2,1] = gg2, \\
& [2,2] = gg3, \\
& [3,1] = gg4;
\end{align*}
\]

The case for lower triangular matrices is the same as the case for symmetric matrices. That is, only the lower triangular elements are valid elements for the parameter assignments.

For upper triangular matrices, only the upper triangular elements (including the diagonal elements) are valid for the parameter assignments. For example, consider the following specification of a 3 × 3 upper triangular matrix UPP:

matrix UPP = gg1-gg4;

The matrix elements assigned with the parameters are the same as the following elementwise specification:

\[
\begin{align*}
\text{matrix } & \text{ UPP } [1,1] = gg1, \\
& [1,2] = gg2, \\
& [1,3] = gg3, \\
& [2,2] = gg4;
\end{align*}
\]

If a 4 × 4 diagonal matrix is specified by the preceding MATRIX statement, the parameters are assigned to the following elements: [1, 1], [2, 2], [3, 3], and [4, 4].

Lastly, if there is no parameter-spec list for the unspecified location notation, all valid parameters in the matrix being specified are unnamed free parameters. For example, if A is a 4 × 4 general rectangular matrix, the following specification assigns 16 unnamed free parameters to all of the elements in A:
PROC CALIS generates parameters _Parm1, _Parm2, . . . , _Parm16 to the elements [1,1], [1,2], [1,3], . . . , [4,3], [4,4], respectively.

However, if S is a 4 × 4 symmetric matrix, the following specification assigns only 10 unnamed free parameters to the lower triangular elements of S:

```
matrix S;
```

PROC CALIS generates parameters _Parm1, _Parm2, . . . , _Parm10 to the elements [1,1], [2,1], [2,2], . . . , [4,3], [4,4], respectively.

**Row-and-Column Location: [i, j], [@i, j], [i, @j], or [@i, @j]**

All these notations provide the starting row (i) and column (j) numbers for the assignment of the parameters in the `parameter-spec` list. The notations are different in the way they proceed to the next element in the matrix. If no `parameter-spec` list is specified, only the single element [i, j] is an unnamed free parameter. For this [i, j] element, PROC CALIS generates a parameter name with the _Parm prefix followed by a unique integer (for example, _Parm1). If a `parameter-spec` list is specified, the assignment of parameters starts with the [i, j] element and proceeds to next element until all the parameters in the `parameter-spec` list are assigned. The following summarizes how the assignment of parameter proceeds, depending on the uses of the @ sign before the starting row or column number:

- [i, j] specifies the [i, j] element, and proceeds to [i+1, j+1], [i+2, j+2], and so on.
- [@i, j] specifies the [i, j] element, and proceeds to [i, j+1], [i, j+2], [i, j+3], and so on.
- [i, @j] specifies the [i, j] element, and proceeds to [i+1, j], [i+2, j], [i+3, j], and so on.
- [@i, @j] specifies the [i, j] element only.

The following examples illustrates the usage of the row-and-column location notation.

The simplest case is the specification of a single element as an unnamed free parameter. For example, the following statement specifies that [1, 4] in matrix A is an unnamed free parameter:

```
matrix A [1,4];
```

PROC CALIS generates a parameter name with the _Parm prefix for this element. In this case, using the @ sign before the row or column number is optional. That is, the following statements are all the same specification:

```
matrix A [1,4];
matrix A [@1,4];
matrix A [1,@4];
matrix A [@1,@4];
```

You can specify more than one unnamed free parameter by using multiple location specifications, as shown in the following example:
matrix A [1,4],[3,5];

Elements [1,4] and [3,5] of matrix A are both unnamed free parameters. However, when a parameter-spec list is specified after the location, more than one parameters might be specified. The use of the @ determines how the elements in the matrix are assigned with the parameters in the parameter-spec list. The following examples illustrate this under various situations.

For example, consider the following specification of a 4 x 4 matrix general matrix A:

```
matrix A
[1,1] = a b c;
```

The three parameters a, b, and c, are assigned to the matrix elements [1,1], [2,2], and [3,3], respectively. That is, this specification is equivalent to the following elementwise specification:

```
matrix A
[1,1] = a ,
[2,2] = b ,
[3,3] = c ;
```

However, with the @ sign, the assignment is different. For example, consider the @ sign attached to the row number in the following specification:

```
matrix A
[@1,1] = a b c;
```

The @ sign fixes the row number to 1. As a result, this specification is equivalent to the following elementwise specification:

```
matrix A
[1,1] = a ,
[1,2] = b ,
[1,3] = c ;
```

Using the @ sign before the column number fixes the column number. For example, consider the following specification of matrix A:

```
matrix A
[2,@2] = a b c;
```

The @ sign fixes the column number to 2. As a result, this specification is equivalent to the following elementwise specification:

```
matrix A
[2,2] = a ,
[3,2] = b ,
[4,2] = c ;
```

If you put the @ sign in both of the row and column numbers, only one element is intended to be assigned. For example, the following specification means that only A[2,3] is assigned with the parameter a:

```
matrix A
[@2,@3] = a;
```

But you could specify this simply as the statement without the @ sign:
Notice that the matrix type does not play a role in determining the elements for the parameter assignments in the row-and-column location specification. You have to make sure that the parameters are assigned in the valid elements of the matrix. For example, suppose that \( S \) is a \( 4 \times 4 \) symmetric matrix and you specify the following statement for its elements:

\[
\text{matrix } A \\
[0,2] = \begin{bmatrix} a & b & c \end{bmatrix};
\]

The elements to be assigned with the parameters \( a, b, \) and \( c \), are \([3,2], [3,3], \) and \([3,4]\), respectively. However, because \( S \) is symmetric, you can specify only the nonredundant elements in the lower triangular of \( S \). Hence, the specification of the \([3,4]\) element is not valid and it generates an error.

Row Location Only: \([i, \], [@i, \] or \([iset, \]

All these notations provide the starting row \([i1, \] for the assignment of the parameters in parameter-spec, where \( i1 \) is \( i \) for the first two location notations or \( i1 \) is the first row specified in iset, where \( iset = (i1, i2, \ldots) \) is a set of row numbers. Because no column location is specified, the starting element is the first valid element in the \( i1 \)-th row of the matrix.

If no parameter-spec list is specified, all the valid elements in the entire \( i1 \)-th row of the matrix are unnamed free parameters. If a set of row numbers is specified in iset, all the valid elements in the all the rows specified in iset are unnamed free parameters.

If a parameter-spec list is specified, the assignment of parameters starts with the first valid elements of the \( i1 \)-th row. The assignment proceeds to next valid elements in the same row. The \([i, \] specification proceeds row by row for parameter assignment while the \([@i, \] specification stays at the same \( i \)-th row. The \([iset, \] specification indicates and limits the sequence of rows to be assigned with the parameter in the parameter-spec list. The assignment stops when all the parameters in the parameter-spec list are assigned.

The following summarizes how the assignment of parameters proceeds in more precise terms:

- \([i, \] specifies the first valid element in row \( i \) and proceeds to the valid elements in rows \( i, i+1, i+2, \ldots \), until all parameters in the parameter-spec list are assigned.
- \([@i, \] specifies the first valid elements in row \( i \) and proceeds to the valid elements in the same row until all parameters in the parameter-spec list are assigned.
- \([iset, \] specifies the first valid elements in row \( i1 \), where \( i1, i2, \ldots \) are the rows specified in iset. It proceeds to the valid elements in rows \( i1, i2, \ldots \), until all parameters in the parameter-spec list are assigned.

The following examples illustrates the usage of the row locations.

The simplest case is the specification of all valid elements of a single row as unnamed free parameters. For example, the following specification of a \( 3 \times 3 \) rectangular matrix \( A \) assigns unnamed free parameters to all the elements in the second row of matrix \( A \):
matrix A [2,];

PROC CALIS generates parameter names with the _Parm prefix for these elements. For example, the [2,1], [2,2], and [2,3] elements are named with _Parm1, _Parm2, and _Parm3, respectively.

Using the @ sign before the row number in this case is optional. That is, the following statement is the same specification:

matrix A [@2,];

If you specify a set of row numbers without the parameter-spec list, all valid elements of the specified rows are unnamed free parameters. For example, consider the following specification of a 6 × 6 symmetric matrix S:

matrix S [1 3 5,];

This specification specifies unnamed free parameters for the lower triangular elements in the first, third, and fifth rows of matrix S. It is equivalent to the following specification:

matrix S [1,],
[3,],
[5,];

As a result, this means that the following elements in matrix S are free parameters: [1,1], [3,1], [3,2], [3,3], [5,1], [5,2], [5,3], [5,4], and [5,5]. Notice that only the elements in the lower triangular of those specified rows in S are free parameters. This shows that parameter assignment with the row location notation depends on the matrix type.

With the use of the parameter-spec list, the parameter assignment with the row location notation stops when all the parameters are assigned. For example, consider the following specification of a 4 × 4 general (rectangular) matrix A:

matrix A [2,] = a b c;

The three parameters a, b, and c, are assigned to the matrix elements [2,1], [2,2], and [2,3], respectively. However, a different assignment of the parameters applies if you use the same specification for a 4 × 4 symmetric matrix S, as shown in the following statement:

matrix S [2,] = a b c;

Because there are redundant elements in a symmetric matrix, you can specify only the lower triangular elements. Therefore, the row location specification is equivalent to the following elementwise specification:

matrix S
[2,1] = a,
[2,2] = b,
[3,1] = c;

When all the valid row elements are assigned with the parameters, the assignment proceeds to the next row. This is why the last parameter assignment is for S[3, 1]. The same assignment sequence applies to matrices with the lower triangular type (LOW).

For matrices with the upper triangular matrix type (UPP), only the elements in the upper triangular are assigned. For example, consider a 4 × 4 upper triangular matrix U with the following row location specification:
**MATRIX Statement**

The assignment of parameters is the same as the following elementwise specification:

```
matrix U
    [2,] = a b c d;
```

The first valid element in the second row of the U matrix is U[2, 2]. Because all the valid elements in the second row are assigned with parameters, the last element has to go to the valid element in the next row. Hence, the parameter d is assigned to U[3, 3].

For matrices with the diagonal matrix type (DIA), only the diagonal elements are assigned. For example, consider a 4 × 4 upper diagonal matrix D with the following row location specification:

```
matrix D
    [2,] = a b c;
```

The assignment of parameters is the same as the following elementwise specification:

```
matrix D
    [2,2] = a ,
    [2,3] = b ,
    [2,4] = c ,
    [3,3] = d ;
```

If you use an @ sign before the row number in the row location specification, the row number cannot move—it cannot proceed to the next row even if the valid elements in that row are already filled with the parameters in parameter-spec. All other behavior of the [@i, ] specification is the same as that of the [i, ] specification. For example, consider the following specification of a 4 × 4 general (rectangular) matrix A:

```
matrix A
    [@2,] = a b c d;
```

The four parameters a, b, c, and d, are assigned to the matrix elements [2, 1], [2, 2], [2, 3], and [2, 4], respectively. This is exactly the same result as the following specification without the @ sign:

```
matrix A
    [2,] = a b c d;
```

Here, all the elements of the second row of matrix A are assigned with elements. However, if one more parameter is specified in the parameter-spec list, the behavior for the two types of row location specifications are different. The following specification without the @ sign proceeds to the next row for the last parameter:

```
matrix A
    [2,] = a b c d e;
```

That is, the parameter e is assigned to the A[3, 1] element. However, the following specification with the @ sign results in an out-of-bound error:
matrix A
[@2,] = a b c d e;

The out-of-bound error is due to the fact that the row number must be fixed so that the parameter e is forced
to be assigned to A[2, 5], which does not exist.

However, the distinction between the row location specifications with and without the @ sign is not very
important in common practice because in most cases you do not want the parameter assignment to proceed
row after row automatically with a long list of parameters. For example, consider the following specification
of a 4 × 4 symmetric matrix S:

matrix S
 [2,] = s21 s22 s31 s32 s33 s41 s42 s43;

This specification is equivalent to the following specification:

matrix S
 [2,] = s21 s22,
 [3,] = s31 s32 s33,
 [4,] = s41 s42 s43;

Although this specification is not as concise as the preceding one, it specifies more clearly about how
parameters are assigned to each of the three rows of the S matrix. In this specification, you make sure that
each of the three row location specifications has just enough parameters for the given row without proceeding
to the next row for additional parameter assignments. With this kind of “careful” row location specifications,
you do not need to use the @ sign before the row numbers at all.

The last type of row location specification is the {iset, } notation, where iset means a set of row numbers. This
specification type provides the set of row numbers for the assignment of the parameters in the parameter-spec
list. For example, consider the following specification of a 4 × 4 general matrix A:

matrix A
 [2 4,] = a21 a22 a23 a24 a41 a42 a43 a44;

This specification is equivalent to the following statement with two row location specifications:

matrix A
 [2,] = a21 a22 a23 a24,
 [4,] = a41 a42 a43 a44;

In other words, the assignment of parameters follows the order of rows provided in the iset. Notice that the
iset notation merely provides the order of rows to be assigned with the parameters in the parameter-spec list;
it is not an error if you provide a shorter parameter list than that of the total number of elements in the rows.
For example, the following specification of a 4 × 4 general matrix A is valid:

matrix A
 [2 4,] = a21 a22 a23 a24;

This specification has the same results as the following statement with one row location:

matrix A
 [2,] = a21 a22 a23 a24;

However, a valid specification does not mean it is a good representation of the problem. Providing more rows
in the iset specification than intended is simply not a good practice.
Although a shorter `parameter-spec` list is acceptable, a longer list results in an error. For example, the following specification of a 4 × 4 symmetric matrix \( S \) results in an error:

\[
\text{matrix } S \\
[2 \text{ to } 3, \] = s21 s22 s31 s32 s33 extra1 extra2;
\]

The [2 to 3,] not only gives the sequence of the rows for the parameter assignment, it also limits the set of rows to assign. Because matrix \( S \) is symmetric and because only the second and the third rows are supposed to be assigned with the `iset` specification, the parameters `extra1` and `extra2` are excessive.

**Column Location Only: [,j], [,@j], or [,jset]**

These notations mirror that of the row `location` notations. Instead of the rows being specified, the columns are specified by these notations. Therefore, you can understand the column `location` notations the same way as the row `location` notations.

All these column `location` notations provide the starting column [, j1] for the assignment of the parameters in `parameter-spec`, where j1 is j for the first two `location` notations or j1 is the first column specified in `jset`, where `jset` = (j1, j2, ...) is a set of column numbers. Because no row location is specified, the starting element is the first valid element in the j1-th column of the matrix.

If no `parameter-spec` list is specified, all the valid elements in the entire j1-th column of the matrix are unnamed free parameters. If a set of column numbers is specified in `jset`, all the valid elements in the all the columns specified in `jset` are unnamed free parameters.

If a `parameter-spec` list is specified, the assignment of parameters starts with the first valid elements of the j1-th column. The assignment proceeds to next valid elements in the same column. The [, j] specification proceeds column by column for parameter assignment while the [,@j] specification stays at the same j-th column. The [,jset] specification indicates and limits the sequence of columns to be assigned with the parameter in the `parameter-spec` list. The assignment stops when all the parameters in the `parameter-spec` list are assigned. The following list summarizes how the assignment of parameters proceeds:

- [, j1] specifies the first valid element in column j and proceeds to the valid elements in columns j, j+1, j+2, ..., until all parameters in the `parameter-spec` list are assigned.
- [, @j] specifies the first valid elements in column j and proceeds to the valid elements in the same column until all parameters in the `parameter-spec` list are assigned.
- [,jset] specifies the first valid elements in column j1, where j1, j2, ..., are the columns specified in `jset`. It proceeds to the valid elements in columns j1, j2, ..., until all parameters in the `parameter-spec` list are assigned.

See the section “Row Location Only: [i,], [@i,], or [iset,]” on page 1301 for examples, which are applicable to the usage of the column `locations`.

**Row-and-Column-Sets Location: [iset,jset], [iset,j], or [i,jset]**

These notations specify the sets of row and column elements for the assignment of the parameters in the `parameter-spec` list. In the first notation, you specify the set of row numbers in `iset` = (i1, i2, ...), and the set of column numbers in `jset` = (j1, j2, ...). The last two notations are special cases of the first notation. The [iset,j] notation specifies only one column with `jset` = j1 = j. The [i,jset] notation specifies only one row with `iset` = i1 = i. For the last two notations, adding the @ sign before j or i is optional. In general, the row-and-column-sets `locations` specify the matrix elements in the following order:
where \( r \) represents the number of rows in the iset and \( s \) represents the number of columns in the jset. Note that this ordering of elements does not necessarily mean that all these elements are specified. The number of elements specified depends on the length of the parameter-spec list.

If no parameter-spec list is specified after the location notation, all the \( r \times s \) elements specified in the iset and jset are unnamed free parameters. PROC CALIS generates parameter names with the _Parm prefix for these elements.

If a parameter-spec list is specified after the location notation, the total number of matrix elements that are assigned with the parameters is the same as the number of parameter specifications in the parameter-spec list.

The following examples illustrates the usage of the row-and-column-sets locations.

The simplest case is the specification of all elements in the iset and jset as free parameters, as shown in the following statement:

\[
\text{matrix _Gamma_ \[2 3,4 1\];}
\]

This means that _Gamma_[2, 4], _Gamma_[2, 1], _Gamma_[3, 4], and _Gamma_[3, 1] are all free parameters in the matrix. For these elements, PROC CALIS generates parameter names with the _Parm prefix followed by a unique integer (for example, _Parm1, _Parm2, ...). This row-and-column-sets location specification is the same as the following specification:

\[
\text{matrix _Gamma_ \[2,4 1\],\[3,4 1\];}
\]

It is also equivalent to the following elementwise specification:

\[
\text{matrix _Gamma_ \[2,4\],\[2,1\],\[3,4\],\[3,1\];}
\]

If you provide a parameter-spec list after the row-and-column-sets location, the parameters in the list are assigned to the matrix elements. For example, consider the following specification:

\[
\text{matrix _Gamma_ \[2 3,4 1\] = gamma1-gamma4;}
\]

This specification is equivalent to the following elementwise specification:

\[
\text{matrix _Gamma_ \[2,4\] = gamma1,}
\]
\[
[2,1] = gamma2,
\]
\[
[3,4] = gamma3,
\]
\[
[3,1] = gamma4;
\]

It is not necessary for all the elements specified in the row-and-column-sets location to be assigned with the parameters in the parameter-spec list. For example, the following iset and jset specify a maximum of six elements, but only five parameters are assigned as a result of a shorter parameter-spec list:
matrix _Gamma_ [2 to 4,1 5] = gamma1-gamma4;

This specification is equivalent to the following elementwise specification:

```
matrix _Gamma_ [2,1] = gamma1,
    [2,5] = gamma2,
    [3,1] = gamma3,
    [3,5] = gamma4,
    [4,1] = gamma5;
```

In this case, _Gamma_[4, 5] is not specified and is fixed at zero by default.

With the row-and-column-sets location specifications, you need to be aware of the matrix type being specified. For example, the following specification of the symmetric matrix S results in an out-of-bounds error:

```
matrix S [1 2,1 2] = s1-s4;
```

This specification is equivalent to the following elementwise specification:

```
matrix S [1,1] = s1,
    [1,2] = s2,
    [2,1] = s3,
    [2,2] = s4;
```

The specification of the S[1, 2] element is not valid because you can specify only the lower triangular elements of a symmetric matrix in PROC CALIS. The upper triangular elements are redundant and are taken into account by PROC CALIS during computations.

### Specifying Fixed and Free Parameters in Model Matrices

For clarity in describing various location notations, the parameter-spec list contains only free parameters in the examples. In general, you can specify fixed values, free parameters, and initial values in the parameter-spec list. The syntax for the parameter-spec list is the same as the parameter-spec list for the VARIANCE statement. You can specify the following five types of the parameters in the MATRIX statement:

- an unnamed free parameter
- an initial value
- a fixed value
- a free parameter with a name provided
- a free parameter with a name and initial value provided

The following example demonstrates these five types of specifications:

```
matrix A [1,2],
    [1,3] = (.2),
    [1,4] = .3,
    [2,3] = a1,
    [2,4] = a2(.5);
```
In this statement, A[1, 2] is an unnamed free parameter. For this element, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1). A[1, 3] is an unnamed free parameter with an initial value of 0.2. PROC CALIS also generates a parameter name for this element. A[1, 4] is fixed at 0.3. This value does not change in estimation. A[2, 3] is a free parameter named a1. No initial value is given for this element. A[2, 4] is a free parameter named a2 with an initial value of 0.5.

You can also specify different types of parameters in the parameter-spec list. The preceding specification is equivalent to the following specification:

```
matrix A [1,2],
[1 2,3 4] = (.2) .3 a1-a2 (.5);
```

Notice that 0.5 is the initial value for a2 but not for a1 because this specification is the same as:

```
matrix A [1,2],
[1 2,3 4] = (.2) .3 a1 a2(.5);
```

When you use parameter-spec lists with mixed parameters, you must be careful about how the initial value syntax is interpreted with and without a parameter name before it. With a parameter before the initial value, the initial value is for the parameter, as shown in the following statement:

```
matrix S [1,1] = s1 s2 (.2);
```

This specification is the same as the following elementwise specification:

```
matrix S [1,1] = s1,
[2,2] = s2(.2);
```

This means that 0.2 is the initial value of parameter s2, but not interpreted as an unnamed free parameter for S[3, 3]. If you do intend to set the free parameter s2 for S[2, 2] without an initial value and set the initial value 0.2 for S[3, 3], you can use a null initial value for the s2 parameter, as shown in the following:

```
matrix S [1,1] = s1 s2() (.2);
```

This specification is the same as the following elementwise specification:

```
matrix S [1,1] = s1,
[2,2] = s2,
[3,3] = (.2);
```

**Modifying a Parameter Specification from a Reference Model**

If you define a new COSAN, LISMOD, or MSTRUCT model by using a reference (old) model in the `REFMODEL` statement, you might want to modify some parameter specifications from the MATRIX statement of the reference model before transferring the specifications to the new model. To change a particular matrix element specification from the reference model, you can simply respecify the same matrix element with the desired parameter specification in the MATRIX statement of the new model. To delete a particular matrix parameter from the reference model, you can specify the desired matrix element with a missing value specification in the MATRIX statement of the new model.
For example, suppose that \(_\text{PHI}_\{1,2\}\) is a free parameter in the reference model but you do not want this matrix element be a free parameter in the new model, you can use the following specification in the new model:

```
matrix \(_\text{PHI}_\) [1,2] = .;
```

Notice that the missing value syntax is valid only when you use the \texttt{REFMODEL} statement. See the section “Modifying a COSAN Model from a Reference Model” on page 1249 for a more detailed example of COSAN model respecification. See the section “Modifying a LISMOD Model from a Reference Model” on page 1284 for a more detailed example of LISMOD model respecification. See the section “Modifying an MSTRUCT Model from a Reference Model” on page 1314 for a more detailed example of MSTRUCT model respecification.

**MEAN Statement**

```
MEAN assignment \(<, assignment \ldots >\);
```

where \texttt{assignment} represents:

```
var-list \(< = parameter-spec>\)
```

The MEAN statement specifies the mean or intercept parameters in connection with the FACTOR, LINEQS, and PATH modeling languages. With the MEAN statement specification, PROC CALIS analyzes the mean structures in addition to the covariance structures.

In each \texttt{assignment} of the MEAN statement, you list the \texttt{var-list} that you want to specify for their means or intercepts. Optionally, you can provide a list of parameter specifications in a \texttt{parameter-spec} after an equal sign for each \texttt{var-list}. The syntax of the MEAN statement is exactly the same as that of the \texttt{VARIANCE} statement. See the \texttt{VARIANCE} statement on page 1362 for details about the syntax.

For the confirmatory FACTOR or PATH model, the variables in a \texttt{var-list} can be exogenous or endogenous. You specify the mean of a variable if the variable is exogenous. You specify the intercept of a variable if the variable is endogenous. However, for the LINEQS model, you can specify only the means of exogenous variables whose type is not error (that is, not the E- or D- variables) in the MEAN statement. You cannot specify the intercept parameters in the MEAN statement for the LINEQS model. Instead, you must specify the intercepts in the equations of the \texttt{LINEQS} statement. For the exploratory FACTOR model, the MEAN statement is used merely to include the mean structures into analysis. In this case, the parameter specification by the \texttt{assignment} syntax is not interpreted. Instead, the intercept parameters for the exploratory factor model are automatically generated by PROC CALIS.

You can specify the following five types of the parameters for the means or intercepts in the MEAN statement:

- an unnamed free parameter
- an initial value
- a fixed value
- a free parameter with a name provided
- a free parameter with a name and initial value provided
For example, consider a PATH model with exogenous variables $x_1$, $x_2$, and $x_3$ and endogenous variables $y_4$ and $y_5$. The following MEAN statement illustrates the five types of specifications in five assignments:

```plaintext
mean x1,
    x2 = (3.0),
    x3 = 1.5,
    y4 = intercept1,
    y5 = intercept2(0.6);
```

In this statement, the mean of $x_1$ is specified as an unnamed free parameter. For this mean, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1). The mean of $x_2$ is an unnamed free parameter with an initial value of 3.0. PROC CALIS also generates a parameter name for this mean. The mean of $x_3$ is a fixed value of 1.5. This value stays the same during the estimation. The intercept of endogenous variable $y_4$ is a free parameter named intercept1. The intercept of endogenous variable $y_5$ is a free parameter named intercept2 with an initial value of 0.6.

The syntax of the MEAN statement is the same as the syntax of the VARIANCE statement. See the VARIANCE statement for more illustrations about the usage.

### Default Mean and Intercept Parameters

If the mean structures are analyzed, either the means or intercepts of the manifest variables in the FACTOR, LINEQS, or PATH model are free parameters by default. If a manifest variable is exogenous, then its mean is a free parameter by default. If a manifest variable is endogenous, then its intercept is a free parameter by default. Except for the exploratory FACTOR model, PROC CALIS generates parameter names for these default free mean or intercept parameters that include the _Add prefix and are appended with unique integer suffixes. The default intercepts of the exploratory FACTOR model use the _a prefix instead. For the confirmatory FACTOR model and the PATH model, you can use the MEAN statement to override these default mean or intercept parameters in situations where you want to set parameter constraints, provide initial or fixed values, or make parameter references. However, you cannot override any default intercept parameters for the exploratory FACTOR model. For the LINEQS model, you can use the MEAN statement specification to override only the default mean parameters. The intercept parameters of the LINEQS model must be specified in the equations of the LINEQS statement.

Fixed zero is another type of default mean or intercept parameters for the FACTOR, LINEQS, or PATH model. All the intercepts and means of the latent variables in these models are fixed zeros by default. For the confirmatory FACTOR and PATH models, you can override these default fixed zeros by using the MEAN statement specifications. However, you cannot override the fixed zero factor means for the exploratory FACTOR model. For the LINEQS model, you can override only the default fixed zeros of latent variables whose type is not error. That is, you can use the MEAN statement to override the default zero mean for the exogenous latent factors (excluding the error or disturbance variables) or use the LINEQS statement to override the default zero intercept for the endogenous latent factors. The fixed zero means for the error or disturbance variables in the LINEQS model reflects model restrictions—there is no way you can override these default zero means.

### Modifying a Mean or Intercept Parameter Specification from a Reference Model

If you define a new FACTOR (confirmatory factor model only), LINEQS, or PATH model by using a reference (old) model in the REFMODEL statement, you might want to modify some parameter specifications from the MEAN statement of the reference model before transferring the specifications to the new model. To change a particular mean or intercept specification from the reference model, you can simply respecify the same
mean or intercept with the desired parameter specification in the MEAN statement of the new model. To delete a particular mean or intercept parameter from the reference model, you can specify the desired mean or intercept with a missing value specification in the MEAN statement of the new model.

For example, suppose that the mean of \( F_1 \) is specified in the reference model, but you do not want this mean specification be transferred to the new model. You can use the following MEAN statement specification in the new model:

\[
\text{mean } F_1 = .;
\]

Note that the missing value syntax is valid only when you use with the REFMODEL statement. See the section “Modifying a FACTOR Model from a Reference Model” on page 1266 for a more detailed example of FACTOR model respecification. See the section “Modifying a LINEQS Model from a Reference Model” on page 1280 for a more detailed example of LINEQS model respecification. See the section “Modifying a PATH Model from a Reference Model” on page 1330 for a more detailed example of PATH model respecification.

As discussed in a preceding section, PROC CALIS generates default free mean or intercept parameters for manifest variables in the FACTOR, LINEQS, or PATH model if you do not specify them explicitly in the MEAN statement (and the LINEQS statement for the LINEQS model). When you use the REFMODEL statement for defining a reference model, these default free mean or intercept parameters in the old (reference) model are not transferred to the new model. Instead, the new model generates its own set of default free mean or intercept parameters after the new model is resolved from the reference model, the REFMODEL statement options, the RENAMEPARAM statement, and the MEAN statement (and the LINEQS statement for the LINEQS model) specifications in the new model. This also implies that if you want any of the mean or intercept parameters to be constrained across the models by means of the REFMODEL specification, you must specify them explicitly in the MEAN statement (or the LINEQS statement for the LINEQS model) of the reference model so that the same mean or intercept specification is transferred to the new model.

MODEL Statement

\[
\text{MODEL } i</options> ;
\]

where \( i \) is an assigned model number between 1 and 9999, inclusively.

A MODEL statement signifies the beginning of a model specification block and designates a model number for the model. All main and subsidiary model specification statements after a MODEL statement belong in that model until another MODEL or GROUP statement is encountered.

The MODEL statement itself does not serve the purpose of model specification, which is actually done by the main and subsidiary model specification statements that follow it. The MODEL statement serves as a “place-holder” of specification of a single model. It also makes the reference to a model easier with an assigned model number. For example, consider the following statements:

```
proc calis;
  group 1 / data=women_data;
  group 2 / data=men_data;
  model 1 / group=1 label='Women Model';
    {model 1 specification here}
  model 2 / group=2 label='Men Model';
    {model 2 specification here}
run;
```
This example illustrates a two-group analysis with two models. One is model 1 labeled as ‘Women Model’ in a MODEL statement. Another is model 2 labeled as ‘Men Model’ in another MODEL statement. The two groups, group 1 and group 2, as defined in two separate GROUP statements, are fitted by model 1 and model 2, respectively, as indicated by the GROUP= option of the MODEL statements. Within the scope of model 1, you provide model specification statements by using the main and subsidiary model specification statements. Usually, one of the following main model specification statements is used: FACTOR, LINEQS, LISMOD, MSTRUCT, PATH, RAM, or REFMODEL. Similarly, you provide another set of model specification statements within the scope of model 2.

Hence, for an analysis with a single group, the use of the MODEL statement is not necessary because the model that fits the group is unambiguous.

You can set model-specific options in each MODEL statement. All but two of these options are also available in the PROC CALIS statement. If you set these options in the PROC CALIS statement, they apply to all models, unless you specify them in the local MODEL statements. If you want to apply some options only to a particular model, specify these options in the MODEL statement that corresponds to that model.

You can also set group-specific options in the MODEL statement. These group options apply to the groups that are specified in GROUP= option of the MODEL statement. See the section “Options Available in the GROUP and PROC CALIS Statements” on page 1273 for a detailed descriptions of these group options.

### Options Available Only in the MODEL Statement

- **LABEL=** *name*
  
  specifies a label for the model. You can use any valid SAS names up to 256 characters for labels. You can also use quoted strings for labels.

- **NAME=** *name*
  
  specifies a label for the model. You can use any valid SAS names up to 256 characters for labels. You can also use quoted strings for labels.

- **GROUP=** [int-list] | {int-list}
- **GROUPS=** [int-list] | {int-list}
  
  specifies a list of integers which represent the groups to be fitted by the model.

### Options Available in the MODEL and PROC CALIS Statements

The following options are available in the MODEL and PROC CALIS statements. If you specify these options in the PROC CALIS statement, they are transferred to all MODEL statements. These options might be overwriten by the respecifications in the local MODEL statements.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEMPHAS= on page 1214</td>
<td>Emphasizes the diagonal entries</td>
</tr>
<tr>
<td>EFFPART</td>
<td>TOTEFF on page 1215</td>
</tr>
<tr>
<td>EXTENDPATH</td>
<td>GENPATH on page 1215</td>
</tr>
<tr>
<td>INEST= on page 1216</td>
<td>Specifies the data set that contains the initial values and constraints</td>
</tr>
<tr>
<td>INMODEL</td>
<td>INRAM= on page 1216</td>
</tr>
<tr>
<td>MEANSTR on page 1223</td>
<td>Analyzes the mean structures</td>
</tr>
</tbody>
</table>
### MSTRUCT Statement

**MSTRUCT Statement**

```
MSTRUCT <VAR=var-list> ;
```

MSTRUCT stands for matrix structures. As opposed to other modeling languages, in which the mean and covariance structures are implied from paths, equations, or complicated model matrix computations, the MSTRUCT language is for direct structured mean and covariance models.

In the MSTRUCT statement, you define the list of variables. You can use MATRIX statements to specify the parameters in the mean and covariance structures:

#### Option Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODIFICATION on page 1224</td>
<td>Computes modification indices</td>
</tr>
<tr>
<td>NOMEANSTR on page 1225</td>
<td>Deactivates the inherited MEANSTR option</td>
</tr>
<tr>
<td>NOMOD on page 1226</td>
<td>Suppresses modification indices</td>
</tr>
<tr>
<td>NOORDERSPEC on page 1226</td>
<td>Displays model specifications and results according to the input order</td>
</tr>
<tr>
<td>NOPARMNAME on page 1226</td>
<td>Suppresses the printing of parameter names in results</td>
</tr>
<tr>
<td>NOSTAND on page 1226</td>
<td>Suppresses the standardized output</td>
</tr>
<tr>
<td>NOSTDERR on page 1226</td>
<td>Suppresses standard error computations</td>
</tr>
<tr>
<td>ORDERSPEC on page 1229</td>
<td>Orders the model output displays according to the parameter types within each model</td>
</tr>
<tr>
<td>OUTEST= on page 1229</td>
<td>Specifies the data set that outputs the estimates and their covariance matrix</td>
</tr>
<tr>
<td>OUTMODEL</td>
<td>OUTRAM= on page 1229</td>
</tr>
<tr>
<td>PARMNAME on page 1230</td>
<td>Displays parameter names in model specifications and results</td>
</tr>
<tr>
<td>PDETERM on page 1231</td>
<td>Computes the determination coefficients</td>
</tr>
<tr>
<td>PESTIM on page 1231</td>
<td>Prints parameter estimates</td>
</tr>
<tr>
<td>PINITIAL on page 1231</td>
<td>Prints initial pattern and values</td>
</tr>
<tr>
<td>PLATCOV on page 1231</td>
<td>Computes the latent variable covariances and scoring coefficients</td>
</tr>
<tr>
<td>PRIMAT on page 1233</td>
<td>Displays results in matrix forms</td>
</tr>
<tr>
<td>READADDPARM on page 1234</td>
<td>Instructs generated default parameters be read in the INMODEL= data set</td>
</tr>
<tr>
<td>STDERR on page 1237</td>
<td>Computes the standard errors</td>
</tr>
</tbody>
</table>

### Options Available in MODEL, GROUP, and PROC CALIS Statements

Some options in the GROUP statement can also be specified in the MODEL statements. Group options that are specified the MODEL statements are transferred to the GROUP statements that define the groups that are fitted by the associated models in the MODEL statements. This is a little more convenient than setting the common group options individually in the GROUP statements for all fitted groups by a model. See the section “Options Available in GROUP, MODEL, and PROC CALIS Statements” on page 1273 for a reference of these options.
MSTRUCT \(<\text{VAR=var-list}>\);  
\hspace{2em} \text{MATRIX } _\text{COV}_ \text{ parameters-in-matrix } ;  
\hspace{2em} \text{MATRIX } _\text{MEAN}_ \text{ parameters-in-matrix } ;

You use the MATRIX _\text{COV}_ statement to specify the covariance and variance parameters in the structured covariance matrix. When applicable, you use the MATRIX _\text{MEAN}_ statement to specify the parameters in the structured mean vector. Each of these matrices can be specified no more than once within a model. See the MATRIX statement on page 1295 for details. If you do not use any MATRIX statement for specifying parameters, a saturated model is assumed. This means that all elements in the covariance and mean (if modeled) matrices are free parameters in the model.

The order of variables in the var-list of the MSTRUCT statement is important; it is used to refer to the row and column variables of the _\text{COV}_ and the _\text{MEAN}_ matrices. The variables specified in the list should be present in the input data set that is intended for the MSTRUCT model. With direct mean and covariance structures on the observed variables, no latent variables are explicitly involved in the MSTRUCT modeling language. However, this does not mean that the MSTRUCT modeling language cannot handle latent variable models. With additional specifications in the PARAMETERS and the SAS programming statements, it is possible to fit certain latent variable models by using the MSTRUCT modeling language. Despite this, the code might get too complicated and error-prone. Hence, using the MSTRUCT modeling language for latent variable modeling is not recommended for novice users. The LINEQS, LISMOD, PATH, or RAM modeling language should be considered first for latent variable modeling. For applications of the MSTRUCT modeling, see Yung, Browne, and Zhang (2014).

**Default Parameters**

It is important to understand the default parameters in the MSTRUCT model. If you know which parameters are default free parameters, you can make your specification more efficient by omitting the specifications of those parameters that can be set by default. For example, you do not need to specify any elements of the _\text{COV}_ matrix if all elements are supposed to free parameters. See the section “Default Parameters in the MSTRUCT Model” on page 1418 for details about the default parameters of the FACTOR model.

**Modifying an MSTRUCT Model from a Reference Model**

This section assumes that you use a REFMODEL statement within the scope of a MODEL statement and that the reference model (or base model) is also an MSTRUCT model. The reference model is called the old model, and the model that refers to the old model is called the new model. If the new model is not intended to be an exact copy of the old model, you can use the following extended MSTRUCT modeling language to make modifications within the scope of the MODEL statement for the new model. The syntax is similar to, but not exactly the same as, the ordinary MSTRUCT modeling language, as described in the section “MSTRUCT Statement” on page 1313. The syntax for respecifying or modifying an MSTRUCT model takes the following form:

\[
\text{MSTRUCT } ;  
\hspace{2em} \text{MATRIX } _\text{COV}_ \text{ parameters-in-matrix } ;  
\hspace{2em} \text{MATRIX } _\text{MEAN}_ \text{ parameters-in-matrix } ;
\]

In the respecification, you should not put any VAR= list in the MSTRUCT statement, as you would do for specifying the original base model. The reason is that parameter respecifications in the new model refer to the variables in the VAR= list of the old model. Therefore, the VAR= list in the new model is implicitly assumed to be exactly the same as that in the old model. This renders the specification of a VAR= list of the
MSTRUCT statement of the new model unnecessary. Because the VAR= option is the only possible option in the MSTRUCT statement, it also implies that the entire MSTRUCT statement is optional for the new model.

You can use the MATRIX _COV_ and MATRIX _MEAN_ statements to modify from the old model by using the same syntax as in ordinary MSTRUCT modeling language. In addition, in the respecification syntax, you can use the missing value ‘.’ to drop a parameter location from the old model.

The new model is formed by integrating with the old model in the following ways:

Duplication: If you do not specify in the new model a parameter location that exists in the old model, the old parameter specification is duplicated in the new model.

Addition: If you specify in the new model a parameter location that does not exist in the old model, the new parameter specification is used in the new model.

Deletion: If you specify in the new model a parameter location that also exists in the old model and the new parameter is denoted by the missing value ‘.’, the old parameter specification is not copied into the new model.

Replacement: If you specify in the new model a parameter location that also exists in the old model and the new parameter is not denoted by the missing value ‘.’, the new parameter specification replaces the old one in the new model.

For example, consider the following statements for a two-group analysis:

```plaintext
proc calis;
  group 1 / data=d1;
  group 2 / data=d2;
  model 1 / group=1;
    mstruct var=V1-V6;
    matrix _COV_ [1,1] = 6*vparm (8.),
      [2,] = cv21,
      [3,] = cv31,
      [4,] = cv41 cv42 cv43,
      [5,] = cv51 cv52 cv53 cv54,
      [6,] = cv61 cv62 cv63 cv64 cv65;
  model 2 / group=2;
    refmodel 1;
    matrix _COV_ [2,] = 3.,
      [3,2] = cv32,
      [4,] = . . . ,
      [5,] = . . . ,
      [6,] = . . . ;
run;
```

In these statements, you specify Model 2 by referring to Model 1 in the REFMODEL statement. Hence, Model 2 is called the new model that refers to the old model, Model 1. Because they are not respecified in the new model, all parameters on the diagonal of the covariance matrix are duplicated from the old model for the new model. Similarly, parameter locations associated with the cv54, cv64, and cv65 parameters are also duplicated in the new model.

An added parameter in the new model is cv32 for the covariance between V3 and V2. This parameter location is not specified in the old model.

In the new model, parameters for the covariances between the variable sets {V1 V2 V3} and {V4 V5 V6} are all deleted from the old model. The corresponding parameter locations for these covariances are given
missing values ‘.’ in the new model, indicating that they are no longer free parameters as in the old model.
Deleting these parameters amounts to setting the corresponding covariances to fixed zeros in the new model.

Finally, covariance between V2 and V1 is changed from a free parameter \( \text{cv21} \) in the old model to a fixed constant 3 in the new model. This illustrates the replacement rule of the respecification syntax.

---

**NLINCON Statement**

\[ \text{NLINCON | NLC constraint < , constraint . . . > ;} \]

where \( \text{constraint} \) represents one of the following:

- \( \text{number operator variable-list number operator} \)
- \( \text{variable-list number operator} \)
- \( \text{number operator variable-list} \)

You can specify nonlinear equality and inequality constraints with the NLINCON or NLC statement. The QUANEW optimization subroutine is used when you specify nonlinear constraints by using the NLINCON statement.

The syntax of the NLINCON statement is similar to that of the BOUNDS statement, except that the NLINCON statement must contain the names of variables that are defined in the program statements and are defined as continuous functions of parameters in the model. They must not be confused with the variables in the data set.

As with the BOUNDS statement, one- or two-sided constraints are allowed in the NLINCON statement; equality constraints must be one sided. Valid operators are \( \leq, <, \geq, >, \) and \( = \) (or, equivalently, \( \text{LE}, \text{LT}, \text{GE}, \text{GT}, \text{and EQ} \)).

PROC CALIS cannot enforce the strict inequalities \( < \) or \( > \) but instead treats them as \( \leq \) and \( \geq \), respectively. The listed nonlinear constraints must be separated by commas. The following is an example of the NLINCON statement that constrains the nonlinear parametric function \( x_1 x_1 + u_1 \) to a fixed value of 1:

\[
\text{nlincon xx = 1; xx = x1 * x1 + u1;}
\]

Note that \( x_1 \) and \( u_1 \) are parameters defined in the model. The following three NLINCON statements, which require \( xx1, xx2, \) and \( xx3 \) to be between zero and ten, are equivalent:

\[
\text{nlincon 0. <= xx1-xx3, xx1-xx3 <= 10;} \\
\text{nlincon 0. <= xx1-xx3 <= 10.;} \\
\text{nlincon 10. >= xx1-xx3 >= 0.;}
\]

---

**NLOPTIONS Statement**

\[ \text{NLOPTIONS options ;} \]

Many options that are available in SAS/OR PROC NLP can be specified for the optimization subroutines in PROC CALIS by using the NLOPTIONS statement. The NLOPTIONS statement provides more displayed and file output control on the results of the optimization process, and it permits the same set of termination criteria as in PROC NLP. These are more technical options that you might not need to specify in most cases.
Several statistical procedures support the use of NLOPTIONS statement. The syntax of NLOPTIONS statement is common to all these procedures and can be found in the section “NLOPTIONS Statement” on page 488 in Chapter 19, “Shared Concepts and Topics.”

See the section “Use of Optimization Techniques” on page 1507 for more information about the use of optimization techniques in PROC CALIS.

### OUTFILES Statement

```
OUTFILES | OUTFILE file-option < file-option . . . > ;
```

where file-option represents one of the following:

- **OUTMODEL** | **OUTRAM** = file-name [ **MODEL** = int-list < , int-list > ]
- **OUTSTAT** = file-name [ **GROUP** = int-list < , int-list > ]
- **OUTWGT** = file-name [ **GROUP** = int-list < , int-list > ]

with file-name representing an output file name and int-list representing list of model or group numbers

Use the OUTFILES statement when you need to output multiple-group or multiple-model information to output files in a complex way. In each OUTFILES statement, each possible file-option should appear no more than once. However, as needed, you can use the OUTFILES statement more than once. For example, suppose you want to create two OUTWGT= files for different sets of groups. You can specify the OUTFILES statement twice, as shown in the following specification:

```
outfiles outwgt=file1 [group=1,2];
outfiles outwgt=file2 [group=3,4];
```

In the first OUTFILES statement, the weights for groups 1 and 2 are output to the file file1. In the second OUTFILES statement, the weights for groups 3 and 4 are output to the file file2.

When the OUTMODEL=, OUTSTAT=, or OUTWGT= option is intended for all groups or models, you can simply specify the option in the PROC CALIS statement. Only when you need to output the group (model) information from more than one group (model), but not all groups (models), to a single output file does the use the OUTFILES statement become necessary. For example, consider the following specification:

```
proc calis method=gls;
  outfiles outmodel=outmodel [model=1,3]
    outwgt=outwgt [group=1,2]
    outstat=outstat [group=2,3];
  group 1 / data=g1;
  group 2 / data=g2;
  group 3 / data=g3 outwgt=outwgt3;
  model 1 / group=1;
    factor N=3;
  model 2 / group=2;
    factor N=2;
  model 3 / group=3;
    factor N=3;
run;
```
You fit three different factor models to three groups: Model 1 for Group 1, Model 2 for Group 2, and Model 3 for group 3. In the OUTFILES statement, you output model information from models 1 and 3 to an output file named outmodel, weight matrices from groups 1 and 2 to outwgt, and statistics from groups 2 and 3 to outstat. In each of these output files, you have information from more than one (but not all) groups or models. In the GROUP statement for group 3, you have another OUTWGT= file named outwgt3 for group 3 alone.

Note that you cannot specify the preceding output file organization by using the following statements:

```sas
proc calis method=gls;
  group 1 / data=g1 outwgt=outwgt;
  group 2 / data=g2 outwgt=outwgt outstat=outstat;
  group 3 / data=g3 outwgt=outwgt3 outstat=outstat;
  model 1 / group=1 outmodel=outmodel;
      factor N=3;
  model 2 / group=2;
      factor N=2;
  model 3 / group=3 outmodel=outmodel;
      factor N=3;
run;
```

This specification will not work because SAS forbids the repeated specification of the same output file in the same PROC CALIS run. That is, you cannot specify OUTWGT=outwgt, OUTSTAT=outstat, or OUTMODEL=outmodel more than once in the PROC CALIS run without causing file invocation problems (however, multiple specification of the same input file is not a problem).

If you specify any of the output files for a group (or a model) in both of the OUTFILES and the GROUP (or MODEL) statements, the destination specified in the more specific GROUP (or MODEL) statement will be used. For example, for the following specification PROC CALIS will save the Model 2 information in the OUTMODEL=outmodel2 data set, but not in the OUTMODEL=outfile1 data set:

```sas
proc calis method=gls;
  outfiles outmodel=outfile1 [model=1,2];
  group 1 / data=g1;
  group 2 / data=g2;
  model 1 / group=1;
      factor N=3;
  model 2 / group=2 outmodel=outmodel2;
      factor N=2;
run;
```

The OUTFILES statement is intended for arranging output files in a complex way. The use of the OUTFILES statement is unnecessary in the following situations:

- If you have a single-sample analysis, you do not need to use the GROUP statement. As a result, you can simply use the OUTSTAT= or OUTWGT= options in the PROC CALIS statement for specifying the output destinations. Therefore, the OUTFILES statement is not needed.

- If you have a single model in your analysis, you do not need to use the MODEL statement. As a result, you can simply use the OUTMODEL= options in the PROC CALIS statement for specifying the output destination. Therefore, the OUTFILES statement is not needed.

- If you have multiple groups or multiple models in your analysis and information for all groups or models is output to the same file, you do not need to use the OUTFILES statement. You can simply
use the OUTSTAT=, OUTWGT=, or OUTMODEL= options in the PROC CALIS statement because the output file information is automatically propagated from the PROC CALIS statement to the groups or models.

- If you have multiple groups or multiple models in your analysis and each group or model has a unique output data file destination (including cases where some groups or models might not have any output files), you do not need to use the OUTFILES statement. You can simply specify the OUTSTAT=, OUTWGT=, or OUTMODEL= options in the GROUP or MODEL statements.

**PARAMETERS Statement**

PARAMETERS | PARMS parameters <= numbers | (numbers)> <=, parameters <= numbers | (numbers)> . . . > ;

The PARAMETERS statement defines parameters that are useful in the modeling. You can specify more than one PARAMETERS statement.

The three main uses of the PARAMETERS statement are as follows:

• Define independent parameters that are not specified in your model. In some modeling situations, it is more convenient (practically or conceptually) to define the model parameters as functions of these independent parameters. PROC CALIS computes the estimates and their standard errors for these independent parameters.

• Define dependent parameters of interest. These dependent parameters are then defined as functions of the model parameters in the SAS programming statements. PROC CALIS computes the estimates and their standard errors for these dependent parameters.

• Provide initial values for the free parameters in the model if they have not been specified in the model specification statements.

For example, the following statements illustrate the three main uses of the PARAMETERS statement:

```
proc calis;
  path
    V1 <= V2 = b1,
    V1 <= V3 = b2,
    V2 <= V4 = b3 (0.9),
    V3 <= V5 = b4;
  parameters a1 (0.1) a2 (0.2) b3 (0.3) b4 (0.4) b5;
  b1 = -a1 / a2;
  b2 = 1 / a2;
  b5 = b3 - b4;
run;
```

In the PARAMETERS statement, you specify five parameters that take different roles in the modeling. Parameters a1 and a2 are independent parameters that are not specified in the PATH statement, which specifies four paths with four parameters for the path coefficients b1, b2, b3, and b4. The two SAS programming statements immediately after the PARAMETERS statement define parameters b1 and b2 as functions of parameters a1 and a2. Because a1 and a2 appear only on the right side of the programming
In addition, b1 and b2 are dependent parameters because they appear on the left side of the first two programming statements. Independent parameters a1 and a2 are provided with initial values 0.1 and 0.2, respectively, in the PARAMETERS statement. Because the initial values these two independent parameters are used, dependent parameters b1 and b2 are also initialized with values –0.5 and 5, respectively.

Parameters b3 and b4 appear in both the PATH and PARAMETERS statements. Because these two parameters are already specified in the PATH statement, their initial values might have been defined in the model. For example, parameter b3 in the PATH statement has an initial value of 0.9. This initial value is not replaced by the initial value of 0.3 specified in the PARAMETERS statement. However, because you do not specify an initial value for parameter b4 in the PATH statement, the initial value of 0.4 specified in the PARAMETERS statement is used. In general, the initial values that are specified in the PARAMETERS statement do not replace the initial values that have already been specified for the same parameters in the model.

Parameter b5 in the PARAMETERS statement is a dependent parameter, which is defined as the difference between the model parameters b3 and b4 in the last SAS programming statement. No initial value for this dependent parameter is provided (nor is it needed). By definition, dependent parameters are functions of other parameters, so their initial values are computed. In this example, parameter b5 takes an initial value of 0.5, the difference between the initial values of b3 and b4.

It is not necessary to provide initial values for the parameters in the PARAMETERS statement. For the independent parameters without initial values specified, PROC CALIS generates the initial values from the START= value in the PROC CALIS statement. The default START= value is 0.5. If you specify the RANDOM= option in the PROC CALIS statement, random numbers are generated for the initial values of the independent parameters. For the dependent parameters, PROC CALIS computes the initial values by using the SAS programming statements.

In general, the number of parameters and the number of initial values do not have to match. When you specify fewer initial values than parameter names, PROC CALIS either generates or computes the initial values. When you specify more values than parameter names, the extra values are ignored.

For example, consider the following statements:

```plaintext
parameters c1 c2 c3 c4 (0.2 0.3 0.4);
parameters d1 d2 d3 d4 (0.1 0.2 0.3 0.4 0.5);
run;
```

The first PARAMETERS statement has a shorter initial value list than the parameter list. The initial values 0.2, 0.3, and 0.4 are assigned to c2, c3, and c4, respectively. PROC CALIS generates the initial value for c1. The second PARAMETERS statement has a longer initial value list than the parameter list. The initial values 0.1, 0.2, 0.3, and 0.4 are assigned to d1, d2, d3, and d4, respectively. The extra initial value 0.5 is ignored.

When the lengths of the parameter list and the initial value list match, you can use an equal sign between the lists freely without affecting the assignments of the initial values. For example, each of the following PARAMETERS statements assigns the initial values alpha=0.5 and beta=–0.5:

```plaintext
parameters alfa 0.5 beta -0.5;
parameters alfa = 0.5 beta = -0.5;
parameters alfa (0.5) beta (-0.5);
parameters alfa = (0.5) beta = (-0.5);
parameters alfa beta 0.5 -0.5;
parameters alfa beta = 0.5 -0.5;
parameters alfa beta (0.5 -0.5);
parameters alfa beta = (0.5 -0.5);
```
However, when the parameter list is longer than the initial value list, the equal sign affects the assignments of the initial values. For example, the following statement assigns the initial values to the last three parameters, \( c_2 \), \( c_3 \), and \( c_4 \):

```
parameters c1 c2 c3 c4 (0.2 0.3 0.4);
```

But with an equal sign between the lists, the following statement assigns the initial values to the first three parameters, \( c_1 \), \( c_2 \), and \( c_3 \):

```
parameters c1 c2 c3 c4 = (0.2 0.3 0.4);
```

To make initial value assignments less ambiguous, you can use commas to separate parameter lists. For example, the following statement clearly assigns the initial values to \( c_2 \), \( c_3 \), and \( c_4 \):

```
parameters c1, c2 c3 c4 (0.2 0.3 0.4);
```

This specification is equivalent to the same specification without the comma. But the specification is certainly less ambiguous with the comma.

Proper grouping of the parameter and initial value lists with the use of commas helps make the specification in the PARAMETERS statement clear. For example, consider the following specification:

```
parameters c1 c2 c3 c4 = (0.1 0.2 0.3) c5 c6 (0.6);
```

Initial values 0.1, 0.2, 0.3, and 0.6 are assigned to parameters \( c_1 \), \( c_2 \), \( c_3 \), and \( c_6 \), respectively. Initial values for other parameters are either generated or computed by PROC CALIS. A much better practice is to use the following equivalent specification:

```
parameters c1 c2 c3 (0.1 0.2 0.3), c4 c5, c6 (0.6);
```

Matching the number of parameters and the number of initial values (if provided) in entries separated by commas in the PARAMETERS statement is highly recommended. It reduces ambiguities and makes your code more interpretable.

It is important to notice that PROC CALIS does not have better alternatives to generate initial values (such as those for setting the initial values of the model parameters) for the independent parameters specified in the PARAMETERS statement other than using either the START= value or random values generated by using the RANDOM= option. These ad hoc initial values are arbitrary and might not always lead to converged solutions. Therefore, you should try to provide good initial values for those independent parameters that are defined only in the PARAMETERS statement.

Do not confuse the PARAMETERS statement with the VAR statement. While you specify the parameters of the model in the PARAMETERS statement, you specify analysis variables in the VAR statement. See the VAR statement on page 1359 for more details.

**CAUTION:** The OUTMODEL= and OUTRAM= data sets do not contain any information about the PARAMETERS statement or the SAS programming statements.
**PARTIAL Statement**

**PARTIAL** variables ;

If you want the analysis to be based on a partial correlation or covariance matrix, use the PARTIAL statement to list the variables used to partial out the variables in the analysis. You can specify only one PARTIAL statement within the scope of each GROUP or PROC CALIS statement.

**PATH Statement**

**PATH** path <, path ... > ;

where path represents any of the following specifications:

- single-headed path for defining functional relationship
- double-headed path for specifying variances or covariances
- 1-path for specifying means or intercepts

For example, the following PATH statement contains only the single-headed paths:

```plaintext
path
  v1 ===> v2,  /* same as: v2 ===> v4 and v2 ===> v5 */
  v3 ===> v5,  /* same as: v5 ===> v3 */
  v4 v5 ===> v6 v7;  /* same as: v4 ===> v6, v4 ===> v7, v5 ===> v6, and v5 ===> v7 */
```

Although the most common definition of paths refer to these single-headed paths, PROC CALIS extends the definition of paths to include the so-called “variance-paths,” “covariance-paths,” and “1-paths” that refer to the variance, covariance, and the mean or intercept parameters, respectively. Corresponding to these extended path definitions, PROC CALIS provides the double-headed path and 1-path syntax. For example, the following PATH statement contains single-headed paths for specifying functional relationships and double-headed paths for specifying variances and covariances:

```plaintext
path
  v1 ===> v3-v5,  /* same as: v1 ===> v3, v1 ===> v4, and v1 ===> v5 */
  v2 ===> v4 v5,  /* same as: v2 ===> v3, v2 ===> v3 */
  v3 ===> v5,  /* error variance of v5 */
  v1 ===> v1,  /* error variance of v1 */
  v2 ===> v3,  /* error variance of v2 and v3 */
  v2 ===> v3,  /* error covariance between v2 and v3 */
  v3 ===> [v4 v5];  /* variances and covariance for v4 and v5 */
```
The following PATH statement contains single-headed paths for specifying functional relationships and 1-paths for specifying means and intercepts:

```
path
  v1  ===> v3-v5,
  v2  ===> v4 v5,
  v3  ===> v5,
  1   ===> v1,    /* intercepts for v1 */
  1   ===> v2-v3, /* intercepts for v2 and v3 */
  1   ===> v4 v5; /* means of v4 and v5 */
```

Details about the syntax of these three different types of *paths* are described later. Instead of using double-headed paths and 1-paths, you can also specify these parameters by the *subsidiary model specification statements* such as the PVAR, PCOV, and the MEAN statements, as shown in the following syntactic structure of the PATH modeling language:

```
PATH path <, path ... > ;
  PVAR partial-variance-parameters ;
  PCOV partial-covariance-parameters ;
  MEAN mean-parameters ;
```

Typically, in this syntactic structure the *paths* contains only single-headed paths for representing the functional relationships among variables, which could be observed or latent. The *paths* are separated by commas. You can specify at most one PATH statement in a model within the scope of either the PROC CALIS statement or a MODEL statement.

Next, the PVAR statement specifies the parameters for the variances or error (partial) variances. The PCOV statement specifies the parameters for the covariances or error (partial) covariances. The MEAN statement specifies the parameters for the means or intercepts. For details about these subsidiary model specification statements, see the syntax of the individual statements.

A natural question now arises. For the specification of variances, covariances, intercepts, and means, should you use the extended path syntax that includes double-headed paths and 1-paths or the subsidiary model specification statements such as the PVAR, PCOV, and MEAN statements? If you want to specify all parameters in a single statement and hence output and view all the parameter estimates in a single output table, then the extended path syntax would be your choice. If you want to use more common language for specifying and viewing the parameters or the estimates of variances, covariances, means, and intercepts, then the subsidiary model specification statements serve the purpose better.

You are not restricted to using extended path syntax or the subsidiary model statements exclusively in a PATH model specification. For example, you might specify the variance of v1 by using the double-headed path syntax and the variance of v2 by using the PVAR statement. The only restriction is that you cannot specify the same parameter twice. In addition, even if you specify your PATH model without using double-headed paths or 1-paths, you can include the estimation results associated with these extended paths in the same output table for the single-headed paths by using the EXTENDPATH or GENPATH option. This way all the estimates of the PATH model can be shown in a single output table.
The Single-Headed Path Syntax for Specifying Functional Relationships

\[ \text{var-list} \text{ arrow var-list2} < = \text{parameter-spec} > \]

where \text{var-list} and \text{var-list2} are lists of variables, \text{parameter-spec} is an optional specification of parameters, and \text{arrow} represents either a left-arrow or a right-arrow.

A left-arrow is one of the following forms:

\[ <==, <<-,, <=, <=, <=, <=, <, \text{or} < \]

or a right-arrow is one of the following forms:

\[ ==>, -->, ==>, -->, =>, -->, \text{or} > \]

In each single-headed path, you specify two lists of variables: \text{var-list} and \text{var-list2}. Depending on the direction of the \text{arrow} specification, one group of variables contains the outcome variables and the other group contains the predictor variables. Optionally, you can specify the \text{parameter-spec} at the end of each path entry. You can specify the following five types of the parameters for the path entries:

- unnamed free parameters
- initial values
- fixed values
- free parameters with names provided
- free parameters with names and initial values provided

For example, in the following statement you specify a model with five paths:

\begin{verbatim}
path
  v1 <== f1 ,
  v2 <== f1 = (0.5),
  v3 <== f1 = 1.0,
  v4 <== f1 = b1,
  v5 <== f1 = b2 (.4);
\end{verbatim}

The first path entry specifies a path from \( f1 \) to \( v1 \). The effect of \( f1 \) (or the path coefficient) on \( v1 \) is an unnamed free parameter. For this path effect parameter, PROC CALIS generates a parameter name with the \_Parm prefix and appended with a unique integer (for example, \_Parm1). The second path entry specifies a path from \( f1 \) to \( v2 \). The effect of \( f1 \) is also an unnamed free parameter with an initial estimate of 0.5. PROC CALIS also generates a parameter name for effect parameter. The third path entry specifies a path from \( f1 \) to \( v3 \). The effect of \( f1 \) is also a fixed value of 1.0. This value stays the same in the model estimation. The fourth path entry specifies a path from \( f1 \) to \( v4 \). The effect of \( f1 \) is a free parameter named \( b1 \). The fifth path entry specifies a path from \( f1 \) to \( v5 \). The effect of \( f1 \) is a free parameter named \( b2 \), with an initial value of 0.4.

You can specify multiple variables in the \text{var-list} and \text{var-list2} lists. For example, the following statement specifies five paths from \( f1 \) to \( v1-v5 \):

\begin{verbatim}
path
  f1 ===> v1-v5;
\end{verbatim}

All the five effects of \( f1 \) on the five variables are unnamed free parameters. If both \text{var-list} and \text{var-list2} lists contain multiple variables, you must be careful about the order of the variables when you also specify parameters at the end of the path entry. For example, the following statement specifies the paths from the predictor variables \( x1-x2 \) to the outcome variables \( y1-y3 \):
The PATH statement specifies six paths in the path entry. These six paths have effect parameters \( a_1 \)–\( a_6 \). This specification is equivalent to the following specification:

\[
\begin{align*}
\text{path} \\
y_1 \leq x_1 = a_1; \\
y_1 \leq x_2 = a_2; \\
y_2 \leq x_1 = a_3; \\
y_2 \leq x_2 = a_4; \\
y_3 \leq x_1 = a_5; \\
y_3 \leq x_2 = a_6;
\end{align*}
\]

The following statement shows another example of multiple-path specification:

\[
\begin{align*}
\text{path} \\
x_1 \leq y_1 \leq y_3 = b_1-b_6;
\end{align*}
\]

This specification is equivalent to the following specification with separate path specifications:

\[
\begin{align*}
\text{path} \\
x_1 \leq y_1 = b_1; \\
x_1 \leq y_2 = b_2; \\
x_2 \leq y_3 = b_3; \\
x_2 \leq y_1 = b_4; \\
x_2 \leq y_2 = b_5; \\
x_2 \leq y_3 = b_6;
\end{align*}
\]

You can also specify parameter with mixed types in any path entry, as shown in the following specification:

\[
\begin{align*}
\text{path} \\
f_1 \leq y_1 \leq y_3 = 1. b_1(.5) (.3), \\
f_2 \leq y_4 \leq y_6 = 1. b_2 b_3(.7);
\end{align*}
\]

This specification is equivalent to the following expanded version:

\[
\begin{align*}
\text{path} \\
f_1 \leq y_1 = 1., \\
f_1 \leq y_2 = b_1(.5), \\
f_1 \leq y_3 = (.3), \\
f_2 \leq y_4 = 1., \\
f_2 \leq y_5 = b_2, \\
f_2 \leq y_6 = b_3(.7);
\end{align*}
\]

Notice that in the original specification with multiple-path entries, 0.5 is interpreted as the initial value for the parameter \( b_1 \), but not as the initial estimate for the path from \( f_1 \) to \( y_3 \). In general, an initial value that follows a parameter name is associated with the free parameter.

If you indeed want to specify that \( b_1 \) is a free parameter \textit{without} an initial estimate and 0.5 is the initial estimate for the path from \( f_1 \) to \( y_3 \) (while keeping all other specification the same), you can use a null initial value specification, as shown in the following statement:
Chapter 29: The CALIS Procedure

path
  f1 ===> y1-y3 = 1. b1() (.5) ,
  f2 ===> y4-y6 = 1. b2 b3(.7);

This way 0.5 becomes the initial value for the path from f1 to y3. Because a parameter list with mixed types might be confusing, you can break down the specifications into separate path entries to remove ambiguities. For example, you can use the following specification equivalently:

path
  f1 ===> y1 = 1.,
  f1 ===> y2 = b1,
  f1 ===> y3 = (.5) ,
  f2 ===> y4-y6 = 1. b2 b3(.7);

The equal signs in the path entries are optional when the parameter lists do not start with a parameter name. For example, the preceding specification is the same as the following specification:

path
  f1 ===> y1 1.,
  f1 ===> y2 = b1,
  f1 ===> y3 (.5) ,
  f2 ===> y4-y6 1. b2 b3(.7);

Notice that in the second path entry, you must retain the equal sign because b1 is a parameter name. Omitting the equal sign makes the specification erroneous because b1 is treated as a variable. This might cause serious estimation problems. Omitting the equal signs might be cosmetically appealing in specifying fixed values or initial values (for example, the first and the third path entries). However, the gain of doing that is not much as compared to the clarity of specification that results from using the equal signs consistently.

NOTE: You do not need to specify single-headed paths from the errors or disturbances (that is, error terms) in the PATH model specification, even though the functional relationships between variables are not assumed to be perfect. Essentially, the roles of error terms in the PATH model are in effect represented by the associated default error variances of the endogenous variables, making it unnecessary to specify any single-headed paths from error or disturbance variables.

The Double-Headed Path Syntax That Uses Two Variable Lists for Specifying Variances and Covariances

\texttt{var-list two-headed-arrow var-list2 < = parameter-spec >}

where a \texttt{two-headed-arrow} is one of the following forms:
\texttt{<==>, <--> , <=>, <-> , or <>}

This syntax enables you to specify covariances between the variables in \texttt{var-list} and the variables in \texttt{var-list2}. Consider the following example:

path
  v1 <==> v2,
  v3 v4 <==> v5 v6 v7 = cv1-cv6;

The first double-headed path specifies the covariance between v1 and v2 as an unnamed free parameter. PROC CALIS generates a name for this unnamed free parameter. The second double-headed path specifies six covariances with parameters named cv1–cv6. This multiple-covariance specification is equivalent to the following elementwise covariance specification:
path
v3  <==>  v5  = cv1,
v3  <==>  v6  = cv2,
v3  <==>  v7  = cv3,
v4  <==>  v5  = cv4,
v4  <==>  v6  = cv5,
v4  <==>  v7  = cv6;

Note that the order of variables in the list is important for determining the assignment of the parameters in the parameter-spec list.

If the same variable appears in both of the var-list and var-list2 lists, the “covariance” specification becomes a variance specification for that variable. For example, the following statement specifies two variances:

path
v1  <==>  v1  = 1.0,
v2  <==>  v2  v3 = sigma2  cv23;

The first double-headed path entry specifies the variance of v1 as a fixed value of 1.0. The second double-headed path entry specifies the variance of v2 as a free parameter named sigma2, and then the covariance between v2 and v3 as a free parameter named cv23.

It results in an error if you attempt to use this syntax to specify the variance and covariances among a set of variables. For example, suppose you intend to specify the variances and covariances among v1–v3 as unnamed free parameters by the following statement:

path
v1-v3  <==>  v1-v3 ;

This specification expands to the following elementwise specification:

path
v1  <==>  v1 ,
v1  <==>  v2 ,
v1  <==>  v3 ,
v2  <==>  v1 ,
v2  <==>  v2 ,
v2  <==>  v3 ,
v3  <==>  v1 ,
v3  <==>  v2 ,
v3  <==>  v3 ;

There are nine variance or covariance specifications, but all of the covariances are specified twice. This is treated as a duplication error. The correct way is to specify only the nonredundant covariances, as shown in the following elementwise specification:

path
v1  <==>  v1 ,
v2  <==>  v1 ,
v2  <==>  v2 ,
v3  <==>  v1 ,
v3  <==>  v2 ,
v3  <==>  v3 ;
However, the elementwise specification is quite tedious when the number of variables is large. Fortunately, there is another syntax for double-headed paths to deal with this situation. This syntax is discussed next.

**The Double-Headed Path Syntax That Uses a Single Variable List for Specifying Variances**

```
two-headed-arrow var-list < = parameter-spec>
```

This syntax enables you to specify variances among the variables in `var-list`. Consider the following example:

```
path
  <==> v1   = (0.8),
  <==> v2-v4 ;
```

The first double-headed path entry specifies the variance of `v1` as an unnamed free parameter with an initial estimate of 0.8. The second double-headed path entry specifies the variances of `v2`–`v4` as unnamed free parameters. No initial values are given for these three variances. PROC CALIS generates names for all these variance parameters. You can specify these variances equivalently by the elementwise covariance specification syntax, as shown in the following, but former syntax is much more efficient.

```
path
  v1 <==> v1   = (0.8),
  v2 <==> v2  ,
  v3 <==> v3 ,
  v4 <==> v4 ;
```

**The Double-Headed Path Syntax That Uses a Single Variable List for Specifying Variances and Covariances**

```
two-headed-arrow [var-list] < = parameter-spec>
```

This syntax enables you to specify all the variances and covariances among the variables in `var-list`. For example, the following statement specifies all the variances and covariances among `v2`–`v4`:

```
path
  <==> [v2-v4] = 1.0 cv32 cv33(0.5) cv42 .7 cv44;
```

This specification is more efficient as compared with the following equivalent specification with elementwise variance or covariance definitions:

```
path
  v2 <==> v2   = 1.0,
  v3 <==> v2   = cv32 ,
  v3 <==> v3   = cv33(0.5),
  v4 <==> v2   = cv42,
  v4 <==> v3   = .7,
  v4 <==> v2   = cv44;
```

**The double-headed path Syntax for Specifying Nonredundant Covariances**

```
two-headed-arrow (var-list)< = parameter-spec>
```

This syntax enables you to specify all the nonredundant covariances among the variables in `var-list`. For example, the following statement specifies all the nonredundant covariances between `v2`–`v4`:
This specification is equivalent to the following elementwise specification:

```
path
  v3 <= v2 = cv1,
v4 <= v2 = cv2,
v4 <= v3 = cv3,
v5 <= v2 = cv4,
v5 <= v3 = cv5,
v5 <= v4 = cv6;
```

**The 1-path Syntax for Specifying Means and Intercepts**

```
1 right-arrow var-list <= parameter-spec>
```

where a `right-arrow` is one of the following forms:

```
===>, ---->, ==>, =>, <-, or >
```

This syntax enables you to specify the means or intercepts of the variables in `var-list` as paths from the constant 1. Consider the following example:

```
path
  v1 <= v2-v4,
  1 ==> v1 = alpha,
  1 ==> v2-v4 = 3*kappa;
```

The first single-headed path specifies that `v1` is predicted by variables `v2`, `v3`, and `v4`. Next, the first 1-path entry specifies either the intercept of `v1` as a free parameter named `alpha`. It is the intercept, rather than the mean, of `v1` because endogenous in the PATH model. The second 1-path entry specifies the means of `v2-v4` as constrained parameters. All these means or intercepts are named `kappa` so that they have the same estimate.

Therefore, whether the parameter is a mean or an intercept specified with the 1-path syntax depends on whether the associated variable is endogenous or exogenous in the model. The intercept is specified if the variable is endogenous. Otherwise, the mean of the variable is specified. Fortunately, any variable in the model can have either a mean or intercept (but not both) to specify. Therefore, the 1-path syntax is applicable to either the mean or intercept specification without causing conflicts.

**Shorter and Longer Parameter Lists**

If you provide fewer parameters in `parameter-spec` than the number of paths in a `path` entry, all the remaining parameters are treated as unnamed free parameters. For example, the following specification specifies the free parameter `beta` to the first path and assigns unnamed free parameters to the remaining four paths:

```
path
  f1 ==> y1 z1 z2 z3 z4 = beta;
```
This specification is equivalent to the following specification:

```plaintext
path
  f1 ===> y1 = beta,
  f1 ===> z1 z2 z3 z4;
```

If you intend to fill up all values with the last parameter specification in the list, you can use the continuation syntax `...`, `[...]`, or `[.]`, as shown in the following example:

```plaintext
path
  f1 ===> y1 z1 z2 z3 z4 = beta gamma [...];
```

This specification is equivalent to the following specification:

```plaintext
path
  f1 ===> y1 z1 z2 z3 z4 = beta 4*gamma;
```

The repetition factor `4*` means that `gamma` repeats 4 times.

However, you must be careful not to provide too many parameters. For example, the following specification results in an error:

```plaintext
path
  SES_Factor ===> y1 z1 z2 z3 z4 = beta gamma1-gamma6;
```

Because there are only five paths in the specification, parameters `gamma5` and `gamma6` are excessive.

### Default Parameters

It is important to understand the default parameters in the PATH model. First, knowing which parameters are default free parameters makes your specification more efficient by omitting the specifications of those parameters that can be set by default. For example, because all variances and covariances among exogenous variables (excluding error terms) are free parameters by default, you do not need to specify them in the PATH model if these variances and covariances are not constrained. Second, knowing which parameters are default fixed zero parameters enables you to specify your model accurately. For example, because all error covariances in the PATH model are fixed zeros by default, you must use the PCOV statement or the double-headed path syntax to specify the partial (error) covariances among the endogenous variables if you want to fit a model with correlated errors. See the section “Default Parameters in the PATH Model” on page 1424 for details about the default parameters of the PATH model.

### Modifying a PATH Model from a Reference Model

If you define a new model by using a reference (old) model in the REFMODEL statement, you might want to modify some path specifications from the PATH statement of the reference model before transferring the specifications to the new model. To change a particular path specification from the reference model, you can simply respecify the same path with the desired parameter specification in the PATH statement of the new model. To delete a particular path and its associated parameter from the reference model, you can specify the desired path with a missing value specification in the PATH statement of the new model.

```plaintext
PATH path <, path ... > ;
  PVAR partial-variance-parameters ;
  PCOV partial-covariance-parameters ;
  MEAN mean-parameters ;
```
The new model is formed by integrating with the old model in the following ways:

**Duplication:** If you do not specify in the new model a parameter location that exists in the old model, the old parameter specification is duplicated in the new model.

**Addition:** If you specify in the new model a parameter location that does not exist in the old model, the new parameter specification is used in the new model.

**Deletion:** If you specify in the new model a parameter location that also exists in the old model and the new parameter is denoted by the missing value ‘.’, the old parameter specification is not copied into the new model.

**Replacement:** If you specify in the new model a parameter location that also exists in the old model and the new parameter is not denoted by the missing value ‘.’, the new parameter specification replaces the old one in the new model.

For example, consider the following specification of a two-group analysis:

```plaintext
proc calis;
  group 1 / data=d1;
  group 2 / data=d2;
  model 1 / group=1;
    path
      v1 ===> f1 = 1.,
      v2 ===> f1 = load1,
      v3 ===> f1 = load2,
      f1 ===> v4 = b1,
      f1 ===> v5 = b2,
      f1 ===> v6 = b3;
  pvar
    E1-E3 = ve1-ve3,
    f1 = vd1,
    v5-v6 = phi4-phi6;
  pcov
    v1 v2 = cve12;
model 2 / group=2;
  refmodel 1;
  path
    v3 ===> f1 = load1,
  pcov
    v1 v2 = .,
    v2 v3 = cve23;
run;
```

You specify Model 2 by referring to Model 1 in the REFMODEL statement. Model 2 is the new model that refers to the old model, Model 1. This example illustrates the four types of model integration rules for the new model:

- **Duplication:** All parameter specifications, except for the partial covariance between v1 and v2 and the v3 ===> f1 path in the old model, are duplicated in the new model.
- **Addition:** The parameter cve23 for the partial covariance between v2 and v3 is added in the new model because there is no corresponding specification in the old model.
• Deletion: The specification of partial covariance between \( v_1 \) and \( v_2 \) in the old model is not copied into the new model, as indicated by the missing value ‘.’ specified in the new model.

• Replacement: The new path \( v_3 \leftarrow f_1 \) replaces the same path in the old model with parameter \( \text{load}1 \) for the path coefficient. Thus, in the new model paths \( v_3 \leftarrow f_1 \) and \( v_2 \leftarrow f_1 \) are now constrained to have the same path coefficient parameter \( \text{load}1 \).

**PATHDIAGRAM Statement**

```
PATHDIAGRAM < options > ;
```

You can use the PATHDIAGRAM statement to select particular path diagrams for output, to specify and modify the layout algorithm, to select variables and paths in path diagrams, to control the formatting of parameter labels, and to fine-tune many graphical and nongraphical features of path diagrams. You can use multiple PATHDIAGRAM statements to control different path diagrams or to produce path diagrams from the same model that have different styles and graphical features.

If you specify a PATHDIAGRAM statement without any options, this has the same effect as specifying the `PLOTS=PATHDIAGRAM` option in the PROC CALIS statement. Both produce a path diagram for the unstandardized solution if you do not specify the `METHOD=NONE` option in the PROC CALIS statement. By default, the path diagram for the unstandardized solution shows the paths, variables, unstandardized estimates and their significance, and fit summary table. For more information about these default settings, see the section “Default Path Diagram Settings” on page 1454. If you specify `METHOD=NONE` in the PROC CALIS statement, PROC CALIS produces a path diagram for the initial specifications, which shows the paths, the variables, and the input fixed values and parameter names. For introductory examples of creating and editing path diagrams, see Yung (2014).

The options of the PATHDIAGRAM statement can be classified into five categories. The following five tables summarize these options. An alphabetical listing of these options that includes more details follows the tables.

### Options for Selecting Different Types of Path Diagrams

You can use the options in the following table to request various types of path diagrams. If you use more than one of these options, PROC CALIS outputs a combination of the specified types of path diagrams.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIAGRAM=</td>
<td>Specifies the solution types for the output path diagrams</td>
</tr>
<tr>
<td>MODEL=</td>
<td>Specifies the models for the output path diagrams</td>
</tr>
<tr>
<td>STRUCTURAL</td>
<td>Requests the path diagrams for the structural components of models</td>
</tr>
</tbody>
</table>

### Options for Controlling the Layout of Paths and Variables

You can use the following options to specify the path diagram layout algorithm, to define paths and variables that you want to include in or exclude from path diagrams, and to provide useful information that might improve the graphical presentation of the output path diagrams.
### Option Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRANGE=</td>
<td>Specifies the graphical algorithm for laying out variables</td>
</tr>
<tr>
<td>DESTROYER=</td>
<td>Specifies paths that should not be considered in laying out variables</td>
</tr>
<tr>
<td>EMPHSTRUCT</td>
<td>Emphasizes the structural components in path diagrams</td>
</tr>
<tr>
<td>ERRORSIZE=</td>
<td>Specifies the size of error variables relative to observed variables</td>
</tr>
<tr>
<td>FACTORSIZE=</td>
<td>Specifies the size of latent factors relative to observed variables</td>
</tr>
<tr>
<td>MEANPARM=</td>
<td>Specifies mean parameters to be displayed as either paths or labels</td>
</tr>
<tr>
<td>OMITPATH=</td>
<td>Omits specified paths from the output path diagram</td>
</tr>
<tr>
<td>STRUCTADD=</td>
<td>Adds observed variables to the definition of the structural component</td>
</tr>
<tr>
<td>USEERROR</td>
<td>Requests the use of explicit error variables</td>
</tr>
<tr>
<td>VARPARM=</td>
<td>Specifies variance parameters to be displayed as either paths or labels</td>
</tr>
</tbody>
</table>

### Options for Selecting the Parameters to Display in the Output Path Diagram

You can use the following *options* to select the parameters to display in output path diagrams.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXOGCOV</td>
<td>Enables the display of covariances between exogenous non-error variables</td>
</tr>
<tr>
<td>NOCOV</td>
<td>Disables the display of covariance parameters</td>
</tr>
<tr>
<td>NOERRCOV</td>
<td>Disables the display of covariances between error variables</td>
</tr>
<tr>
<td>NOERRVAR</td>
<td>Disables the display of variances of error variables</td>
</tr>
<tr>
<td>NOEXOGCOV</td>
<td>Disables the display of covariances between exogenous non-error variables</td>
</tr>
<tr>
<td>NOEXOGVAR</td>
<td>Disables the display of variances between exogenous non-error variables</td>
</tr>
<tr>
<td>NOINITPARM</td>
<td>Disables the display of parameters in the path diagram for the initial specification</td>
</tr>
<tr>
<td>NOMEANS</td>
<td>Disables the display of means or intercepts of all variables</td>
</tr>
<tr>
<td>NOVARIANCE</td>
<td>Disables the display of variances of all variables</td>
</tr>
</tbody>
</table>

### Options for Formatting the Parameters in the Output Path Diagram

You can use the following *options* to control the formats of the parameters that are displayed in path diagrams.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECP=</td>
<td>Specifies number of the decimal places in the estimates</td>
</tr>
<tr>
<td>NOESTIM</td>
<td>Disables the display of numerical estimates</td>
</tr>
<tr>
<td>NOFLAG</td>
<td>Disables the display of flags for statistical significance</td>
</tr>
<tr>
<td>PARMNAMES</td>
<td>Enables the display of parameter names or labels</td>
</tr>
</tbody>
</table>

### Miscellaneous Options for Controlling the Output Path Diagram

You can use the following *options* to control the displays of the title, the fit table, and the labels of the variables.
### Option Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DECPFIT=</strong></td>
<td>Specifies the number of decimal places in the fit statistics</td>
</tr>
<tr>
<td><strong>DIAGRAMLABEL=</strong></td>
<td>Specifies the label of the diagram in the ODS</td>
</tr>
<tr>
<td><strong>FITINDEX=</strong></td>
<td>Selects fit information to display in the fit summary table</td>
</tr>
<tr>
<td><strong>LABEL=</strong></td>
<td>Specifies labels of the variables shown in the path diagram</td>
</tr>
<tr>
<td><strong>NOFITTABLE</strong></td>
<td>Suppresses the display of fit statistics</td>
</tr>
<tr>
<td><strong>NOTITLE</strong></td>
<td>Suppresses the display of the title</td>
</tr>
<tr>
<td><strong>TITLE=</strong></td>
<td>Specifies the title to display in the output path diagram</td>
</tr>
</tbody>
</table>

In order to customize your path diagram output effectively by using these options, it is useful to know the default settings of the graphical and nongraphical elements in the output path diagram. For information about these default settings and the corresponding overriding or modifying options, see the section “Default Path Diagram Settings” on page 1454.

### PATHDIAGRAM Statement Options

**ARRANGE=**\textit{name}

**ARRANGEMENT=**\textit{name}

**METHOD=**\textit{name}

specifies the algorithm for laying out the variables in the path diagram. You can specify the following names:

- **AUTOMATIC** requests the “best” algorithm. PROC CALIS analyzes the interrelationships among variables (excluding error variables) in the model and selects the most appropriate algorithm from those that you can explicitly request by specifying the FLOW, GRIP, and GROUPEDFLOW options. PROC CALIS first checks whether the ideal conditions for the process-flow algorithm are met; if so, the process-flow algorithm is used. If not, PROC CALIS checks whether the ideal conditions for the grouped-flow algorithm are met; if so, the grouped-flow algorithm is used. Otherwise, the more general GRIP algorithm is used.

- **FLOW** requests the process-flow algorithm, which is most appropriate when all variables (not including the error variables) exhibit hierarchical functional relationships in the model. That is, all variables in the model can be ordered hierarchically such that functional relationships (or directional paths) can occur only between variables at adjacent levels (that is, no cross-level, within-level, or reciprocal functional relationships). For example, the process-flow algorithm is ideal for confirmatory factor models and higher-order factor models in their pure forms (no cross-level or within-level paths in the model).

- **GRIP** requests the GRIP (Graph dRawing with Intelligent Placement) algorithm. This algorithm is more general than the process-flow and grouped-flow algorithms. Hence, the GRIP algorithm should be used when the ideal conditions for the process-flow and grouped-flow algorithms are not met.

- **GROUPEDFLOW** requests the grouped-flow algorithm, which is most appropriate when all latent factors exhibit hierarchical functional relationships in the model. That is, all latent factors can be ordered hierarchically such that their functional relationships (or directional paths) occur only between factors at adjacent levels (no cross-level,
within-level, or reciprocal functional relationships). The ideal conditions for the grouped-flow algorithm are the same as those for the process-flow algorithm, except that the grouped-flow algorithm considers only the latent factors that have an ideal process-flow pattern. Because each latent factor is usually clustered with a group of measured variables, this pattern can be described as an ideal “grouped” process-flow pattern (hence the name grouped-flow algorithm).

By default, ARRANGE=AUTOMATIC.

For more information and for illustrations of these methods, see the section “The Process-Flow, Grouped-Flow, and GRIP Layout Algorithms” on page 1440.

DECP=i

sets the decimal places of the estimates that are displayed in the path diagram, where i is between 0 and 4. The default value is 2. The displayed estimates are at most seven digits long, including the decimal point for the nonzero value of i.

DECPFIT=i

sets the decimal places of the fit statistics or information that is shown in the fit summary table of the path diagram, where i is between 0 and 4. The default value is 2. The displayed numerical values are at most 10 digits long, including the decimal point for the nonzero value of i.

DESTRUCTOR=[path <, path . . . >] | {path <, path . . . >}

DESTROYERPATH=[path <, path . . . >] | {path <, path . . . >}

specifies a list of paths that are considered to be “destroyer” paths to the layout algorithm that is used (or specified using the ARRANGE= option), where path represents

\[
\text{var-list direction var-list2}
\]

and direction is the direction of the path, as indicated by one of the following:

###

---

For example:

```
pathdiagram destroyer=[x1 ===> x2, x2 <=== x5];
pathdiagram destroyer=[x1 x5 ===> x2];
```

Note that the two preceding statements specify the same set of destroyer paths: “x1 ===> x2” and “x5 ===> x2.”

Destroyer paths are shown in the path diagram, but they are not used in determining the layout of variables. Destroyer paths are paths that violate the ideal conditions for a particular layout algorithm so that the placement of variables in the path diagram cannot take advantage of that algorithm (especially for the process-flow or grouped-flow algorithm). To counter the violations, PROC CALIS ignores the destroyer paths when laying out the variables. After determining the locations of all variables, PROC CALIS adds the destroyer paths back to the path diagram. If you can identify these destroyer paths and you have only a few of them (for example, fewer than five), specifying these destroyer paths in the DESTROYER= option can significantly improve the path diagram. However, if you have too many destroyer paths, this option might not be effective.
For more information and for illustrations, see the section “Handling Destroyers in Path Diagrams” on page 1448.

**NOTE:** If a path in the model serves as the only directed path that connects to a particular variable, it is generally not advisable to apply the DESTROYER= option to that path. During the layout process, the DESTROYER= option disconnects this variable from the rest of the variables in the model, so the location of this disconnected variable in the path diagram is arbitrary. This might lead to undesirable graphical results when PROC CALIS adds the destroyer paths back to the path diagram.

**DIAGRAM=** *name* | [ *names* ] | { *names* }  
**SOLUTION=** *name* | [ *names* ] | { *names* }  

specifies the solution types for the path diagram output. You can specify the following *names*:

**ALL**  
requests separate path diagrams for the initial, unstandardized, and standardized solutions.

**INITIAL | INIT**  
requests a path diagram for the initial solution. This diagram displays the fixed values and the parameters that you specify for the model. However, it does not display generated parameter names or initial estimates. To produce a “bare-bones” path diagram that shows only the variables and their interrelationships, use the NOINITPARM option, which suppresses the display of fixed values and the parameters.

**STANDARD | STAND**  
requests a path diagram for the standardized solution. This diagram displays the standardized parameter estimates and their significance in paths or as labels of variables. By default, it also displays the fit summary table.

**UNSTANDARD | UNSTAND**  
requests a path diagram for the unstandardized solution. This diagram displays the unstandardized parameter estimates and their significance in paths or as labels of variables. By default, it also displays the fit summary table.

For example, to display only the path diagram for the standardized solutions, you can use the following statement:

```
pathdiagram diagram=standard;
```

To display the path diagrams for the initial and unstandardized solution, you can use the following statement:

```
pathdiagram diagram=[initial unstandard];
```

**DIAGRAMLABEL=** *name*  
**DLABEL=** *name*  

specifies the label of the path diagram. You can use any valid SAS names or quoted strings up to 256 characters for *name*. However, only up to 40 characters of the label are used by ODS. If you do not specify this option, PROC CALIS uses the *name* provided in the TITLE= option. If you specify neither the DIAGRAMLABEL= nor TITLE= option, PROC CALIS generates a label for the path diagram. The generated label reflects the model number (if provided in the MODEL statement), the solution type (initial, unstandardized, or standardized), and whether the structural model is being shown. For example:
Note that if you specify multiple path diagrams in the same PATHDIAGRAM statement, PROC CALIS applies the same label to all requested path diagrams. If unique labels are preferred, you can use separate PATHDIAGRAM statements to specify labels for different path diagrams.

**EMPHSTRUCT< = i >**
requests that the structural component of the model be emphasized in the path diagram, where \( i \) is any number between 0.2 and 5. The variables in the structural component are called structural variables. Usually, only latent factors are considered as structural variables. For this option, the structural variables have relatively larger sizes than other variables (approximately four times as large as the observed variables in each dimension). You can control the relative size by providing a suitable value for \( i \), which is 4 by default. The corresponding path diagram displays and labels only the structural variables. Nonstructural variables are displayed but not labeled. In addition, the diagram displays and labels paths among structural variables. The diagram displays but does not label paths among structural and nonstructural variables. Finally, the diagram does not display or label paths among nonstructural variables.

If you consider some observed variables as structural variables in your model, use the **STRUCTADD=** option to include these observed variables in the structural component. This option is not applicable to the path diagram for the structural model that you request by specifying the **STRUCTURAL** option, which displays only the structural component of the model. In contrast, the **EMPHSTRUCT** option produces the complete model but emphasizes the structural component.

For more information and for illustrations, see the section “*Showing or Emphasizing the Structural Components*” on page 1456.

**ERRORSIZE=**
**ERRSIZE=**
specifies the size of error variables relative to that of observed variables, where \( size \) is between 0.2 and 5. The default value is 0.5, meaning that the size of error variables is about half that of observed variables.

**EXOGCOV**
**EXOGCOVARIANCE**
requests that the path diagram show the double-headed paths that represent the covariances among exogenous non-error variables. By default, these double-headed paths are displayed only for exploratory or confirmatory factor models, which you specify by using the **FACTOR** statement for other types of models.

**FACTORSIZE=**
**FACTSIZE=**
specifies the size of latent factors relative to that of observed variables, where \( size \) is between 0.2 and 5. The default value is 1.5, meaning that the size ratio of factors to observed variables is about 3 to 2.
**FITINDEX=**<code>[names] | {names}</code>
defines fit statistics or information in <code>names</code> to display in the fit summary table, which is shown along with the path diagrams for unstandardized and standardized solutions. PROC CALIS uses the order of the fit statistics or information specified in the FITINDEX= option to display the information in the fit summary table.

For example:

```plaintext
pathdiagram fitindex=[chisq df probchi srmr rmsea];
pathdiagram fitindex=all;
pathdiagram fitindex=[default aic sbc caic];
```

For the default list of fit statistics and information, see the FITINDEX=DEFAULT option.

You can use the following <code>names</code> to refer to all or individual fit statistics or information available in the fit summary table.

- **ALL** displays all available fit statistics or information. If you specify the ALL option along with other specific options for individual fit statistics or information, PROC CALIS displays the specific fit statistics or information in the fit summary table first, followed by the remaining available fit statistics or information.
- **DEFAULT** displays a default set of fit statistics or information, which is the same as specifying the following in the FITINDEX= option: AGFI, CFI, CHISQ, DF, LL_RMSEA, LOGLIKE, PROBCHI, PROBCLFIT, RMSEA, SRMR, and UL_RMSEA.
  
  If you specify the DEFAULT option along with other specific options for individual fit statistics or information, PROC CALIS displays the specific fit statistics or information in the fit summary table first, followed by the remaining available default fit statistics or information.

- **AGFI** displays the adjusted GFI.
- **AIC** displays the Akaike information criterion.
- **CAIC** displays Bozdogan’s corrected AIC.
- **CFI | BENTLERCFI** displays Bentler’s comparative fit index.
- **CHISQ** displays the goodness-of-fit index by Jöreskog and Sörbom.
- **CN | CRITICAL_N** displays Hoelter’s critical N.
- **DF** displays the degrees of freedom for the chi-square test for model fit.
- **ECVI** displays the expected cross-validation index.
- **GFI** displays the goodness-of-fit index by Jöreskog and Sörbom.
- **LL_ECVI | ECVI_LL** displays the lower confidence limit for RMSEA.
- **LL_RMSEA | RMSEA_LL** displays the lower confidence limit for RMSEA.
- **LOGLIKE** displays the fitted model –2 log-likelihood function value for METHOD=FIML only.
- **NIOBS** displays the number of incomplete observations for METHOD=FIML.
- **NOBS** displays the number of observations that are used in the analysis.
NPARGS | NPARM  displays the number of independent parameters.
PGFI      displays the parsimonious GFI.
PROBCHI   displays the $p$-value of the chi-square statistic for model fit.
PROBCLFIT displays the probability of close fit.
RMR       displays the root mean square residual.
RMSEA     displays the root mean square error of approximation.
SBC       displays the Schwarz Bayesian criterion.
SRMR      displays the standardized root mean square residual.
UL_ECVI | ECVI_UL  displays the upper confidence limit for ECVI.
UL_RMSEA | RMSEA_UL  displays the upper confidence limit for RMSEA.

LABEL= [varlabel <, varlabel . . .>] | {varlabel <, varlabel . . .>}
specifies the labels of variables to be displayed in path diagrams, where each varlabel is in the form

\textit{variable} = \textit{label}

You can use any valid SAS names or quoted strings up to 256 characters for labels. The labels are used to label the corresponding variables in output path diagrams. If you do not specify labels, the original variable names are used as labels.

For example, instead of using $x_1$ and $y_1$ to label the variables in the path diagram, the following statement specifies more meaningful labels:

\texttt{pathdiagram label=[x1="Start Use" y1="Spending"];
}

Note that PROC CALIS does not currently use the variable labels from the LABEL statement for the path diagram.

MEANPARM=PATH | LABEL
MEAN=PATH | LABEL
specifies whether mean parameters are displayed as paths (PATH) or as labels that are attached to variables (LABEL). The default MEANPARM= value is LABEL when you model mean structures. This option does not apply when you model only covariance structures.

MODEL=[int-list] | {int-list}
MODELS=[int-list] | {int-list}
requests path diagrams for a list of models, which are specified by their associated model numbers. By default, the output shows path diagrams of all models in the analysis. This option is useful if you want to restrict the path diagram output to a particular set of models. For example:

\texttt{pathdiagram model=[1 to 3];
pathdiagram model=[2 4 5];
}

The first PATHDIAGRAM statement requests path diagrams for models 1, 2, and 3. The second PATHDIAGRAM statement requests path diagrams for models 2, 4, and 5.
NOCOV
NOCOVARIANCE
  suppresses the display of covariances between variables.

NOERRCOV
NOERRORCOVARIANCE
  suppresses the default display of covariances among error variables.

NOERRVAR
NOERRORVARIANCE
  suppresses the default display of error variances, which are represented as either double-headed paths or labels that are attached to error variables.

NOESTIM
NOEST
  suppresses the default display of all numerical estimates (including fixed estimates) in path diagrams for unstandardized and standardized solutions.

NOEXOGCOV
NOEXOGCOVARIANCE
  suppresses the display of covariances between exogenous non-error variables. By default, only the exploratory or confirmatory factor models, which you specify using the FACTOR statement, show the covariances between exogenous non-error variables. For other models, NOEXOGCOV is the default.

NOEXOGVARIANCE
  suppresses the default display of variances of exogenous non-error variables. This applies to variance parameters that are represented as either double-headed paths or labels that are attached to the exogenous variables.

NOFITTABLE
NOFIT
  suppresses the default display of fit summary tables in path diagrams for standardized or unstandardized solutions.

NOFLAG
  suppresses the default flagging of significant estimates in path diagrams. By default, estimates that are significant at the 0.05 $\alpha$-level are flagged with “*”, and estimates that are also significant at the 0.01 $\alpha$-level are flagged with “**”. Fixed estimates are marked with “(fixed)”.

NOINITPARM
  suppresses the default display of user-specified parameter names and fixed values in path diagrams for initial specifications, which you request by specifying the DIAGRAM=INITIAL option in the PATHDIAGRAM statement. This option is not applicable to path diagrams for unstandardized or standardized solutions.
NOMEAN
suppresses the default display of the mean or intercept parameters and their estimates in models that contain mean structures. These mean parameters and estimates can be in the form of either paths or labels that are attached to variables.

NOTITLE
suppresses the display of the default title. You can use the TITLE= option to provide your own title.

NOVARIANCE
suppresses the default display of all variances. This applies to variance parameters that are represented as either double-headed paths or labels that are attached to nodes.

OMITPATHS=[\textit{path} < \textit{path} . . .] | \{\textit{path} <, \textit{path} . . .} \]
OMIT=[\textit{path} < \textit{path} . . .] | \{\textit{path} <, \textit{path} . . .} \]
specifies a list of \textit{paths} to be omitted from the output path diagram, where \textit{path} represents \textit{var-list direction var-list2}

and \textit{direction} is the direction of the path, as indicated by one of the following:


For example:

\begin{verbatim}
    pathdiagram omitpath=[y1 ===> y4, y4 <=== y3];
    pathdiagram omitpath=[y3 y1 ===> y4];
\end{verbatim}

Note that the two preceding statements specify the same set of paths that are omitted: “\textit{y1 ===> y4}” and “\textit{y3 ===> y4}.”

The omitted paths are not shown in the path diagram, nor are they used in determining the layout of the variables. The OMITPATHS= option is useful when you want to see how a particular set of paths affects the display of a path diagram. If omitting a certain set of paths improves the display, the omitted paths can be considered as destroyer paths. You might then specify these paths in the DESTROYER= option to get an improved diagram.

\textbf{NOTE:} If a path in the model serves as the only directed path that connects to a particular variable, it is generally not advisable to apply the OMITPATHS= option to that path. The OMITPATHS= option disconnects this variable from the rest of the variables in the model, so the location of the disconnected variable in the path diagram is arbitrary. This might lead to undesirable graphical results.

PARMNAMES

PARM
requests the display of parameter names or labels. By default, path diagrams for unstandardized or standardized solutions do not show any parameter names or labels, whereas path diagrams for initial specifications show only user-specified parameter names or labels (but not the generated parameter names or labels) and fixed values.
**Chapter 29: The CALIS Procedure**

**STRUCTADD=**{variables} | {variables}

specifies a list of observed variables that should be added to the structural component of a model. Traditionally, the structural component of a complete model includes only the latent factors and their interrelationships. However, this definition might be too restrictive in many applications. For the purpose of showing structural components in path diagrams, you can use the STRUCTADD= option to add observed variables to the definition of the structural component. The resulting path diagram for the structural component contains all latent factors, the additional observed variables, and their functional relationships.

For example, the following statement adds the observed variables x1 and x2 to the path diagram for displaying the structural component:

```
pathdiagram struct structadd=[x1 x2];
```

The following statement adds the observed variables x3 and x5 to the structural component for displaying the path diagram that emphasizes the structural component:

```
pathdiagram emphstruct structadd=[x3 x5];
```

For more information and for illustrations, see the section “Expanding the Definition of the Structural Variables” on page 1460.

**STRUCTURAL < (ONLY) >**

**STRUCT < (ONLY) >**

requests the path diagram for the so-called structural component of the model, or simply the component model. By default, the output shows this path diagram for the entire model, not just the structural component.

Traditionally, all structural equation models are considered to have two main components. One component is the so-called structural model, which involves latent factors and their functional relationships only. The other is the so-called measurement model, which involves observed variables and latent variables and the functional relationships that connect the observed variables to the latent factors. By specifying the STRUCTURAL option, you request the path diagram for the structural model in addition to the path diagram for the complete model. To display a path diagram for the structural model only, use the STRUCTURAL(ONLY) option.

For example, the following statement produces two path diagrams, one for the complete model and one for the structural component:

```
pathdiagram struct;
```

However, the following statement produces one path diagram for the structural component:

```
pathdiagram struct(only);
```

If you consider some observed variables as structural variables in your model, you can use the STRUCTADD= option to include these observed variables in the structural component.

For more information and for illustrations, see the section “Showing or Emphasizing the Structural Components” on page 1456.
**TITLE**=`name`

Specifies the title of the path diagram. You can use any valid SAS name or a quoted string of up to 256 characters for `name`. If you do not specify this option, PROC CALIS generates titles for path diagrams. The generated title reflects the model number (if provided in the MODEL statement), the model label (if provided in the MODEL statement), the solution type (initial, unstandardized, or standardized), and whether the structural model is being shown. For example:

```plaintext
pathdiagram title=ThisTitleDoesNotUseQuotations;
pathdiagram title="Title looks better with the use of quoted strings";
```

Note that if you specify multiple path diagrams in the same PATHDIAGRAM statement, PROC CALIS applies the same title to all requested path diagrams. If unique titles are preferred, you can use separate PATHDIAGRAM statements to specify titles for different path diagrams.

**USEERROR**

Requests that error variables be displayed in the path diagram. By default, PROC CALIS does not display error variables in path diagrams, because showing errors in path diagrams usually creates more clutter. If you prefer to show the error variables explicitly in path diagrams, specify this option.

**VARPARM=PATH | LABEL**

**VARIANCE=PATH | LABEL**

Specifies whether the variance parameters are displayed as paths (PATH) or as labels that are attached to variables (LABEL). The default VARPARM= value is PATH if your model does not fit mean structures. When you fit a model that contains mean structures, VARPARM= value is set to be the same as the MEANPARM= value.

---

**PCOV Statement**

PCOV assignment <, assignment . . . >;

Where assignment represents:

```plaintext
var-list < * var-list2> < = parameter-spec>
```

The PCOV statement is a subsidiary model specification statement for the PATH model. You can use the PCOV statement only with the PATH modeling language. The PCOV statement specifies the covariances of exogenous variables, or the error covariances of endogenous variables in the PATH model. It can also specify the covariance between an exogenous variable and the error term of an endogenous variables, although this usage is rare in practice.

In each assignment of the COV statement, you specify variables in the var-list and the var-list2 lists, followed by the covariance parameter specification in the parameter-spec list. The latter two specifications are optional. The syntax of the PCOV statement is the same as that of the COV statement. See the COV statement on page 1252 for details about specifying within- and between-list (partial) covariances.

The concept behind the PCOV statement is broader than that of the COV statement. The PCOV statement supports the partial covariance parameter specification in addition to the covariance parameter specification,
which is the only type of parameter that the COV statement supports. This difference is also reflected from the sets of var-list and var-list2 that you can use in the PCOV statement. In the COV statement, variables on the left-hand side of an assignment must be exogenous. However, in the PCOV statement, you can specify both exogenous and endogenous variables. If a pair of variables are both exogenous in a specification, you are defining a covariance parameter between the variables. If a pair of variables are both endogenous in a specification, you are defining a partial covariance parameter between of the variables. This partial covariance is usually interpreted as the error covariance between the two endogenous variables. If one variable is exogenous while the other is endogenous, you are defining a covariance parameter between the exogenous variable and the error term for the endogenous variable.

You can specify the following five types of the parameters for the partial covariances in the PCOV statement:

- an unnamed free parameter
- an initial value
- a fixed value
- a free parameter with a name provided
- a free parameter with a name and initial value provided

For example, consider a PATH model with exogenous variables $x_1$, $x_2$, and $x_3$ and endogenous variables $y_4$, $y_5$ and $y_6$. The following PCOV statement shows the five types of specifications in five assignments:

```
  pcov  x1  x2 ,  
  x1 x3 = (0.5),  
  x2 x3 = 2.0,  
  y4 y5 = psi1,  
  y5 y6 = psi2(0.4);  
```

In this statement, the covariance between $x_1$ and $x_2$ is specified as an unnamed free parameter. For this covariance, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1). The covariance between $x_1$ and $x_3$ is an unnamed free parameter with an initial value of 0.5. PROC CALIS also generates a parameter name for this covariance. The covariance between $x_2$ and $x_3$ is a fixed value of 2.0. This value stays the same during the estimation. The error covariance between endogenous variables $y_4$ and $y_5$ is a free parameter named psi1. The error covariance between endogenous variables $y_5$ and $y_6$ is a free parameter named psi2 with an initial value of 0.4.

The syntax of the PCOV statement is the same as the syntax of the COV statement. See the COV statement for more illustrations about the usage.

**Default Covariance Parameters**

Although the PCOV statement specification is conceptually broader than the COV statement specification, their related default set of covariance parameters is the same—that is, all covariances among exogenous manifest and latent variables (excluding error or disturbance variables) are free parameters. Because the PCOV statement applies only to the PATH model, it is easy to understand why the covariances do not apply to the error or disturbance terms. The PATH model, as implemented in PROC CALIS, simply does not use any explicit error or disturbance terms. For the default free covariance parameters, PROC CALIS generate the parameter names with the _Add prefix and appended with unique integer suffixes. You can also use the
PCOV statement specification to override these default covariance parameters in situations where you want to set parameter constraints, provide initial or fixed values, or make parameter references.

Another type of default partial covariances are fixed zeros. This default applies to the partial (error) covariances among all endogenous variables, and to the partial covariances between all exogenous variables and all endogenous variables in the path model. Again, you can override the default fixed values by providing explicit specification of these partial or error covariances in the PCOV statement.

### Modifying a Covariance or Partial Covariance Parameter Specification from a Reference Model

If you define a new PATH model by using a reference (old) model in the REFMODEL statement, you might want to modify some parameter specifications from the PCOV statement of the reference model before transferring the specifications to the new model. To change a particular partial covariance specification from the reference model, you can simply respecify the same covariance with the desired parameter specification in the PCOV statement of the new model. To delete a particular partial covariance parameter from the reference model, you can specify the desired partial covariance with a missing value specification in the PCOV statement of the new model.

For example, suppose that you are defining a new PATH model by using the REFMODEL statement and that the covariance between variables f1 and v2 is defined as a fixed or free parameter in the reference model. If you do not want this fixed parameter specification to be copied into your new model, you can use the following specification in the new model:

```
p cov f1 v2 = .;
```

Note that the missing value syntax is valid only when you use it with the REFMODEL statement. See the section “Modifying a PATH Model from a Reference Model” on page 1330 for a more detailed example of the PATH model respecification.

As discussed in the section “Default Covariance Parameters” on page 1344, PROC CALIS generates some default free covariance parameters for the PATH model if you do not specify them explicitly in the PCOV statement. When you use the REFMODEL statement for defining a reference model, these default free covariance parameters in the old (reference) model are not transferred to the new model. Instead, the new model generates its own set of default free covariance parameters after the new model is resolved from the reference model, the REFMODEL statement options, the RENAMEPARM statement, and the PCOV statement specifications in the new model. This also implies that if you want any of the (partial) covariance parameters to be constrained across the models by means of the REFMODEL specification, you must specify them explicitly in the PCOV statement of the reference model so that the same (partial) covariance specification is transferred to the new model.

---

### PVAR Statement

The PVAR statement specifies the variance or error (partial) variance parameters in connection with the confirmatory FACTOR and PATH models.
In each assignment of the PVAR statement, you list the var-list that you want to specify for their variances or error (partial) variances. Optionally, you can provide a list of parameter specifications (parameter-spec) after an equal sign for each var-list list. The syntax of the PVAR statement is exactly the same as that of the VARIANCE statement. See the VARIANCE statement on page 1362 for details about the syntax.

The concept behind the PVAR statement is broader than that of the VARIANCE statement. The PVAR statement supports the partial variance parameter specification in addition to the variance parameter specification, which is the only type of parameters that the VARIANCE statement supports. This difference is reflected from the set of var-list you can use in the PVAR statement. You can specify both exogenous variables and endogenous variables in the var-list list of the PVAR statement, but you can specify only exogenous variables in the var-list list of the VARIANCE statement. This conceptualization of the PVAR statement is needed in the FACTOR and PATH modeling languages because error variables are not explicitly defined in these models. You specify the variance of a variable if the variable in the var-list list of the PVAR statement is an exogenous (independent) variable in the FACTOR or PATH model. You specify the error (partial) variance of a variable if the variable in the var-list list of the PVAR statement is an endogenous (dependent) variable in the FACTOR or PATH model.

You can specify the following five types of the parameters for the partial variances in the PVAR statement:

- an unnamed free parameter
- an initial value
- a fixed value
- a free parameter with a name provided
- a free parameter with a name and initial value provided

For example, consider a PATH model with exogenous variables \(x_1, x_2,\) and \(x_3\) and endogenous variables \(y_4\) and \(y_5\). The following PVAR statement illustrates the five types of specifications in five assignments:

```plaintext
pvar x1 ,
   x2 = (2.0),
   x3 = 1.0,
   y4 = psi1,
   y5 = psi2(0.6);
```

In this statement, the variance of \(x_1\) is specified as an unnamed free parameter. For this variance, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1). The variance of \(x_2\) is an unnamed free parameter with an initial value of 2.0. PROC CALIS also generates a parameter name for this variance. The variance of \(x_3\) is a fixed value of 1.0. This value stays the same during the estimation. The error variance of endogenous variable \(y_4\) is a free parameter named psi1. The error variance of endogenous variable \(y_5\) is a free parameter named psi2 with an initial value of 0.6.

The syntax of the PVAR statement is the same as the syntax of the VARIANCE statement. See the VARIANCE statement for more illustrations about the usage.
Default Partial Variance Parameters

By default, all variances of the *exogenous* manifest and latent variables and all error (partial) variances of the *endogenous* manifest and latent variables are free parameters in the FACTOR or PATH model. For these default free variance parameters, PROC CALIS generates the parameter names with the \_Add prefix and appended with unique integer suffixes. You can also use the PVAR statement specification to override these default variance parameters in situations where you want to specify parameter constraints, provide initial or fixed values, or make parameter references.

In the FACTOR or PATH model, a variable can either be exogenous or endogenous. Therefore, the default free parameters covers all the possible variance or partial variance parameters in the model. There are no default fixed zeros for any variances or partial variances in the model.

Modifying a Variance or Partial Variance Parameter Specification from a Reference Model

If you define a new FACTOR or PATH model by using a reference (old) model in the REFMODEL statement, you might want to modify some parameter specifications from the PVAR statement of the reference model before transferring the specifications to the new model. To change a particular variance or partial variance specification from the reference model, you can simply respecify the same variance or partial variance with the desired parameter specification in the PVAR statement of the new model. To delete a particular variance parameter from the reference model, you can specify the desired variance or partial variance with a missing value specification in the PVAR statement of the new model.

For example, suppose that the variance of V1 is specified in the reference PATH model but you do not want this variance specification to be transferred to the new model. You can use the following PVAR statement specification in the new model:

```
  pvar
  v2 = .;
```

Note that the missing value syntax is valid only when you use the REFMODEL statement. See the section “Modifying a FACTOR Model from a Reference Model” on page 1266 for a more detailed example of FACTOR model respecification. See the section “Modifying a PATH Model from a Reference Model” on page 1330 for a more detailed example of PATH model respecification.

As discussed the section “Default Partial Variance Parameters” on page 1347, PROC CALIS generates default free variance parameters for the exogenous variables and default free error variance parameters for the endogenous variables in the confirmatory FACTOR or PATH model. When you use the REFMODEL statement for defining a reference model, these default free variance parameters in the old (reference) model are not transferred to the new model. Instead, the new model generates its own set of default free variance parameters after the new model is resolved from the reference model, the REFMODEL statement options, the RENAMEPARM statement, and the PVAR statement specifications in the new model. If you want any of the variance or error (partial) variance parameters to be constrained across the models by means of the REFMODEL specification, you must specify them explicitly in the PVAR statement of the reference model so that the same variance or error (partial) variance specification is transferred to the new model.
**RAM Statement**

\[
\text{RAM} < \text{VAR=} \text{variable-list} \mid [ \text{variable-list=} \text{number-list} <, \text{variable-list=} \text{number-list} \ldots > ], > < \text{ram-entry} <, \text{ram-entry} \ldots > > ;
\]

where \text{variable-list} is a list of variables for the rows and columns of the \_A\_ and \_P\_ matrices and the rows of the \_W\_ vector of the RAM model, \text{number-list} is a list of positive integers that denote the order of the specified variables, and \text{ram-entry} is a parameter specification for an element in one of the three RAM model matrices. You can specify latent variables in addition to observed variables in the \text{VAR=} option.

RAM stands for the reticular action model developed by McArdle (1980). The RAM model implemented in PROC CALIS extends the original RAM model with the specification of the mean vector in the \_W\_ vector. See the section “The RAM Model” on page 1425 for details about the model.

The RAM statement specification consists of the list of the variables in the model and the parameters and their locations the RAM model matrices. For example, consider the following simple RAM model specification:

\[
\text{ram var=} \ x1-x2 \ y3, \\
\_A\_ \ 3 \ 1, \\
\_A\_ \ 3 \ 2;
\]

In this statement, variables \(x1\), \(x2\), and \(y3\) are specified in the \text{VAR=} option. The variable order in the \text{VAR=} option is important. The same variable order applies to the rows and columns of the \_A\_ matrix. Next, there are two \text{ram-entries}. The first \text{ram-entry} specifies that the third variable \(y3\) has a path from the first variable \(x1\). Similarly, the second \text{ram-entry} specifies that \(y3\) has a path from \(x2\).

**Specifying the \text{VAR=} Option**

In the \text{VAR=} option, you specify the list of observed and latent variables in the RAM model. There are two ways to specify the \text{VAR=} list. The first way is a simple listing of variables. For example, you specify a total of 18 variables in the RAM model in the following statement:

\[
\text{ram var=} \ a \ b \ c \ x1-x5 \ y1-y10;
\]

The order of the variables in this \text{VAR=} list is important. The same variable order applies to the rows and columns of the \_A\_ and \_P\_ matrices and the rows of the \_W\_ matrices. Although it is not required to arrange the variables according to whether they are observed or latent in the \text{VAR=} list, you can do so for your own convenience. PROC CALIS checks each variable in the \text{VAR=} list against the associated data sets to determine whether the variable is observed or latent.

When you specify the parameters in the \text{ram-entries}, you represent variables by the variable numbers that refer to the \text{VAR=} list. Therefore, it is important to make correct association of the variables and their order on the \text{VAR=} list. To this end, you can add some comments in your \text{VAR=} list to make the variable numbers explicit. For example,

\[
\text{ram var=} \ a \ /* \ 1 */ \\
\ b \ /* \ 2 */ \\
\ c \ /* \ 3 */ \\
\ x1-x5 \ /* \ 4-8 */ \\
\ y1-y10 \ /* \ 9-18 */;
\]
Another way to specify VAR= list is to provide the variable-lists together with explicit ordering indicated in the number-lists. For example, in the following statement you specify exactly the same variable list as that in the preceding example:

```
ram var= [x1-x5 = 4 to 8, c = 3, y1-y10 = 9 to 18, a = 1, b = 2];
```

Apart from showing how you can construct the VAR= list in a very general way with the number-lists, there is no particular reason why the variable-lists in the preceding specification are not in either an ascending or a descending order. Perhaps a more natural and useful way to use this type of explicit ordering specification is to place variables in exactly the same order as intended. For example, the following VAR= specification serves as a “key” of the variable numbers in the subsequent ram-entries:

```
ram var= [x1 = 1, x2 = 2, y1 = 3, y2 = 4, y3 = 5],
      _A_ 1 2 ,
      _P_ 2 2 ;
```

With reference to the explicit variable numbers in the VAR= list, you can interpret the _A_[1, 2] specification immediately as the effect from x2 to x1, and the _P_[2, 2] specification as the variance of x2.

If the VAR= option is not specified in the RAM statement, the n observed variables in the VAR statement are used as the first n variables in the VAR= list. If you specify neither the VAR= option in the RAM statement nor the VAR statement, all n numerical variables in the associated data sets serve as the first n variables in the RAM model matrices. If there are more than n variables used in the ram-entries, the extra variables are all treated as latent variables in the RAM model.

Latent variables generated by PROC CALIS for the RAM model are named in two different ways, depending on whether your RAM model is specified under a MODEL statement. If you do not use the MODEL statement (for example, in situations with single-group analyses), latent variables are named _Factor1, _Factor2, and so on. If your RAM model is defined within the scope of a MODEL statement, latent variables are named _Mdlk_F1, _Mdlk_F2, and so on, where k is substituted with the model number that is specified in the MODEL statement. For example, _Mdl2_F1 is a latent factor that is specified under a RAM model within the scope of the MODEL statement with 2 as its model number.

Because data sets might contain nonnumerical variables, implicit variable ordering deduced from the data sets is sometimes obscured. Therefore, it is highly recommended that you use the VAR= option to list all the variables in the RAM model.

**Specifying a ram-entry**

```
matrix-name | matrix-number row-number column-number < parameter-spec>
```

A ram-entry is a parameter specification of a matrix element of the RAM model. In each ram-entry, you first specify the matrix by using either the matrix-name or the matrix-number. Then you specify the row-number and the column-number of the element of the matrix. At the end of the ram-entry, you can optionally specify various kinds of parameters in parameter-spec. You can specify as many ram-entries as needed in your RAM model. Ram-entries are separated by commas. For example, consider the following specification:

```
ram var= x1-x2 y3,
      _A_ 3 1 1.,
      _A_ 3 2;
```

You specify three variables in the VAR= option of the RAM statement. In the first ram-entry, variable y3 has a path from variable x1 with a fixed path coefficient 1. In the second ram-entry, variable y3 has a path from
variable $x_2$. Because the \textit{parameter-spec} is blank, the corresponding path coefficient (or the effect from $x_2$ on $y_3$) is a free parameter by default.

**Specifying the matrix-name or matrix-number**

The three model matrices in the RAM model are: $A$, $P$, and $W$. See the section “The RAM Model” on page 1425 for the mathematical formulation of the RAM model. The \textit{matrix-name} or \textit{matrix-number} specifications in the \textit{ram-entries} refer to these model matrices. You can use the following keywords for \textit{matrix-name} or \textit{matrix-number}:

- $A$, _RAMA_, or 1 for the elements in the $A$ matrix, which is for path coefficients or effects
- $P$, _RAMP_, or 2 for the elements in the $P$ matrix, which is for variances and covariances
- $W$, _RAMW_, or 3 for the elements in the $W$ vector, which is for intercepts and means

**Specifying the row-number and column-number**

After you specify the \textit{matrix-name} or \textit{matrix-number} in a \textit{ram-entry}, you need to specify the \textit{row-number} and \textit{column-number} that correspond to the intended element of the matrix being specified.

**Specifying the parameter-spec**

You can specify three types of parameters in \textit{parameter-spec}:

- A free parameter without an initial estimate: blank or \textit{parameter-name}

You can specify a free parameter for the matrix element in a \textit{ram-entry} by either omitting the \textit{parameter-spec} (that is, leaving it blank) or specifying a \textit{parameter-name}. For example, both of the following \textit{ram-entries} specify that $A_{3,1}$ is a free parameter in the RAM model:

\[
_A\ 3\ 1
\]

and

\[
_A\ 3\ 1\ beta
\]

The difference is that in the latter you name the effect (path coefficient) for the $A_{3,1}$ element as \textit{beta}, while in the former PROC CALIS generates a free parameter name (prefixed with \_Parm and followed by a unique parameter number) for the specified element. Leaving the \textit{parameter-spec} blank is handy if you do not need to refer to this parameter in your code. But when you need to specify parameter constraints by referring to parameter names, the \textit{parameter-name} syntax becomes necessary. For example, the following specification constrains the $A_{3,1}$ and $A_{3,2}$ paths to have equal effects (path coefficients) because they have the same \textit{parameter-name} \textit{beta}:

\[
\text{ram var= x1-x2 y3,}
_A\ 3\ 1\ beta,
_A\ 3\ 2\ beta;
\]
• A free parameter with an initial estimate: \((\text{number})\) or \(\text{parameter-name} (\text{number})\)

You can specify a free parameter with an initial estimate in a \textit{ram-entry} by either specifying the initial estimate within parentheses or specifying a \textit{parameter-name} followed by the parenthesized initial estimate. For example, both of the following \textit{ram-entries} specify that \(_A_[3,1]\) is a free parameter with an initial estimate of 0.3 in the RAM model:

\[
_A_ \ 3 \ 1 \ (0.3)
\]

and

\[
_A_ \ 3 \ 1 \ \text{beta} \ (0.3)
\]

In the latter you name the effect (path coefficient) for the \(_A_[3,1]\) element as beta, while in the former PROC CALIS generates a free parameter name (prefixed with \_Parm and followed by a unique parameter number). The latter syntax is necessary when you need to specify parameter constraints by referring to the parameter name beta. The former syntax is more convenient when you do not need to refer to this parameter in other specifications.

For the latter syntax with a \textit{parameter-name} specified, you can omit the pair of parentheses or exchange the position of \textit{parameter-name} and \textit{number} (or both) without changing the nature of the parameter. That is, you can use the following equivalent specifications for a named free parameter with initial values:

\[
_A_ \ 3 \ 1 \ \text{beta} \ 0.3
\]

and

\[
_A_ \ 3 \ 1 \ .3 \ \text{beta}
\]

• A fixed parameter value: \textit{number}

You can specify a fixed value by simply providing it as the \textit{parameter-spec} in a \textit{ram-entry}. For example, in the following syntax you specify that \(_A_[3,1]\) is a fixed value of 0.3:

\[
_A_ \ 3 \ 1 \ 0.3
\]

The fixed value for \(_A_[3,1]\) does not change during the estimation. To distinguish this syntax from the initial value specification, notice that you do not put \(0.3\) inside parentheses, nor do you put a \textit{parameter-name} before or after the provided value.
Notes and Cautions about Specifying \textit{ram-entries}

- Older versions of PROC CALIS treat a blank \textit{parameter-spec} in the \textit{ram-entry} as a fixed constant 1. This is no longer the case in this version of PROC CALIS. Fixed values such as 1.0 must be specified explicitly.

- The \textit{row-number} and \textit{column-number} in the \textit{ram-entries} refer to the VAR= variable list of the RAM statement. An exception is for the \_W\_ vector, of which the \textit{column-number} should always be 1 and does not refer to any particular variable.

- When a \textit{row-number} or \textit{column-number} in a \textit{ram-entry} (except for the \textit{column-number} of \_W\_) does not have any reference in the VAR= variable list (or is greater than the number of default observed variables when the VAR= option is not specified), PROC CALIS treats the corresponding row or column variable as a latent variable and generates variable names for it.

- The largest row or column number used in any \textit{ram-entry} should not exceed the sum of observed and latent variables intended in the RAM model. Otherwise, some extraneous latent variables might be created.

\section*{Default Parameters}

It is important to understand the default parameters in the RAM model. First, if you know which parameters are default free parameters, you can make your specification more efficient by omitting the specifications of those parameters that can be set by default. For example, because all exogenous variances and error variances in the RAM model are free parameters by default, you do not need to specify the diagonal elements of the \_P\_ matrix if they are not constrained in the model. Second, if you know which parameters are default free parameters, you can specify your model accurately. For example, because all the error covariances in the RAM model are fixed zeros by default, you must specify the corresponding off-diagonal elements of the \_P\_ matrix in the \textit{ram-entries}. See the section “Default Parameters in the RAM Model” on page 1432 for details about the default parameters of the RAM model.

\section*{Modifying a RAM Model from a Reference Model}

This section assumes that you use a \texttt{REFMODEL} statement within the scope of a \texttt{MODEL} statement and that the reference model (or base model) is also a RAM model. The reference model is called the old model, and the model that refers to this old model is called the new model. If the new model is not intended to be an exact copy of the old model, you can use the following extended RAM modeling language to make modifications on the model specification. The syntax for modifications is very much the same as the ordinary RAM modeling language (see the section “RAM Statement” on page 1348), except that you cannot specify the VAR= option in the RAM statement. The reason is that the VAR= variable list in the new RAM model should be exactly the same as the old model; otherwise, the \textit{row-number} and \textit{column-number} in the \textit{ram-entries} would not have the same references and thus would make model referencing meaningless. Hence, the syntax for respecifying (modifying) the RAM model contains only the \textit{ram-entries}:

\begin{verbatim}
RAM \textit{ram-entry} <, \textit{ram-entry} \ldots > ;
\end{verbatim}

The syntax of the \textit{ram-entry} is the same as that of the original RAM statement, with an addition of the missing value specification for the \textit{parameter-spec}, which denotes the deletion of a parameter location.

The new model is formed by integrating with the old model in the following ways:
Duplication: If you do not specify in the new model a parameter location (matrix element) that exists in the old model, the old parameter specification is duplicated in the new model.

Addition: If you specify in the new model a parameter location (matrix element) that does not exist in the old model, the new parameter specification is added to the new model.

Deletion: If you specify in the new model a parameter location (matrix element) that also exists in the old model and the new parameter-spec is denoted by the missing value ‘.’, the old parameter specification is not copied into the new model.

Replacement: If you specify in the new model a parameter location (matrix element) that also exists in the old model and the new parameter is not denoted by the missing value ‘.’, the new parameter specification replaces the old one in the new model.

For example, consider the following two-group analysis:

```plaintext
proc calis;
  group 1 / data=d1;
  group 2 / data=d2;
  model 1 / group=1;
    ram
      var = [V1-V6 = 1 to 6, F1 = 7],
      _A_ 1 7 1.,
      _A_ 2 7 load1,
      _A_ 3 7 load2,
      _A_ 7 4 ,
      _A_ 7 5 ,
      _A_ 7 6 ,
      _P_ 1 1 ,
      _P_ 2 2 ,
      _P_ 3 3 ,
      _P_ 7 7 ,
      _P_ 4 4 ,
      _P_ 5 5 ,
      _P_ 6 6 ,
      _P_ 1 2 cve12;
    model 2 / group=2;
    refmodel 1;
    ram
      _A_ 3 7 load1,
      _P_ 1 2 .,
      _P_ 2 3 cve23;
run;
```

In this example, you specify Model 2 by referring to Model 1 in the REFMODEL statement. Model 2 is the new model which refers to the old model, Model 1. This example illustrates the four types of model integration process by PROC CALIS:

- Duplication: All parameter specifications, except for _A_[3,7] and _P_[1,2], in the old model are duplicated in the new model.
- Addition: The new parameter cve23 is added for the matrix element _P_[2,3] in the new model.
- Deletion: The parameter location _P_[1,2] and associated parameter cve12 are not copied into the new model, as indicated by the missing value ‘.’ in the new model specification.
• Replacement: The $A_{[3,7]}$ path in the new model replaces the same path in the old model with another parameter for the path coefficient. As a result, in the new model paths specified by $A_{[3,7]}$ and $A_{[2,7]}$ are constrained to have the same path coefficient parameter load1.

PROC CALIS might have generated some default parameters (named with the ‘_Add’ prefix) for the old (reference) model. These default parameters in the old (reference) model do not transfer to the new model. Only after the new model is resolved from the reference model, the REFMODEL statement options, the RENAMEPARM statement, and the model respecification are the default parameters of the new RAM model generated. In this way, the generated parameters in the new model are not constrained to be the same as the corresponding parameters in the old (reference) model. If you want any of these default parameters to be constrained across the models, you must specify them explicitly in the ram-entries of the RAM statement of the reference model so that these specifications are duplicated to the new model via the REFMODEL statement.

**REFMODEL Statement**

```
REFMODEL model-number < / options > ;
```

The REFMODEL statement is not a modeling language itself. It is a tool for referencing and modifying models. It is classified into one of the modeling languages because its role is similar to other modeling languages.

```
REFMODEL model-number < / options > ;
  RENAMEPARM parameter renaming ;
  main model specification statement ;
  subsidiary model specification statements ;
```

In the REFMODEL statement, you specify the `model-number` (between 1 and 9,999, inclusive) of the model you are making reference to. The reference model must be well-defined in the same PROC CALIS run. In the `options`, you can rename all the parameters in the reference model by adding a prefix or suffix so that the current model has a new set of parameters. The RENAMEPARM statement renames individual parameters in the reference model to new names. In the `main model specification statement` and the `subsidiary model specification statements`, you can respecify or modify the specific parts of the reference model. The specification of these statements must be compatible with the model type of the reference model.

**NOTE:** The REFMODEL statement does not simply copy model specifications from a reference model. If you do not change any of the parameter names of the reference model by any of the REFMODEL statement options, the REFMODEL statement copies only the explicit specifications from the reference model to the new model. However, the REFMODEL statement does not copy the default parameters from the reference model to the new model. For example, consider the following statements:

```
proc calis;
  group 1 / data=a1;
  group 2 / data=a2;
  model 1 / group=1;
    path x1 ===> x2;
  model 2 / group=2;
    refmodel 1;
run;
```
In this example, Model 2 makes reference to Model 1. This means that the path relationship between \(x_1\) and \(x_2\) as specified in Model 1 is exactly the same path relationship you want Model 2 to have. The path coefficients in these two models are constrained to be the same. However, the variance parameter of \(x_1\) and the error variance parameter for \(x_2\) are not constrained in these models. Rather, these two parameters are set by default in these models separately. If you intend to constrain all parameters in the two models, you can specify all the parameters in Model 1 explicitly and use the \texttt{REFMODEL} statement for Model 2, as shown in the following statements:

```sas
proc calis;
  group 1 / data=a1;
  group 2 / data=a2;
  model 1 / group=1;
    path x1 ===> x2;
    pvar x1 x2;
  model 2 / group=2;
    refmodel 1;
run;
```

This way Model 2 makes reference to all the explicitly specified parameters in Model 1. Hence the two models are completely constrained. However, a simpler way to fit exactly the same model to two groups is to use a single model definition, as shown in the following statements:

```sas
proc calis;
  group 1 / data=a1;
  group 2 / data=a2;
  model 1 / group=1,2;
    path x1 ===> x2;
run;
```

This specification has the same estimation results as those for the preceding specification.

When you also use one of the \texttt{REFMODEL} statement options, the \texttt{REFMODEL} statement is no longer a simple copy of explicit parameter specifications from the reference model. All parameters are renamed in the new model in the model referencing process. The following \texttt{options} are available in the \texttt{REFMODEL} statement:

- **\texttt{ALLNEWPARMS}**
  - Appends to the parameter names in the reference model with \_\_mdl and then an integer suffix denoting the model number of the current model. For example, if \texttt{qq} is a parameter in the reference model for a current model with model number 3, then this option creates \texttt{qq mdl3} as a new parameter name.

- **\texttt{PARM\_PREFIX=prefix}**
  - Inserts to all parameter names in the reference model with the \texttt{prefix} provided. For example, if \texttt{qq} is a parameter in the reference model for a current model, then \texttt{PARM\_PREFIX=pre} creates \texttt{pre qq} as a new parameter name.

- **\texttt{PARM\_SUFFIX=suffix}**
  - Appends to all parameter names in the reference model with the \texttt{suffix} provided. For example, if \texttt{qq} is a parameter in the reference model for a current model, then \texttt{PARM\_SUFFIX=_suf} creates \texttt{qq suf} as a new parameter name.

Instead of renaming all parameters, you can also rename parameters individually by using the \texttt{RENAMEPARM} statement within the scope of the \texttt{REFMODEL} statement.
You can also add the main and subsidiary model specification statements to modify a particular part from the reference model. For example, you might like to add or delete some equations or paths, or to change a fixed parameter to a free parameter or vice versa in the new model. All can be done in the respecification in the main and subsidiary model specification statements within the scope of the `MODEL` statement to which the `REFMODEL` statement belongs. Naturally, the modeling language used in respecification must be the same as that of the reference model. See the individual statements for modeling languages for the syntax of respecification. Note that when you respecify models by using the main and subsidiary model specification statements together with the `RENAMEPARM` statement or the `REFMODEL` options for changing parameter names, the parameter name changes occur after respecifications.

**RENAMEPARM Statement**

```
RENAMEPARM assignment <, assignment ... > ;
```

where `assignment` represents:

```
old_parameters = parameter-spec
```

You can use the `RENAMEPARM` statement to rename parameters or to change the types of parameters of a reference model so that new parameters are transferred to the new model in question. The `RENAMEPARM` statement is a subsidiary model specification statement that should be used together with the `REFMODEL` statement. The syntax of the `RENAMEPARM` statement is similar to that of the `VARIANCE` statement—except that in the `RENAMEPARM` statement, you put parameter names on the left-hand side of equal signs, whereas you put variable names on the left-hand side in the `VARIANCE` statement. You can use no more than one `RENAMEPARM` statement within the scope of each `REFMODEL` statement.

In the `REFMODEL` statement, you transfer all the model specification information from a base model to the new model being specified. The `RENAMEPARM` statement enables you to modify the parameter names or types in the base model before transferring them to the new model. For example, in the following example, you define Model 2, which is a new model, by referring it to Model 1, the base model, in the `REFMODEL` statement.

```
model 1;
  lineqs
    V1 = F1 + E1,
    V2 = b2 F1 + E2,
    V3 = b3 F1 + E3,
    V4 = b4 F1 + E4;
  variance F1 = vF1,
    E1-E4 = ve1-ve4;

model 2;
  refmodel 1;
  renameparm ve1-ve4=new1-new4, b2=newb2(.2), b4=.3;
```

Basically, the `LINEQS` model specification in Model 1 is transferred to Model 2. In addition, you redefine some parameters in the base model by using the `RENAMEPARM` statement. This example illustrates two kinds of modifications that the `RENAMEPARM` statement can do:
• creating new parameters in the new model

The error variances for E1–E4 in Model 2 are different from those defined in Model 1 because new parameters new1–new4 are now used. Parameter b2 is renamed as newb2 with a starting value at 0.2 in Model 2. So the two models have distinct path coefficients for the F1-to-V2 path.

• changing free parameters into fixed constants

By using the specification b4= .3 in the RENAMEPARM statement, b4 is no longer a free parameter in Model 2. The path coefficient for the F1-to-V4 path in Model 2 is now fixed at 0.3.

The RENAMEPARM statement is handy when you have just few parameters to change in the reference model defined by the REFMODEL statement. However, when there are a lot of parameters to modify, the RENAMEPARM statement might not be very efficient. For example, to make all parameters unique to the current model, you might consider using the ALLNEWPARMS, PARM_PREFIX=, or PARM_SUFFIX= option in the REFMODEL statement.

SAS Programming Statements

You can use SAS programming statements to define dependent parameters, parametric functions, and equality constraints among parameters.

Several statistical procedures support the use of SAS programming statements. The syntax of SAS programming statements are common to all these procedures and can be found in the section “Programming Statements” on page 511 in Chapter 19, “Shared Concepts and Topics.”

SIMTESTS Statement

SIMTESTS | SIMTEST sim-test < sim-test . . . > ;

where sim-test represents one of the following:

- test-name = [ functions ]
- test-name = { functions }

and functions are either parameters in the model or parametric functions computed in the SAS programming statements.

When the estimates in a model are asymptotically multivariate-normal, continuous and differentiable functions of the estimates are also multivariate-normally distributed. In the SIMTESTS statement, you can test these parametric functions simultaneously. The null hypothesis for the simultaneous tests is assumed to have the following form:

\[ H_0 : h_1(\theta) = 0, h_2(\theta) = 0, \ldots \]

where \( \theta \) is the set of model parameters (independent or dependent) in the analysis and each \( h_i() \) is a continuous and differentiable function of the model parameters.

To test parametric functions simultaneously in the SIMTESTS statement, you first assign a name for the simultaneously test in test-name. Then you put the parametric functions for the simultaneous test inside
a pair of parentheses: either the ‘{’ and ‘}’ pair, or the ‘[’ and ‘]’ pair. For example, if $\theta_1$, $\theta_2$, $\theta_3$, and $\theta_4$ are parameters in the model and you want to test the equality of $\theta_1$ and $\theta_2$ and the equality of $\theta_3$ and $\theta_4$ simultaneously, you can use the following statements:

```sas
simtests
   Equality_test = [t1_t2_diff t3_t4_diff];
   t1_t2_diff = theta1 - theta2;
   t3_t4_diff = theta3 - theta4;
```

In the SIMTESTS statement, you test two functions $t1_t2_diff$ and $t3_t4_diff$ simultaneously in the test named `Equality_test`. The two parametric functions $t1_t2_diff$ and $t3_t4_diff$ are computed in the SAS programming statements as differences of some parameters in the model.

See also the `TESTFUNC` statement on page 1358 for testing parametric functions individually.

---

**STD Statement**

```sas
STD assignment < , assignment . . . > ;
```

where `assignment` represents:

```
    var-list = parameter-spec
```

The STD statement functions exactly the same as the VARIANCE statement. The STD statement is obsolete and might not be supported in future versions of PROC CALIS. Use the VARIANCE statement instead.

---

**STRUCTEQ Statement**

```sas
STRUCTEQ variables < / label > ;
```

where `label` represents:

```
    LABEL | NAME = name
```

The STRUCTEQ statement functions exactly the same as the DETERM statement.

---

**TESTFUNC Statement**

```sas
TESTFUNC functions ;
```

where `functions` are either parameters in the model or parametric functions computed in the SAS programming statements.

When the estimates in a model are asymptotically multivariate-normal, any continuous and differentiable function of the estimates is also normally distributed. In the TESTFUNC statement, you can test these parametric functions using z-tests. The form of the null hypothesis is as follows:

$H_0 : h(\theta) = 0$

where $\theta$ is the set of model parameters (independent or dependent) in the analysis and $h()$ is a continuous and differentiable function of the model parameters.
For example, if \( \theta_1 \), \( \theta_2 \), and \( \theta_3 \) are parameters in the model, and you want to test whether \( \theta_1 \) and \( \theta_2 \) are the same and whether \( \theta_3 \) is the same as the average of \( \theta_1 \) and \( \theta_2 \), you can use the following statements:

\[
\begin{align*}
\text{testfunc} & \quad t1\_t2\_diff \quad t3\_t1t2\_diff; \\
t1\_t2\_diff & \quad = \theta_1 - \theta_2; \\
t3\_t1t2\_diff & \quad = \theta_3 - (\theta_1 + \theta_2)/2;
\end{align*}
\]

In the TESTFUNC statement, you test two functions: \( t1\_t2\_diff \) and \( t3\_t1t2\_diff \). These two functions are defined in the SAS programming statements that follow after the TESTFUNC statement. Thus, \( t1\_t2\_diff \) represents the difference between \( \theta_1 \) and \( \theta_2 \), and \( t3\_t1t2\_diff \) represents the difference between \( \theta_3 \) and the average of \( \theta_1 \) and \( \theta_2 \).

See the SIMTESTS statement if you want to test several null hypotheses simultaneously.
limit the set of the observed variables for analysis. The reason is simply that there is no other options in the FACTOR statement that will serve the same purpose. For example, you analyze only v1–v3 in the following exploratory factor model even though there might be more observed variables available in the data set:

```plaintext
proc calis;
   var v1-v3;
   factor n=1;
```

If you do not specify the VAR statement, PROC CALIS simply selects all numerical variables for analysis. However, to avoid confusions it is a good practice to specify the observed variables explicitly in the VAR statement.

Case 2. Models With a VAR= or Similar Option for Defining the Set of Observed Variables for Analysis. The classes of models considered here are: COSAN, LISMOD, MSTRUCT, and RAM. Except for the LISMOD models, in all other three classes of models you can specify the observed variables in the model by using the a VAR= option in the respective main model specification statement. For the LISMOD models, you can specify all observed variables that should be included in the model in the XVAR= and YVAR= options of the LISMOD statement. Therefore, the use of the VAR statement for these models might become unnecessary. For example, the following MSTRUCT statement specifies the observed variables v1–v6 in the VAR= option:

```plaintext
proc calis;
   mstruct var=v1-v6;
```

It would have been redundant to use a VAR statement to specify v1–v6 additionally. The same conclusion applies to the COSAN and the RAM models.

Another example is when you specify a LISMOD model. In the following LISMOD specification, variables v1–v8 would be the set of observed variables for analysis:

```plaintext
proc calis;
   var v1-v8;
   lismod xvar = v1-v4,
         yvar = v5-v8,
         eta = factor1,
         xi = factor2;
```

Again, there is no need to add a VAR statement merely repeating the specification of variables v1–v8.

If you do specify the VAR statement in addition to the specification of variable lists in these models, PROC CALIS will check the consistency between the lists. Conflicts arise if the two lists do not match.

For example, the following statements will generate an error in model specification because v6 specified in the MSTRUCT model is not defined as an observed variable available for analysis in the VAR statement (even if v6 might indeed be present in the data set):

```plaintext
proc calis;
   var v1-v5;
   mstruct var=v1-v6;
```

So it is an error when you specify fewer observed variables in the VAR statement than in the VAR= option in the model. How about if you specify more variables in the VAR statement? PROC CALIS will also general an error because the extra variables in VAR statement will not be well-defined in the model. For example, v7–v10 specified in the VAR statement are supposed to be included into the model, but they not listed on either the XVAR= or YVAR= list in the following LISMOD statement:
proc calis;
  var v1-v10;
  lismod xvar = v1-v3,
         yvar = v4-v6,
         eta = factor1,
         xi = factor2;

Therefore, if you must specify the VAR statement for these models, the specifications of the observed variables must be consistent in the VAR statement and in the relevant model options. However, to avoid potential conflicts in these situations, you are recommended to specify the observed variables in the VAR=, XVAR=, or YVAR= lists only.

When the VAR= option is not specified in the COSAN, MSTRUCT, or RAM statement, the VAR statement specification will be used as the list of observed variables in the model. If both of the VAR= option and VAR statement specification are lacking, then all numerical variables in the associated data set will be used in the model. However, to avoid confusions the preferred method is to specify the list of observed variables explicitly on the VAR=, XVAR=, or YVAR= option of the main model specification statements.

Case 3. Models With Indirect Ways to Include the Set of Observed Variables for Analysis. Two types of models are considered here: LINEQS and PATH. For these models, the main use of the VAR statement is to include those observed variables that are not mentioned in model specifications.

For example, in the following statements for a LINEQS model variable v3 is not mentioned in the LINEQS statement:

```sas
proc calis;
  var v1-v3;
  lineqs v1 = a1 * v2 + e1;
```

With the specification in the VAR statement, however, variable v3 is included into the model as an exogenous manifest variable. Similarly, the same applies to the following PATH model specification:

```sas
proc calis;
  var v1-v3;
  path v1 <== v2;
```

Again, variable v3 is included into the PATH model because it is specified in the VAR statement.

The two preceding examples also suggest that you do not need to use the VAR statement when your already mentions all observed variables in the model specification. For example, if your target set of observed variables are v1–v3, the use of the VAR statement in the following specification is unnecessary:

```sas
proc calis;
  var v1-v3;
  path v1 <== v2;
  pvar v3;
```

For the two types of models considered here, you can also use the VAR statement to define and limit the set of observed variables for analysis. For example, you might have v1, v2, v3 in your data set as observed variables for analysis; but somehow in your model v2 should be treated as a latent variable. You might use the following code to exclude v2 as an observed variable in the model:
proc calis;
  var v1 v3;
  path v1 <= v2;
  pvar v3;

The role of the VAR statement here is to define and limit the set of observed variables available for the model. Hence, only variables v1 and v3 are supposed to be observed variables in the model while variable v2 in the PATH model is treated as latent.

In sum, in the current situation the use of the VAR statement should depend on whether a variable should or should not be included as an observed variable in your theoretical model.

**Case 4. Confirmatory Factor Model With the FACTOR statement.** In this case, the VAR statement still limits the set of observed variables being analyzed in the confirmatory factor model. However, because all observed variables in a confirmatory factor analysis must be loaded on (or related to) some factors through the specification of factor-variable-relations in the FACTOR statement, all observed variables in the model should have been specified (or mentioned) in the FACTOR statement already, making it redundant to use the VAR statement for the same purpose.

---

**VARIANCE Statement**

```
VARIANCE assignment <, assignment ...>
```

where `assignment` represents:

```
var-list < = parameter-spec >
```

The VARIANCE statement specifies the variance parameters in connection with the LINEQS model. Notice that the VARIANCE statement is different from the VAR statement, which specifies variables for analysis. In previous versions of PROC CALIS, the STD statement name was used instead of the VARIANCE statement name. Although these two names result in the same functionalities, the VARIANCE statement name reflects the intended usages better.

In the LINEQS model, variance parameters are defined only for *exogenous* manifest and latent variables (including error and disturbance variables) in the model. Therefore, you cannot list any *endogenous* variables in the `var-list` list of the VARIANCE statement. You can specify no more than one VARIANCE statement for each LINEQS model.

In each `assignment` of the VARIANCE statement, you list the `var-list` whose variances you want to specify. Optionally, you can provide a list of parameter specifications (`parameter-spec`) after an equal sign for each `var-list` list.

You can specify the following five types of the parameters for the variances of the exogenous variables in the VARIANCE statement:

- an unnamed free parameter
- an initial value
- a fixed value
• a free parameter with a name provided
• a free parameter with a name and initial value provided

Consider a LINEQS model with exogenous variables \( V_1, V_2, F_1, D_2, \) and \( E_3 \). The following VARIANCE statement illustrates the five types of parameter specifications in five assignments:

\[
\text{variance} \\
V_1, \\
V_2 = (.5), \\
F_1 = 1.0, \\
D_2 = \text{dvar}, \\
E_3 = \text{evar}(0.7); \\
\]

In this statement, the variance of \( V_1 \) is specified as an unnamed free parameter. For this variance, PROC CALIS generates a parameter name with the \(_\text{Parm}\) prefix and appended with a unique integer (for example, \(_\text{Parm1}\)). The variance of \( V_2 \) is an unnamed free parameter with an initial value of 0.5. PROC CALIS also generates a parameter name for this variance. The variance of \( F_1 \) is a fixed value of 1.0. This value stays the same during the estimation. The variance of \( D_2 \) is a free parameter named \( \text{dvar} \). The variance of \( E_3 \) is a free parameter named \( \text{evar} \) with an initial value of 0.7.

When you need to specify a long parameter name list, you can consider using the prefix-name specification for the parameter list. For example, the following statement specifies 100 unique parameter names for the variances of \( E_1 \sim E_{100} \):

\[
\text{variance} \\
E_1\sim E_{100} = 100 * \text{evar}__; /* \text{evar} with two trailing underscores */ \\
\]

In the VARIANCE statement, \( \text{evar}__ \) is a prefix-name with the root \( \text{evar} \). The notation \( 100* \) means this prefix-name is applied 100 times, resulting in a generation of the 100 unique parameter names \( \text{evar}001, \text{evar}002, \ldots, \text{evar}100 \).

The root of the prefix-name should have few characters so that the generated parameter name is not longer than 32 characters. To avoid unintentional equality constraints, the prefix-names should not coincide with other parameter names.

**Mixed Parameter Lists**

You can specify different types of parameters for the list of variances. For example, the following statement uses a list of parameters with mixed types:

\[
\text{variance} \\
E_1\sim E_6 = \text{vp}1 \text{ vp}2(2.0) \text{ vp}3 4. (.3) \text{ vp}6(.4); \\
\]

This is equivalent to the following specification:

\[
\text{variance} \\
E_1 = \text{vp}1 \\
E_2 = \text{vp}2(2.0), \\
E_3 = \text{vp}3, \\
E_4 = 4., \\
E_5 = (.3), \\
E_6 = \text{vp}6(.4); \\
\]
Notice that an initial value followed after a parameter name is associated with the free parameter. For example, in the original mixed list specification, the specification \(2.0\) after \(vp2\) is interpreted as the initial value for the parameter \(vp2\), but not as the initial estimate for the variance of \(E3\).

However, if you indeed want to specify that \(vp2\) is a free parameter without an initial value and \(2.0\) is an initial estimate for the variance of \(E3\) (while keeping all other things the same), you can use a null initial value specification for the parameter \(vp2\), as shown in the following statement:

\[
\text{variance} \\
E1-E6 = vp1 \ vp2() (2.0) \ 4. \ (.3) \ vp6(.4);
\]

This way \(2.0\) becomes the initial estimate for the variance of \(E3\). Because a parameter list with mixed types might be confusing, you can break down the specifications into separate assignments to remove ambiguities. For example, you can use the following equivalent specification:

\[
\text{variance} \\
E1 = vp1 \\
E2 = vp2, \\
E3 = (2.), \\
E4 = 4., \\
E5 = (.3), \\
E6 = vp6(.4);
\]

**Shorter and Longer Parameter Lists**

If you provide fewer parameters than the number of variances in the \(\text{var-list}\) list, all the remaining parameters are treated as unnamed free parameters. For example, the following specification assigns a fixed value of 1.0 to the variance of \(F1\) while treating the other three variances as unnamed free parameters:

\[
\text{variance} \\
F1-F4 = 1.0;
\]

This specification is equivalent to the following specification:

\[
\text{variance} \\
F1 = 1.0, \ F2-F4;
\]

If you intend to fill up all values with the last parameter specification in the list, you can use the continuation syntax \([\ldots]\), \([\ldots]\), or \([\ldots]\), as shown in the following example:

\[
\text{variance} \\
E1-E100 = 1.0 \ psi \ [\ldots];
\]

This means that the variance of \(E1\) is fixed at 1.0, while the variances of \(E1-E100\) are all free parameter named \(psi\). All variances except that for \(E1\) are thus constrained to be equal by using the same parameter name.

However, you must be careful not to provide too many parameters. For example, the following specification results in an error:

\[
\text{variance} \\
E1-E6 = 1.0 \ psi2-psi6 \ extra;
\]

The parameters after \(psi6\) are excessive.
Default Variance Parameters

In the LINEQS model, by default all variances of exogenous manifest and latent variables (including error and disturbance variables) are free parameters. For these default free parameters, PROC CALIS generates the parameter names with the _Add prefix and appended with unique integer suffixes. You can also use the VARIANCE statement specification to override these default variance parameters in situations where you want to specify parameter constraints, provide initial or fixed values, or make parameter references.

Because only exogenous variables can have variance parameters in the LINEQS model and all these exogenous variances are free parameters by default, there are no default fixed zeros for any variances in the LINEQS model.

Modifying a Variance Parameter Specification from a Reference Model

If you define a new LINEQS model by using a reference (old) model in the REFMODEL statement, you might want to modify some parameter specifications from the VARIANCE statement of the reference model before transferring the specifications to the new model. To change a particular variance specification from the reference model, you can simply respecify the same variance with the desired parameter specification in the VARIANCE statement of the new model. To delete a particular variance parameter from the reference model, you can specify the desired variance with a missing value specification in the VARIANCE statement of the new model.

For example, suppose that the variance of V1 is specified in the reference model but you do not want this variance specification to be transferred to the new model, you can use the following VARIANCE statement specification in the new model:

```
variance V1 = .;
```

Note that the missing value syntax is valid only when you use the REFMODEL statement. See the section “Modifying a LINEQS Model from a Reference Model” on page 1280 for a more detailed example of the LINEQS model respecification.

As discussed in a preceding section, PROC CALIS generates default free variance parameters for the LINEQS model if you do not specify them explicitly in the VARIANCE statement. When you use the REFMODEL statement for defining a reference model, these default free variance parameters in the old (reference) model are not transferred to the new model. Instead, the new model generates its own set of default free variance parameters after the new model is resolved from the reference model, the REFMODEL statement options, the RENAMEPARM statement, and the VARIANCE statement specifications in the new model. This also implies that if you want any of the variance parameters to be constrained across the models by means of the REFMODEL specification, you must specify them explicitly in the VARIANCE statement of the reference model so that the same variance specification is transferred to the new model.
Chapter 29: The CALIS Procedure

VARNAMES Statement

VARNAMES name-assignment <, name-assignment . . . > ;
VARNAMES name-assignment <, name-assignment . . . > ;
VNAMES name-assignment <, name-assignment . . . > ;

where name-assignment represents one of the following forms:

matrix-name variable-names
matrix-name = [ variable-names ]
matrix-name = matrix-name

You can use the VARNAMES statement in connection with the COSAN modeling language to assign variable names for matrices. The matrix-name refers to any matrix you define in the COSAN statement. The variable-names that follow the matrix-name are assigned to the column variables of the matrix of interest. This applies to the first two types of VARNAMES specifications. For example,

varnames F f1-f3;

is exactly the same as

varnames F = [ f1-f3 ];

Both of these assign f1, f2, and f3 as the names for the first three column variables of matrix F.

You can also use another kind of name-assignment in connection with a COSAN statement. Two matrix names equated by an equal sign assign the column names of the matrix on the right-hand side to the column names of the matrix on the left-hand side. This assignment assumes that the column names of at least one of the two matrices are already defined. For example, assuming that J and A are model matrices defined in a COSAN statement, the following VARNAMES statement specification specifies that both J and A have the same set of column variable names V1–V6 and F1–F3:

varnames J = [ V1-V6 F1-F3 ] ,
A = J ;

This is the same as the following specification:

varnames J = [ V1-V6 F1-F3 ] ,
A = [ V1-V6 F1-F3 ] ;

The VARNAMES statement appears to enable you to specify only the column variable names for matrices. However, PROC CALIS also uses these column variable names to assign row variable names of the related matrices in the covariance and mean structure formulas for the COSAN model. PROC CALIS uses the following rules to determine the row variable names of a matrix in the model:

• If a matrix is the first matrix of any term in the covariance or mean structure formula, the row variable names are the names of the manifest variables.

• If a matrix is the central covariance matrix of any term in the covariance structure formula, the row variable names are the same as the column variable names.
For any other matrices, the row variable names are the same as the column variable names of the preceding matrix in the multiplicative formula for the covariance or mean structures.

**WEIGHT Statement**

```
WEIGHT variable;
```

The WEIGHT statement specifies the weight variable for the observations. It is one of the subsidiary group specification statements. You can use the WEIGHT statement no more than once within the scope of each GROUP statement or the PROC CALIS statement.

Weighting is often done when the error variance associated with each observation is different and the values of the weight variable are proportional to the reciprocals of the variances. The WEIGHT and FREQ statements have a similar effect, except the WEIGHT statement does not alter the number of observations unless VARDEF=WGT or VARDEF=WDF. An observation is used in the analysis only if the WEIGHT variable is greater than 0 and is not missing.

**Details: CALIS Procedure**

**Input Data Sets**

You can use four different kinds of input data sets in the CALIS procedure, and you can use them simultaneously. The DATA= data set contains the data to be analyzed, and it can be an ordinary SAS data set containing raw data or a special TYPE=COV, TYPE=UCOV, TYPE=CORR, TYPE=UCORR, TYPE=SSCP, or TYPE=FACTOR data set containing previously computed statistics. The INEST= data set specifies an input data set that contains initial estimates for the parameters used in the optimization process, and it can also contain boundary and general linear constraints on the parameters. If the model does not change too much, you can use an OUTEST= data set from a previous PROC CALIS analysis; the initial estimates are taken from the values of the _TYPE_=PARMS observation. The INMODEL= or INRAM= data set contains information of the analysis models (except for user-written programming statements). Often the INMODEL= data set is created as the OUTMODEL= data set from a previous PROC CALIS analysis. See the section “OUTMODEL= or OUTRAM= Data Set” on page 1374 for the structure of both OUTMODEL= and INMODEL= data sets. Using the INWGT= data set enables you to read in the weight matrix \( W \) that can be used in generalized least squares, weighted least squares, or diagonally weighted least squares estimation.

**BASEFIT= or INBASEFIT= Data Set**

The BASEFIT= or INBASEFIT= data set saves the fit function value and the degrees of freedom of a baseline model for computing various fit indices, especially the incremental fit indices. Typically, the BASEFIT= data set is created as an OUTFIT= data set from a previous PROC CALIS fitting of a customized baseline model. See the section “OUTFIT= Data Set” on page 1385 for details about the format of the OUTFIT= and BASEFIT= data sets.
**DATA= Data Set**

A TYPE=COV, TYPE=UCOV, TYPE=CORR, or TYPE=UCORR data set can be created by the CORR procedure or various other procedures. It contains means, standard deviations, the sample size, the covariance or correlation matrix, and possibly other statistics depending on which procedure is used.

If your data set has many observations and you plan to run PROC CALIS several times, you can save computer time by first creating a TYPE=COV, TYPE=UCOV, TYPE=CORR, or TYPE=UCORR data set and using it as input to PROC CALIS.

For example, assuming that PROC CALIS is first run with an OUTMODEL=MODEL option, you can run the following statements in subsequent analyses with the same model in the first run:

```plaintext
/* create TYPE=COV data set */
proc corr cov nocorr data=raw outp=cov(type=cov);
run;
/* analysis using correlations */
proc calis corr data=cov inmodel=model;
run;
/* analysis using covariances */
proc calis data=cov inmodel=model;
run;
```

Most procedures automatically set the TYPE= option of an output data set appropriately. However, the CORR procedure sets TYPE=CORR unless an explicit TYPE= option is used. Thus, (TYPE=COV) is needed in the preceding PROC CORR request, since the output data set is a covariance matrix. If you use a DATA step with a SET statement to modify this data set, you must declare the TYPE=COV, TYPE=UCOV, TYPE=CORR, or TYPE=UCORR attribute in the new data set.

You can use a VAR statement with PROC CALIS when reading a TYPE=COV, TYPE=UCOV, TYPE=CORR, TYPE=UCORR, or TYPE=SSCP data set to select a subset of the variables or change the order of the variables.

**CAUTION:** Problems can arise from using the CORR procedure when there are missing data. By default, PROC CORR computes each covariance or correlation from all observations that have values present for the pair of variables involved ("pairwise deletion"). The resulting covariance or correlation matrix can have negative eigenvalues. A correlation or covariance matrix with negative eigenvalues is recognized as a singular matrix in PROC CALIS, and you cannot compute (default) generalized least squares or maximum likelihood estimates. You can specify the RIDGE option to ridge the diagonal of such a matrix to obtain a positive definite data matrix. If the NOMISS option is used with the CORR procedure, observations with any missing values are completely omitted from the calculations ("listwise deletion"), and there is no possibility of negative eigenvalues (but there is still a chance for a singular matrix).

PROC CALIS can also create a TYPE=COV, TYPE=UCOV, TYPE=CORR, or TYPE=UCORR data set that includes all the information needed for repeated analyses.

If the data set DATA=RAW does not contain missing values, the following statements should give the same PROC CALIS results as the previous example:

```plaintext
/* using correlations */
proc calis corr data=raw outstat=cov inmodel=model;
run;
/* using covariances */
proc calis data=cov inmodel=model;
run;
```
You can create a TYPE=COV, TYPE=UCOV, TYPE=CORR, TYPE=UCORR, or TYPE=SSCP data set in a DATA step. Be sure to specify the TYPE= option in parentheses after the data set name in the DATA statement and include the _TYPE_ and _NAME_ variables. If you want to analyze the covariance matrix but your DATA= data set is a TYPE=CORR or TYPE=UCORR data set, you should include an observation with _TYPE_ =STD giving the standard deviation of each variable. By default, PROC CALIS analyzes the recomputed covariance matrix even when a TYPE=CORR data set is provided, as shown in the following statements:

```sas
data correl(type=corr);
  input _type_ $ _name_ $ X1-X3;
  datalines;
  std . 4. 2. 8.
  corr X1 1.0 . .
  corr X2 .7 1.0 .
  corr X3 .5 .4 1.0
;  
proc calis inmodel=model;
run;
```

INEST= Data Set

You can use the INEST= (or INVAR=) input data set to specify the initial values of the parameters used in the optimization and to specify boundary constraints and the more general linear constraints that can be imposed on these parameters.

The variables of the INEST= data set must correspond to the following:

- a character variable _TYPE_ that indicates the type of the observation
- \( n \) numeric variables with the parameter names used in the specified PROC CALIS model
- the BY variables that are used in a DATA= input data set
- a numeric variable _RHS_ (right-hand side); needed only if linear constraints are used
- additional variables with names corresponding to constants used in the programming statements

The content of the _TYPE_ variable defines the meaning of the observation of the INEST= data set. PROC CALIS recognizes observations with the following _TYPE_ specifications.

**PARMS** specifies initial values for parameters that are defined in the model statements of PROC CALIS. The _RHS_ variable is not used. Additional variables can contain the values of constants that are referred to in programming statements. At the beginning of each run of PROC CALIS, the values of the constants are read from the PARMS observation for initializing the constants in the SAS programming statements.

**UPPERBD | UB** specifies upper bounds with nonmissing values. The use of a missing value indicates that no upper bound is specified for the parameter. The _RHS_ variable is not used.

**LOWERBD | LB** specifies lower bounds with nonmissing values. The use of a missing value indicates that no lower bound is specified for the parameter. The _RHS_ variable is not used.
specifies the linear constraint $\sum_j a_{ij} x_j \leq b_i$. The $n$ parameter values contain the coefficients $a_{ij}$, and the _RHS_ variable contains the right-hand-side $b_i$. The use of a missing value indicates a zero coefficient $a_{ij}$.

GE | >= | > specifies the linear constraint $\sum_j a_{ij} x_j \geq b_i$. The $n$ parameter values contain the coefficients $a_{ij}$, and the _RHS_ variable contains the right-hand-side $b_i$. The use of a missing value indicates a zero coefficient $a_{ij}$.

EQ | = specifies the linear constraint $\sum_j a_{ij} x_j = b_i$. The $n$ parameter values contain the coefficients $a_{ij}$, and the _RHS_ variable contains the right-hand-side $b_i$. The use of a missing value indicates a zero coefficient $a_{ij}$.

The constraints specified in the INEST=, INVAR=, or ESTDATA= data set are added to the constraints specified in BOUNDS and LINCON statements.

You can use an OUTEST= data set from a PROC CALIS run as an INEST= data set in a new run. However, be aware that the OUTEST= data set also contains the boundary and general linear constraints specified in the previous run of PROC CALIS. When you are using this OUTEST= data set without changes as an INEST= data set, PROC CALIS adds the constraints from the data set to the constraints specified by a BOUNDS and LINCON statement. Although PROC CALIS automatically eliminates multiple identical constraints, you should avoid specifying the same constraint a second time.

INMODEL= or INRAM= Data Set

This data set is usually created in a previous run of PROC CALIS. It is useful if you want to reanalyze a problem in a different way such as using a different estimation method. You can alter an existing OUTMODEL= data set in the DATA step to create the INMODEL= data set that describes a modified model. See the section “OUTMODEL= or OUTRAM= Data Set” on page 1374 for more details about the INMODEL= data set.

INWGT= Data Set

This data set enables you to specify a weight matrix other than the default matrix for the generalized, weighted, and diagonally weighted least squares estimation methods. If you also specify the INWGTINV option (or use the INWGT(INV)=option), the INWGT= data set is assumed to contain the inverse of the weight matrix, rather than the weight matrix itself. The specification of any INWGT= data set for unweighted least squares or maximum likelihood estimation is ignored. For generalized and diagonally weighted least squares estimation, the INWGT= data set must contain a _TYPE_ and a _NAME_ variable as well as the manifest variables used in the analysis. The value of the _NAME_ variable indicates the row index $i$ of the weight $w_{ij}$. For weighted least squares, the INWGT= data set must contain _TYPE_, _NAME_, _NAM2_, and _NAM3_ variables as well as the manifest variables used in the analysis. The values of the _NAME_, _NAM2_, and _NAM3_ variables indicate the three indices $i, j, k$ of the weight $w_{ij,k}$. You can store information other than the weight matrix in the INWGT= data set, but only observations with _TYPE_=WEIGHT are used to specify the weight matrix W. This property enables you to store more than one weight matrix in the INWGT= data set. You can then run PROC CALIS with each of the weight matrices by changing only the _TYPE_ observation in the INWGT= data set with an intermediate DATA step.

See the section “OUTWGT= Data Set” on page 1385 for more details about the INWGT= data set.
Output Data Sets

OUTEST= Data Set

The OUTEST= (or OUTVAR=) data set is of TYPE=EST and contains the final parameter estimates, the gradient, the Hessian, and boundary and linear constraints. For METHOD=ML (with or without the ROBUST option), METHOD=FIML, METHOD=GLS, and METHOD=WLS, the OUTEST= data set also contains the approximate standard errors, the information matrix (crossproduct Jacobian), and the approximate covariance matrix of the parameter estimates ((generalized) inverse of the information matrix). If there are linear or nonlinear equality or active inequality constraints at the solution, the OUTEST= data set also contains Lagrange multipliers, the projected Hessian matrix, and the Hessian matrix of the Lagrange function.

The OUTEST= data set can be used to save the results of an optimization by PROC CALIS for another analysis with either PROC CALIS or another SAS procedure. Saving results to an OUTEST= data set is advised for expensive applications that cannot be repeated without considerable effort.

The OUTEST= data set contains the BY variables, two character variables _TYPE_ and _NAME_, t numeric variables corresponding to the parameters used in the model, a numeric variable _RHS_ (right-hand side) that is used for the right-hand-side value \( b_i \) of a linear constraint or for the value \( f = f(x) \) of the objective function at the final point \( x^* \) of the parameter space, and a numeric variable _ITER_ that is set to zero for initial values, set to the iteration number for the OUTITER output, and set to missing for the result output.

The _TYPE_ observations in Table 29.7 are available in the OUTEST= data set, depending on the request.

### Table 29.7 _TYPE_ Observations in the OUTEST= Data Set

<table>
<thead>
<tr>
<th><em>TYPE</em></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTBC</td>
<td>If there are active boundary constraints at the solution ( x^* ), three observations indicate which of the parameters are actively constrained, as follows:</td>
</tr>
<tr>
<td>GE</td>
<td>indicates the active lower bounds</td>
</tr>
<tr>
<td>LE</td>
<td>indicates the active upper bounds</td>
</tr>
<tr>
<td>EQ</td>
<td>indicates the active masks</td>
</tr>
<tr>
<td>COV</td>
<td>Contains the approximate covariance matrix of the parameter estimates; used in computing the approximate standard errors.</td>
</tr>
<tr>
<td>COVRANK</td>
<td>contains the rank of the covariance matrix of the parameter estimates.</td>
</tr>
<tr>
<td>CRPJ_LF</td>
<td>Contains the Hessian matrix of the Lagrange function (based on CRPJAC).</td>
</tr>
<tr>
<td>CRPJAC</td>
<td>Contains the approximate Hessian matrix used in the optimization process. This is the inverse of the information matrix.</td>
</tr>
</tbody>
</table>
### Table 29.7 continued

<table>
<thead>
<tr>
<th><em>TYPE</em></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>If linear constraints are used, this observation contains the $i$th linear constraint $\sum_j a_{ij} x_j = b_i$. The parameter variables contain the coefficients $a_{ij}$, $j = 1, \ldots, n$, the <em>RHS</em> variable contains $b_i$, and <em>NAME</em>=ACTLC or <em>NAME</em>=LDACTLC.</td>
</tr>
<tr>
<td>GE</td>
<td>If linear constraints are used, this observation contains the $i$th linear constraint $\sum_j a_{ij} x_j \geq b_i$. The parameter variables contain the coefficients $a_{ij}$, $j = 1, \ldots, n$, and the <em>RHS</em> variable contains $b_i$. If the constraint $i$ is active at the solution $x^*$, then <em>NAME</em>=ACTLC or <em>NAME</em>=LDACTLC.</td>
</tr>
<tr>
<td>GRAD</td>
<td>Contains the gradient of the estimates.</td>
</tr>
<tr>
<td>GRAD_LF</td>
<td>Contains the gradient of the Lagrange function. The <em>RHS</em> variable contains the value of the Lagrange function.</td>
</tr>
<tr>
<td>HESSIAN</td>
<td>Contains the Hessian matrix.</td>
</tr>
<tr>
<td>HESS_LF</td>
<td>Contains the Hessian matrix of the Lagrange function (based on HESSIAN).</td>
</tr>
<tr>
<td>INFORMAT</td>
<td>Contains the information matrix of the parameter estimates (only for METHOD=ML, METHOD=GLS, or METHOD=WLS).</td>
</tr>
<tr>
<td>INITGRAD</td>
<td>Contains the gradient of the starting estimates.</td>
</tr>
<tr>
<td>INITIAL</td>
<td>Contains the starting values of the parameter estimates.</td>
</tr>
<tr>
<td>JACNLC</td>
<td>Contains the Jacobian of the nonlinear constraints evaluated at the final estimates.</td>
</tr>
<tr>
<td>LAGM BC</td>
<td>Contains Lagrange multipliers for masks and active boundary constraints.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><em>NAME</em></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE</td>
<td>Indicates the active lower bounds</td>
</tr>
<tr>
<td>LE</td>
<td>Indicates the active upper bounds</td>
</tr>
<tr>
<td>EQ</td>
<td>Indicates the active masks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><em>NAME</em></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEC_NUM</td>
<td>Number of the linear equality constraint</td>
</tr>
<tr>
<td>LEC_VAL</td>
<td>Corresponding Lagrange multiplier value</td>
</tr>
<tr>
<td>LIC_NUM</td>
<td>Number of the linear inequality constraint</td>
</tr>
<tr>
<td>LIC_VAL</td>
<td>Corresponding Lagrange multiplier value</td>
</tr>
</tbody>
</table>
**Table 29.7 continued**

<table>
<thead>
<tr>
<th>TYPE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAGM NLC</td>
<td>contains Lagrange multipliers for nonlinear equality and active inequality constraints in pairs of observations that contain the constraint number and the value of the Lagrange multiplier.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLEC_NUM</td>
<td>Number of the nonlinear equality constraint</td>
</tr>
<tr>
<td>NLEC_VAL</td>
<td>Corresponding Lagrange multiplier value</td>
</tr>
<tr>
<td>NLIC_NUM</td>
<td>Number of the linear inequality constraint</td>
</tr>
<tr>
<td>NLIC_VAL</td>
<td>Corresponding Lagrange multiplier value</td>
</tr>
</tbody>
</table>

LE
If linear constraints are used, this observation contains the \(i\)th linear constraint \(\sum_j a_{ij} x_j \leq b_i\). The parameter variables contain the coefficients \(a_{ij}, \ j = 1, \ldots, n\), and the \_RHS\_ variable contains \(b_i\). If the constraint \(i\) is active at the solution \(x^*\), then \_NAME\_=ACTLC or \_NAME\_=LDACTLC.

LOWERBD
If boundary constraints are used, this observation contains the lower bounds. Those parameters not subjected to lower bounds contain missing values. The \_RHS\_ variable contains a missing value, and the \_NAME\_ variable is blank.

NACTBC
All parameter variables contain the number \(n_{abc}\) of active boundary constraints at the solution \(x^*\). The \_RHS\_ variable contains a missing value, and the \_NAME\_ variable is blank.

NACTLC
All parameter variables contain the number \(n_{alc}\) of active linear constraints at the solution \(x^*\) that are recognized as linearly independent. The \_RHS\_ variable contains a missing value, and the \_NAME\_ variable is blank.

NLC_EQ
Contains values and residuals of nonlinear constraints. The \_NAME\_ variable is described as follows:

<table>
<thead>
<tr>
<th>NAME</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLC</td>
<td>Inactive nonlinear constraint</td>
</tr>
<tr>
<td>NLC_ACT</td>
<td>Linear independent active nonlinear constraint</td>
</tr>
<tr>
<td>NLC_ACTLD</td>
<td>Linear dependent active nonlinear constraint</td>
</tr>
</tbody>
</table>

NLDACTBC
Contains the number of active boundary constraints at the solution \(x^*\) that are recognized as linearly dependent. The \_RHS\_ variable contains a missing value, and the \_NAME\_ variable is blank.
### Table 29.7 continued

<table>
<thead>
<tr>
<th><em>TYPE</em></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLDACTLC</td>
<td>Contains the number of active linear constraints at the solution ( x^* ) that are recognized as linearly dependent. The <em>RHS</em> variable contains a missing value, and the <em>NAME</em> variable is blank.</td>
</tr>
<tr>
<td><em>NOBS</em></td>
<td>Contains the number of observations.</td>
</tr>
<tr>
<td>PARMS</td>
<td>Contains the final parameter estimates. The <em>RHS</em> variable contains the value of the objective function.</td>
</tr>
<tr>
<td>PCRPJ_LF</td>
<td>Contains the projected Hessian matrix of the Lagrange function (based on CRPJAC).</td>
</tr>
<tr>
<td>PHESS_LF</td>
<td>Contains the projected Hessian matrix of the Lagrange function (based on HESSIAN).</td>
</tr>
<tr>
<td>PROJCRPJ</td>
<td>Contains the projected Hessian matrix (based on CRPJAC).</td>
</tr>
<tr>
<td>PROJGRAD</td>
<td>If linear constraints are used in the estimation, this observation contains the ( n - n_{\text{act}} ) values of the projected gradient ( g_z = Z'g ) in the variables corresponding to the first ( n - n_{\text{act}} ) parameters. The <em>RHS</em> variable contains a missing value, and the <em>NAME</em> variable is blank.</td>
</tr>
<tr>
<td>PROJHESS</td>
<td>Contains the projected Hessian matrix (based on HESSIAN).</td>
</tr>
<tr>
<td>STDERR</td>
<td>Contains approximate standard errors (only for METHOD=ML, METHOD=GLS, or METHOD=WLS).</td>
</tr>
<tr>
<td>TERMINAT</td>
<td>The <em>NAME</em> variable contains the name of the termination criterion.</td>
</tr>
<tr>
<td>UPPERBD</td>
<td>If boundary constraints are used, this observation contains the upper bounds. Those parameters not subjected to upper bounds contain missing values. The <em>RHS</em> variable contains a missing value, and the <em>NAME</em> variable is blank.</td>
</tr>
</tbody>
</table>

If the technique specified by the OMETHOD= option cannot be performed (for example, no feasible initial values can be computed or the function value or derivatives cannot be evaluated at the starting point), the OUTEST= data set can contain only some of the observations (usually only the PARMS and GRAD observations).

#### OUTMODEL= or OUTRAM= Data Set

The OUTMODEL= (or OUTRAM=) data set is of TYPE=CALISMDL and contains the model specification, the computed parameter estimates, and the standard error estimates. This data set is intended to be reused as an INMODEL= data set to specify good initial values in a subsequent analysis by PROC CALIS.
The OUTMODEL= data set contains the following variables:

- the BY variables, if any
- an _MDLNUM_ variable for model numbers, if used
- a character variable _TYPE_, which takes various values that indicate the type of model specification
- a character variable _NAME_, which indicates the model type, parameter name, or variable name
- a character variable _MATNR_, which indicates the matrix number (COSAN models only)
- a character variable _VAR1_, which is the name or number of the first variable in the specification
- a character variable _VAR2_, which is the name or number of the second variable in the specification
- a numerical variable _ESTIM_ for the final estimate of the parameter location
- a numerical variable _STDERR_ for the standard error estimate of the parameter location
- a numerical variable _SDEST_ for the final standardized estimate of the parameter location
- a numerical variable _SDSE_ for the standard error of the standardized estimate of the parameter location

Although the _SDEST_ and _SDSE_ variables are created for COSAN models, the values for these two variables are always missing because there are no rules to carry out the standardization of COSAN models.

Each observation (record) of the OUTMODEL= data set contains a piece of information regarding the model specification. Depending on the type of the specification indicated by the value of the _TYPE_ variable, the meanings of _NAME_, _VAR1_, and _VAR2_ differ. The following tables summarize the meanings of the _NAME_, _MATNR_ (COSAN models only), _VAR1_, and _VAR2_ variables for each value of the _TYPE_ variable, given the type of the model.

### COSAN Models

<table>
<thead>
<tr>
<th><em>TYPE</em></th>
<th>Description</th>
<th><em>NAME</em></th>
<th><em>MATNR</em></th>
<th><em>VAR1</em></th>
<th><em>VAR2</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>MDLTYPE</td>
<td>Model type</td>
<td>COSAN</td>
<td>Matrix number</td>
<td>Column location</td>
<td></td>
</tr>
<tr>
<td>VAR</td>
<td>Variable</td>
<td>Variable name</td>
<td>Matrix number</td>
<td>Number of rows</td>
<td>Number of columns</td>
</tr>
<tr>
<td>MATRIX</td>
<td>Matrix</td>
<td>Matrix name</td>
<td>Matrix number</td>
<td>Term number</td>
<td>Location in term number</td>
</tr>
<tr>
<td>MODEL</td>
<td>Model formula</td>
<td>COV or MEAN</td>
<td>Matrix number</td>
<td></td>
<td>Column number</td>
</tr>
<tr>
<td>ESTIM</td>
<td>Parameters</td>
<td>Parameter name</td>
<td>Matrix number</td>
<td>Row number</td>
<td></td>
</tr>
</tbody>
</table>

The value of the _NAME_ variable is COSAN for the _TYPE_=_MDLTYPE observation.

The _TYPE_=_VAR observations store the information about the column variables in matrices. The _NAME_ variable stores the variable names. The value of _VAR1_ indicates the column location of the variable in the matrix with the matrix number stored in _MATNR_.

The _TYPE_=MATRIX observations store the information about the model matrices. The _NAME_ variable stores the matrix names. The value of _MATNR_ indicates the corresponding matrix number. The values of _VAR1_ and _VAR2_ indicate the numbers of rows and columns, respectively, of the matrix.

The _TYPE_=MODEL observations store the covariance and mean structure formulas. The _NAME_ variable indicates whether the mean (MEAN) or covariance (COV) structure information is stored. The value of _MATNR_ indicates the matrix number in the mean or covariance structure formula. The _VAR1_ variable indicates the term number, and the _VAR2_ variable indicates the location of the matrix in the term.

The _TYPE_=ESTIM observations store the information about the parameters and their estimates. The _NAME_ variable stores the parameter names. The value of _MATNR_ indicates the matrix number. The values of _VAR1_ and _VAR2_ indicate the associated row and column numbers, respectively, of the parameter.

### FACTOR Models

<table>
<thead>
<tr>
<th><em>TYPE</em>=</th>
<th>Description</th>
<th><em>NAME</em></th>
<th><em>VAR1</em></th>
<th><em>VAR2</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>MDLTYPE</td>
<td>Model type</td>
<td>Model type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FACTVAR</td>
<td>Variable</td>
<td>Variable name</td>
<td>Variable number</td>
<td>Variable type</td>
</tr>
<tr>
<td>LOADING</td>
<td>Factor loading</td>
<td>Parameter name</td>
<td>Manifest variable</td>
<td>Factor variable</td>
</tr>
<tr>
<td>COV</td>
<td>Covariance</td>
<td>Parameter name</td>
<td>First variable</td>
<td>Second variable</td>
</tr>
<tr>
<td>PVAR</td>
<td>(Partial) variance</td>
<td>Parameter name</td>
<td>Variable</td>
<td></td>
</tr>
<tr>
<td>MEAN</td>
<td>Mean or intercept</td>
<td>Parameter name</td>
<td>Variable</td>
<td></td>
</tr>
<tr>
<td>ADDCOV</td>
<td>Added covariance</td>
<td>Parameter name</td>
<td>First variable</td>
<td>Second variable</td>
</tr>
<tr>
<td>ADDPVAR</td>
<td>Added (partial) variance</td>
<td>Parameter name</td>
<td>Variable</td>
<td></td>
</tr>
<tr>
<td>ADDMEAN</td>
<td>Added mean or intercept</td>
<td>Parameter name</td>
<td>Variable</td>
<td></td>
</tr>
</tbody>
</table>

For factor models, the value of the _NAME_ variable is either EFACTOR (exploratory factor model) or CFACTOR (confirmatory factor model) for the _TYPE_=MDLTYPE observation.

The _TYPE_=FACTVAR observations store the information about the variables in the model. The _NAME_ variable stores the variable names. The value of _VAR1_ indicates the variable number. The value of _VAR2_ indicates the type of the variable: either DEPV for dependent observed variables or INDF for latent factors.

Other observations specify the parameters and their estimates in the model. The _NAME_ values for these observations are the parameter names. Observation with _TYPE_=LOADING, _TYPE_=COV, or _TYPE_=ADDCOV are for parameters that are associated with two variables. The _VAR1_ and _VAR2_ values of these two types of observations indicate the variables involved.

Observations with _TYPE_=PVAR, _TYPE_=MEAN, _TYPE_=ADDPVAR, or _TYPE_=ADDMEAN are for parameters that are associated with a single variable. The value of _VAR1_ indicates the variable involved.
### LINEQS Models

<table>
<thead>
<tr>
<th><em>TYPE</em></th>
<th>Description</th>
<th><em>NAME</em></th>
<th><em>VAR1</em></th>
<th><em>VAR2</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>MDLTYPE</td>
<td>Model type</td>
<td>LINEQS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EQSVAR</td>
<td>Variable</td>
<td></td>
<td>Variable number</td>
<td>Variable type</td>
</tr>
<tr>
<td>EQUATION</td>
<td>Path coefficient</td>
<td>Parameter</td>
<td>Outcome variable</td>
<td>Predictor variable</td>
</tr>
<tr>
<td>COV</td>
<td>Covariance</td>
<td></td>
<td>Parameter</td>
<td>First variable</td>
</tr>
<tr>
<td>VARIANCE</td>
<td>Variance</td>
<td></td>
<td>Parameter</td>
<td>Variable</td>
</tr>
<tr>
<td>MEAN</td>
<td>Mean</td>
<td></td>
<td>Parameter</td>
<td>Variable</td>
</tr>
<tr>
<td>ADDCOV</td>
<td>Added covariance</td>
<td>Parameter</td>
<td>First variable</td>
<td>Second variable</td>
</tr>
<tr>
<td>ADDVARIA</td>
<td>Added variance</td>
<td>Parameter</td>
<td>Variable</td>
<td></td>
</tr>
<tr>
<td>ADDINTE</td>
<td>Added intercept</td>
<td>Parameter</td>
<td>Variable</td>
<td></td>
</tr>
<tr>
<td>ADDMEAN</td>
<td>Added mean</td>
<td></td>
<td>Parameter</td>
<td>Variable</td>
</tr>
</tbody>
</table>

The value of the _NAME_ variable is LINEQS for the _TYPE_=MDLTYPE observation.

The _TYPE_=EQSVAR observations store the information about the variables in the model. The _NAME_ variable stores the variable names. The value of _VAR1_ indicates the variable number. The value of _VAR2_ indicates the type of the variable. There are six types of variables in the LINEQS model:

- DEPV for dependent observed variables
- INDV for independent observed variables
- DEPF for dependent latent factors
- INDF for independent latent factors
- INDD for independent error terms
- INDE for independent disturbance terms

Other observations specify the parameters and their estimates in the model. The _NAME_ values for these observations are the parameter names. Observation with _TYPE_=EQUATION, _TYPE_=COV, or _TYPE_=ADDCOV are for parameters that are associated with two variables. The _VAR1_ and _VAR2_ values of these two types of observations indicate the variables involved.

Observations with _TYPE_=VARIANCE, _TYPE_=MEAN, _TYPE_=ADDVARIA, _TYPE_=ADDINTE, or _TYPE_=ADDMEAN are for parameters associated with a single variable. The value of _VAR1_ indicates the variable involved.
LISMOD Models

<table>
<thead>
<tr>
<th><em>TYPE</em></th>
<th>Description</th>
<th><em>NAME</em></th>
<th><em>VAR1</em></th>
<th><em>VAR2</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>MDLTYPE</td>
<td>model type</td>
<td>LISMOD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XVAR</td>
<td>x-variable</td>
<td>Variable</td>
<td>Variable</td>
<td>number</td>
</tr>
<tr>
<td>YVAR</td>
<td>y-variable</td>
<td>Variable</td>
<td>Variable</td>
<td>number</td>
</tr>
<tr>
<td>ETAVAR</td>
<td>$\eta$-variable</td>
<td>Variable</td>
<td>Variable</td>
<td>number</td>
</tr>
<tr>
<td>XIVAR</td>
<td>$\xi$-variable</td>
<td>Variable</td>
<td>Variable</td>
<td>number</td>
</tr>
<tr>
<td>ALPHA</td>
<td><em>ALPHA</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td></td>
</tr>
<tr>
<td>BETA</td>
<td><em>BETA</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>GAMMA</td>
<td><em>BETA</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>KAPPA</td>
<td><em>KAPPA</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td></td>
</tr>
<tr>
<td>LAMBDA X</td>
<td><em>LAMBDA X</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>LAMBDA Y</td>
<td><em>LAMBDA Y</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>NUX</td>
<td><em>NUX</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td></td>
</tr>
<tr>
<td>NUY</td>
<td><em>NUY</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td></td>
</tr>
<tr>
<td>PHI</td>
<td><em>PHI</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>PSI</td>
<td><em>PSI</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>THETA X</td>
<td><em>THETA X</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>THETA Y</td>
<td><em>THETA Y</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>ADDALPHA</td>
<td>Added <em>ALPHA</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td></td>
</tr>
<tr>
<td>ADDKAPPA</td>
<td>Added <em>KAPPA</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td></td>
</tr>
<tr>
<td>ADDNUX</td>
<td>Added <em>NUX</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td></td>
</tr>
<tr>
<td>ADDNUY</td>
<td>Added <em>NUY</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td></td>
</tr>
<tr>
<td>ADDPHI</td>
<td>Added <em>PHI</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>ADDPSI</td>
<td>Added <em>PSI</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>ADTTHETA X</td>
<td>Added <em>THETA X</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>ADTTHETA Y</td>
<td>Added <em>THETA Y</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
</tbody>
</table>

The value of the _NAME_ variable is LISMOD for the _TYPE_ = MDLTYPE observation. Other observations specify either the variables or the parameters in the model.

Observations with _TYPE_ values equal to XVAR, YVAR, ETAVAR, and XIVAR indicate the variables in the respective lists in the model. The _NAME_ variable of these observations stores the names of the variables, and the _VAR1_ variable stores the variable numbers in the respective list. The variable numbers in this data set are not arbitrary—that is, they define the variable orders in the rows and columns of the LISMOD model matrices. The _VAR2_ variable of these observations is not used.

All other observations in this data set specify the parameters in the model. The _NAME_ values of these observations are the parameter names. The corresponding _VAR1_ and _VAR2_ values of these observations indicate the row and column locations of the parameters in the LISMOD model matrices that are specified in the _TYPE_ variable. For example, when the value of _TYPE_ is ADDPHI or PHI, the parameter specified is located in the _PHI_ matrix, with its row and column numbers indicated by the _VAR1_ and _VAR2_ values, respectively. Some observations for specifying parameters do not have values in the _VAR2_ variable. This means that the associated LISMOD matrices are vectors so that the column numbers are always 1 for these observations.
**MSTRUCT Models**

<table>
<thead>
<tr>
<th><em>TYPE</em></th>
<th>Description</th>
<th><em>NAME</em></th>
<th><em>VAR1</em></th>
<th><em>VAR2</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>MDLTYPE</td>
<td>Model type</td>
<td>MSTRUCT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VAR</td>
<td>Variable</td>
<td></td>
<td>Variable number</td>
<td></td>
</tr>
<tr>
<td>COVMAT</td>
<td>Covariance</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>MEANVEC</td>
<td>Mean</td>
<td>Parameter</td>
<td>Row number</td>
<td></td>
</tr>
<tr>
<td>ADCOV MAT</td>
<td>Added covariance</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>AMEANVEC</td>
<td>Added mean</td>
<td>Parameter</td>
<td>Row number</td>
<td></td>
</tr>
</tbody>
</table>

The value of the _NAME_ variable is MSTRUCT for the _TYPE_ = MDLTYPE observation. Other observations specify either the variables or the parameters in the model.

Observations with _TYPE_ values equal to VAR indicate the variables in the model. The _NAME_ variable of these observations stores the names of the variables, and the _VAR1_ variable stores the variable numbers in the variable list. The variable numbers in this data set are not arbitrary—that is, they define the variable orders in the rows and columns of the mean and covariance matrices. The _VAR2_ variable of these observations is not used.

All other observations in this data set specify the parameters in the model. The _NAME_ values of these observations are the parameter names. The corresponding _VAR1_ and _VAR2_ values of these observations indicate the row and column locations of the parameters in the mean or covariance matrix, as specified in the _TYPE_ model. For example, when _TYPE_ = COVMAT, the parameter specified is located in the covariance matrix, with its row and column numbers indicated by the _VAR1_ and _VAR2_ values, respectively. For observations with _TYPE_ = MEANVEC, the _VAR2_ variable is not used because the column numbers are always 1 for parameters in the mean vector.

**PATH Models**

<table>
<thead>
<tr>
<th><em>TYPE</em></th>
<th>Description</th>
<th><em>NAME</em></th>
<th><em>VAR1</em></th>
<th><em>VAR2</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>MDLTYPE</td>
<td>Model type</td>
<td>PATH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PATHVAR</td>
<td>Variable</td>
<td>Variable name</td>
<td>Variable number</td>
<td>Variable type</td>
</tr>
<tr>
<td>LEFT</td>
<td>Path coefficient</td>
<td>Parameter</td>
<td>Outcome variable</td>
<td>Predictor variable</td>
</tr>
<tr>
<td>RIGHT</td>
<td>Path coefficient</td>
<td>Parameter</td>
<td>Predictor variable</td>
<td>Outcome variable</td>
</tr>
<tr>
<td>PCOV</td>
<td>(Partial) covariance</td>
<td>Parameter</td>
<td>First variable</td>
<td>Second variable</td>
</tr>
<tr>
<td>PCOV PATH</td>
<td>(Partial) covariance path</td>
<td>Parameter</td>
<td>First variable</td>
<td>Second variable</td>
</tr>
<tr>
<td>PVAR</td>
<td>(Partial) variance</td>
<td>Parameter</td>
<td>Variable</td>
<td></td>
</tr>
<tr>
<td>PVARP ATH</td>
<td>(Partial) variance path</td>
<td>Parameter</td>
<td>Variable</td>
<td>Variable</td>
</tr>
<tr>
<td>MEAN</td>
<td>Mean or intercept</td>
<td>Parameter</td>
<td>Variable</td>
<td></td>
</tr>
<tr>
<td>ONEPATH</td>
<td>Mean or intercept path</td>
<td>Parameter</td>
<td><em>ONE</em></td>
<td>Variable</td>
</tr>
<tr>
<td>ADDPCOV</td>
<td>Added (partial) covariance</td>
<td>Parameter</td>
<td>First variable</td>
<td>Second variable</td>
</tr>
<tr>
<td>ADDPV AR</td>
<td>Added (partial) variance</td>
<td>Parameter</td>
<td>Variable</td>
<td></td>
</tr>
<tr>
<td>ADDMEAN</td>
<td>Added mean</td>
<td>Parameter</td>
<td>Variable</td>
<td></td>
</tr>
</tbody>
</table>

The value of the _NAME_ variable is PATH for the _TYPE_ = MDLTYPE observation.
Chapter 29: The CALIS Procedure

The _TYPE_ = PATHVAR observations store the information about the variables in the model. The _NAME_ variable stores the variable names. The value of _VAR1_ indicates the variable number. The value of _VAR2_ indicates the type of the variable. There are four types of variables in the PATH model:

- DEPV for dependent observed variables
- INDV for independent observed variables
- DEPF for dependent latent factors
- INDF for independent latent factors

Other observations specify the parameters in the model. The _NAME_ values for these observations are the parameter names. Observation with _TYPE_ = LEFT, _TYPE_ = RIGHT, _TYPE_ = PCOV, or _TYPE_ = ADDPCOV are for parameters that are associated with two variables. The _VAR1_ and _VAR2_ values of these two types of observations indicate the variables involved.

Observations with _TYPE_ = PVAR, _TYPE_ = MEAN, _TYPE_ = ADDPVAR, or _TYPE_ = ADDMEAN are for parameters that are associated with a single variable. The value of _VAR1_ indicates the variable involved.

**RAM Models**

<table>
<thead>
<tr>
<th><em>TYPE</em> =</th>
<th>Description</th>
<th><em>NAME</em></th>
<th><em>VAR1</em></th>
<th><em>VAR2</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>MDLTYPE</td>
<td>Model type</td>
<td>RAM</td>
<td>Variable number</td>
<td>Variable type</td>
</tr>
<tr>
<td>RAMVAR</td>
<td>Variable name</td>
<td>Variable</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td><em>A</em></td>
<td><em>A</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td><em>P</em></td>
<td><em>P</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td><em>W</em></td>
<td><em>W</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>ADD_P_</td>
<td>Added <em>P</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
<tr>
<td>ADD_W_</td>
<td>Added <em>W</em> entry</td>
<td>Parameter</td>
<td>Row number</td>
<td>Column number</td>
</tr>
</tbody>
</table>

The value of the _NAME_ variable is RAM for the _TYPE_ = MDLTYPE observation.

For the _TYPE_ = RAMVAR observations, the _NAME_ variable stores the variable names, the _VAR1_ variable stores the variable number, and the _VAR2_ variable stores the variable type. There are four types of variables in the PATH model:

- DEPV for dependent observed variables
- INDV for independent observed variables
- DEPF for dependent latent factors
- INDF for independent latent factors

Other observations specify the parameters in the model. The _NAME_ variable stores the parameter name. The _TYPE_ variable indicates the associated matrix with the row number indicated in _VAR1_ and column number indicated in _VAR2_.

Reading an OUTMODEL= Data Set As an INMODEL= Data Set in Subsequent Analyses

When the OUTMODEL= data set is treated as an INMODEL= data set in subsequent analyses, you need to pay attention to observations with _TYPE_ values prefixed by “ADD”, “AD”, or “A” (for example, ADDCOV, ADTHETAY, or AMEANVEG). These observations represent default parameter locations that are generated by PROC CALIS in a previous run. Because the context of the new analyses might be different, these observations for added parameter locations might no longer be suitable in the new runs. Hence, these observations are not read as input model information. Fortunately, after reading the INMODEL= specification in the new analyses, CALIS analyzes the new model specification again. It then adds an appropriate set of parameters in the new context when necessary. If you are certain that the added parameter locations in the INMODEL= data set are applicable, you can force the input of these observations by using the READADDPARM option in the PROC CALIS statement. However, you must be very careful about using the READADDPARM option. The added parameters from the INMODEL= data set might have the same parameter names as those for the generated parameters in the new run. This might lead to unnecessary constraints in the model.

OUTSTAT= Data Set

The OUTSTAT= data set is similar to the TYPE=COV, TYPE=UCOV, TYPE=CORR, or TYPE=UCORR data set produced by the CORR procedure. The OUTSTAT= data set contains the following variables:

- the BY variables, if any
- the _GPNUM_ variable for groups numbers, if used in the analysis
- two character variables, _TYPE_ and _NAME_
- the manifest and the latent variables analyzed

The OUTSTAT= data set contains the following information (when available) in the observations:

- the mean and standard deviation
- the skewness and kurtosis (if the DATA= data set is a raw data set and the KURTOSIS option is specified)
- the number of observations
- if the WEIGHT statement is used, sum of the weights
- the correlation or covariance matrix to be analyzed
- the robust covariances, standard deviations, and means for robust estimation
- the predicted correlation or covariance matrix
- the standardized or normalized residual correlation or covariance matrix
- if the model contains latent variables, the predicted covariances between latent and manifest variables and the latent variable (or factor) score regression coefficients (see the PLATCOV option on page 1231)
In addition, for FACTOR models the OUTSTAT= data set contains:

- the unrotated factor loadings, the error variances, and the matrix of factor correlations
- the standardized factor loadings and factor correlations
- the rotation matrix, rotated factor loadings, and factor correlations
- standardized rotated factor loadings and factor correlations

If effects are analyzed, the OUTSTAT= data set also contains:

- direct, indirect, and total effects and their standard error estimates
- standardized direct, indirect, and total effects and their standard error estimates

Each observation in the OUTSTAT= data set contains some type of statistic as indicated by the _TYPE_ variable. The values of the _TYPE_ variable are shown in the following tables:

### Basic Descriptive Statistics

<table>
<thead>
<tr>
<th>Value of <em>TYPE</em></th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJCOV</td>
<td>Adjusted covariances</td>
</tr>
<tr>
<td>ADJSTD</td>
<td>Adjusted standard deviations</td>
</tr>
<tr>
<td>CORR</td>
<td>Correlations</td>
</tr>
<tr>
<td>COV</td>
<td>Covariances</td>
</tr>
<tr>
<td>KURTOSIS</td>
<td>Univariate kurtosis</td>
</tr>
<tr>
<td>MEAN</td>
<td>Means</td>
</tr>
<tr>
<td>N</td>
<td>Sample size</td>
</tr>
<tr>
<td>NPARTIAL</td>
<td>Number of partial variables</td>
</tr>
<tr>
<td>PARTCOV</td>
<td>Covariances after partialling</td>
</tr>
<tr>
<td>PARTCORR</td>
<td>Correlations after partialling</td>
</tr>
<tr>
<td>PARTMEAN</td>
<td>Means after partialling</td>
</tr>
<tr>
<td>PARTSTD</td>
<td>Standard deviations after partialling</td>
</tr>
<tr>
<td>ROBCOV</td>
<td>Robust covariances</td>
</tr>
<tr>
<td>ROBMEAN</td>
<td>Robust means</td>
</tr>
<tr>
<td>ROBSTD</td>
<td>Robust standard deviations</td>
</tr>
<tr>
<td>SKEWNESS</td>
<td>Univariate skewness</td>
</tr>
<tr>
<td>STD</td>
<td>Standard deviations</td>
</tr>
<tr>
<td>SUMWGT</td>
<td>Sum of weights (if the WEIGHT statement is used)</td>
</tr>
<tr>
<td>VARDIV</td>
<td>Variance divisor</td>
</tr>
<tr>
<td>VARDIVAJ</td>
<td>Variance divisor adjustment</td>
</tr>
</tbody>
</table>

For the _TYPE_ = CORR or COV observations, the _NAME_ variable contains the name of the manifest variable that corresponds to each row for the covariance or correlation. For other observations, _NAME_ is blank.
### Predicted Moments and Residuals

<table>
<thead>
<tr>
<th>value of <em>TYPE</em></th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>METHOD=DWLS</strong></td>
<td></td>
</tr>
<tr>
<td>DWLSPRED</td>
<td>DWLS predicted moments</td>
</tr>
<tr>
<td>DWLSRES</td>
<td>DWLS raw residuals</td>
</tr>
<tr>
<td>DWLSSRES</td>
<td>DWLS variance standardized residuals</td>
</tr>
<tr>
<td><strong>METHOD=GLS</strong></td>
<td></td>
</tr>
<tr>
<td>GLSASRES</td>
<td>GLS asymptotically standardized residuals</td>
</tr>
<tr>
<td>GLSNRES</td>
<td>GLS normalized residuals</td>
</tr>
<tr>
<td>GLSPRED</td>
<td>GLS predicted moments</td>
</tr>
<tr>
<td>GLSRES</td>
<td>GLS raw residuals</td>
</tr>
<tr>
<td>GLSSRES</td>
<td>GLS variance standardized residuals</td>
</tr>
<tr>
<td><strong>METHOD=ML or FIML</strong></td>
<td></td>
</tr>
<tr>
<td>MAXASRES</td>
<td>ML asymptotically standardized residuals</td>
</tr>
<tr>
<td>MAXNRES</td>
<td>ML normalized residuals</td>
</tr>
<tr>
<td>MAXPRED</td>
<td>ML predicted moments</td>
</tr>
<tr>
<td>MAXRES</td>
<td>ML raw residuals</td>
</tr>
<tr>
<td>MAXSRES</td>
<td>ML variance standardized residuals</td>
</tr>
<tr>
<td><strong>METHOD=ULS</strong></td>
<td></td>
</tr>
<tr>
<td>ULSASRES</td>
<td>ULS asymptotically standardized residuals</td>
</tr>
<tr>
<td>ULSRES</td>
<td>ULS raw residuals</td>
</tr>
<tr>
<td>ULSSRES</td>
<td>ULS variance standardized residuals</td>
</tr>
<tr>
<td><strong>METHOD=WLS</strong></td>
<td></td>
</tr>
<tr>
<td>WLSASRES</td>
<td>WLS asymptotically standardized residuals</td>
</tr>
<tr>
<td>WLSNRES</td>
<td>WLS normalized residuals</td>
</tr>
<tr>
<td>WLSASRES</td>
<td>WLS predicted moments</td>
</tr>
<tr>
<td>WLSRES</td>
<td>WLS raw residuals</td>
</tr>
<tr>
<td>WLSSRES</td>
<td>WLS variance standardized residuals</td>
</tr>
</tbody>
</table>

For residuals or predicted moments of means, the _NAME_ variable is a fixed value denoted by _Mean_. For residuals or predicted moments for covariances or correlations, the _NAME_ variable is used for names of variables.
Effects and Latent Variable Scores Regression Coefficients

<table>
<thead>
<tr>
<th>Value of <em>TYPE</em></th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unstandardized Effects</strong></td>
<td></td>
</tr>
<tr>
<td>DEFFECT</td>
<td>Direct effects</td>
</tr>
<tr>
<td>DEFF_SE</td>
<td>Standard error estimates for direct effects</td>
</tr>
<tr>
<td>IEFFECT</td>
<td>Indirect effects</td>
</tr>
<tr>
<td>IEFF_SE</td>
<td>Standard error estimates for indirect effects</td>
</tr>
<tr>
<td>TEFFECT</td>
<td>Total effects</td>
</tr>
<tr>
<td>TEFF_SE</td>
<td>Standard error estimates for total effects</td>
</tr>
<tr>
<td><strong>Standardized Effects</strong></td>
<td></td>
</tr>
<tr>
<td>SDEFF</td>
<td>Standardized direct effects</td>
</tr>
<tr>
<td>SDEFF_SE</td>
<td>Standard error estimates for standardized direct effects</td>
</tr>
<tr>
<td>SIEFF</td>
<td>Standardized indirect effects</td>
</tr>
<tr>
<td>SIEFF_SE</td>
<td>Standard error estimates for standardized indirect effects</td>
</tr>
<tr>
<td>STEFF</td>
<td>Standardized total effects</td>
</tr>
<tr>
<td>STEFF_SE</td>
<td>Standard error estimates for standardized total effects</td>
</tr>
<tr>
<td><strong>Latent Variable Scores Coefficients</strong></td>
<td></td>
</tr>
<tr>
<td>LSSCORE</td>
<td>Latent variable (or factor) scores regression coefficients for ULS method</td>
</tr>
<tr>
<td>SCORE</td>
<td>Latent variable (or factor) scores regression coefficients other than ULS method</td>
</tr>
</tbody>
</table>

For latent variable or factor scores coefficients, the _NAME_ variable contains factor or latent variables in the observations. For other observations, the _NAME_ variable contains manifest or latent variable names.

You can use the latent variable score regression coefficients with PROC SCORE to compute factor scores. If the analyzed matrix is a covariance rather than a correlation matrix, the _TYPE_=STD observation is not included in the OUTSTAT= data set. In this case, the standard deviations can be obtained from the diagonal elements of the covariance matrix. Dropping the _TYPE_=STD observation prevents PROC SCORE from standardizing the observations before computing the factor scores.

Factor Analysis Results

<table>
<thead>
<tr>
<th>Value of <em>TYPE</em></th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRVAR</td>
<td>Error variances</td>
</tr>
<tr>
<td>FCOV</td>
<td>Factor correlations or covariances</td>
</tr>
<tr>
<td>LOADINGS</td>
<td>Unrotated factor loadings</td>
</tr>
<tr>
<td>RFCOV</td>
<td>Rotated factor correlations or covariances</td>
</tr>
<tr>
<td>RLOADING</td>
<td>Rotated factor loadings</td>
</tr>
<tr>
<td>ROTMAT</td>
<td>Rotation matrix</td>
</tr>
<tr>
<td>STDERRVAR</td>
<td>Error variances in standardized solutions</td>
</tr>
<tr>
<td>STDFCOV</td>
<td>Standardized factor correlations</td>
</tr>
<tr>
<td>STDLOAD</td>
<td>Standardized factor loadings</td>
</tr>
<tr>
<td>STDRFCOV</td>
<td>Standardized rotated factor correlations or covariances</td>
</tr>
<tr>
<td>STDRLOAD</td>
<td>Standardized rotated factor loadings</td>
</tr>
</tbody>
</table>

For the _TYPE_=ERRVAR observation, the _NAME_ variable is blank. For all other observations, the _NAME_ variable contains factor names.
OUTWGT= Data Set

You can create an OUTWGT= data set that is of TYPE=WEIGHT and contains the weight matrix used in generalized, weighted, or diagonally weighted least squares estimation. The OUTWGT= data set contains the weight matrix on which the WRIDGE= and the WPENALTY= options are applied. However, if you input the inverse of the weight matrix with the INWGT= and INWGTINV options (or the INWGT(INV)= option alone) in the same analysis, the OUTWGT= data set contains the same elements of the inverse of the weight matrix. For unweighted least squares or maximum likelihood estimation, no OUTWGT= data set can be written. The weight matrix used in maximum likelihood estimation is dynamically updated during optimization. When the ML solution converges, the final weight matrix is the same as the predicted covariance or correlation matrix, which is included in the OUTSTAT= data set (observations with _TYPE_=MAXPRED).

For generalized and diagonally weighted least squares estimation, the weight matrices \( W \) of the OUTWGT= data set contain all elements \( w_{ij} \), where the indices \( i \) and \( j \) correspond to all manifest variables used in the analysis. Let \( \text{varnam}_i \) be the name of the \( i \)th variable in the analysis. In this case, the OUTWGT= data set contains \( n \) observations with the variables shown in the following table:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>TYPE</em></td>
<td>WEIGHT (character)</td>
</tr>
<tr>
<td><em>NAME</em></td>
<td>Name of variable ( \text{varnam}_i ) (character)</td>
</tr>
<tr>
<td>( \text{varnam}_1 )</td>
<td>Weight ( w_{ij} ) for variable ( \text{varnam}_1 ) (numeric)</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( \text{varnam}_n )</td>
<td>Weight ( w_{ij} ) for variable ( \text{varnam}_n ) (numeric)</td>
</tr>
</tbody>
</table>

For weighted least squares estimation, the weight matrix \( W \) of the OUTWGT= data set contains only the nonredundant elements \( w_{ij,k} \). In this case, the OUTWGT= data set contains \( n(n+1)(2n+1)/6 \) observations with the variables shown in the following table:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>TYPE</em></td>
<td>WEIGHT (character)</td>
</tr>
<tr>
<td><em>NAME</em></td>
<td>Name of variable ( \text{varnam}_i ) (character)</td>
</tr>
<tr>
<td><em>NAM2</em></td>
<td>Name of variable ( \text{varnam}_j ) (character)</td>
</tr>
<tr>
<td><em>NAM3</em></td>
<td>Name of variable ( \text{varnam}_k ) (character)</td>
</tr>
<tr>
<td>( \text{varnam}_1 )</td>
<td>Weight ( w_{ij,k} ) for variable ( \text{varnam}_1 ) (numeric)</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( \text{varnam}_n )</td>
<td>Weight ( w_{ij,k} ) for variable ( \text{varnam}_n ) (numeric)</td>
</tr>
</tbody>
</table>

Symmetric redundant elements are set to missing values.

OUTFIT= Data Set

You can create an OUTFIT= data set that is of TYPE=CALISFIT and that contains the values of the fit indices of your analysis. If you use two estimation methods such as LSML or LSWLS, the fit indices are for the second analysis. An OUTFIT= data set contains the following variables:

- a character variable _TYPE_ for the types of fit indices
- a numerical variable IndexCode for the codes of the fit indices
• a character variable FitIndex for the names of the fit indices

• a numerical variable FitValue for the numerical values of the fit indices

• a character variable PrintChar for the character-formatted fit index values.

The possible values of _TYPE_ are:

ModellInfo: basic modeling statistics and information
Absolute: stand-alone fit indices
Parsimony: fit indices that take model parsimony into account
Incremental: fit indices that are based on comparison with a baseline model

**Possible Values of FitIndex When _TYPE_=ModellInfo**

<table>
<thead>
<tr>
<th>Value of FitIndex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations</td>
<td>Number of observations used in the analysis</td>
</tr>
<tr>
<td>Number of Complete Observations</td>
<td>Number of complete observations</td>
</tr>
<tr>
<td></td>
<td>(METHOD=FIML)</td>
</tr>
<tr>
<td>Number of Incomplete Observations</td>
<td>Number of incomplete observations</td>
</tr>
<tr>
<td></td>
<td>(METHOD=FIML)</td>
</tr>
<tr>
<td>Number of Variables</td>
<td>Number of variables</td>
</tr>
<tr>
<td>Number of Moments</td>
<td>Number of mean or covariance elements</td>
</tr>
<tr>
<td>Number of Parameters</td>
<td>Number of parameters</td>
</tr>
<tr>
<td>Number of Active Constraints</td>
<td>Number of active constraints in the solution</td>
</tr>
<tr>
<td>Saturated Model Estimation</td>
<td>Estimation status of the saturated model</td>
</tr>
<tr>
<td></td>
<td>(METHOD=FIML)</td>
</tr>
<tr>
<td>Saturated Model Function Value</td>
<td>Saturated model function value</td>
</tr>
<tr>
<td></td>
<td>(METHOD=FIML)</td>
</tr>
<tr>
<td>Saturated Model -2 Log-Likelihood</td>
<td>Saturated model –2 log-likelihood function value (METHOD=FIML)</td>
</tr>
<tr>
<td>Baseline Model Estimation</td>
<td>Estimation status of the baseline model</td>
</tr>
<tr>
<td></td>
<td>(METHOD=FIML)</td>
</tr>
<tr>
<td>Baseline Model Function Value</td>
<td>Baseline model function value</td>
</tr>
<tr>
<td>Baseline Model -2 Log-Likelihood</td>
<td>Baseline model –2 log-likelihood function value (METHOD=FIML)</td>
</tr>
<tr>
<td>Baseline Model Chi-Square</td>
<td>Baseline model chi-square value</td>
</tr>
<tr>
<td>Baseline Model Chi-Square DF</td>
<td>Baseline model chi-square degrees of freedom</td>
</tr>
<tr>
<td>Baseline Model DF</td>
<td>Baseline model degrees of freedom</td>
</tr>
<tr>
<td></td>
<td>(METHOD=ULS or METHOD=DWLS)</td>
</tr>
<tr>
<td>Pr &gt; Baseline Model Chi-Square</td>
<td>p value of the baseline model chi-square</td>
</tr>
</tbody>
</table>
### Possible Values of FitIndex When _TYPE_ =Absolute

<table>
<thead>
<tr>
<th>Value of FitIndex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fit Function</td>
<td>Fit function value</td>
</tr>
<tr>
<td>-2 Log-Likelihood</td>
<td>-2 log-likelihood function value for the model (METHOD=FIML)</td>
</tr>
<tr>
<td>Chi-Square</td>
<td>Model chi-square value</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>Degrees of freedom for the model chi-square test</td>
</tr>
<tr>
<td>Model DF</td>
<td>Degrees of freedom for model (METHOD=ULS or METHOD=DWLS)</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>Probability of obtaining a larger chi-square than the observed value</td>
</tr>
<tr>
<td>Elliptic Corrected Chi-Square</td>
<td>Elliptic-corrected chi-square value</td>
</tr>
<tr>
<td>Pr &gt; Elliptic Corr. Chi-Square</td>
<td>Probability of obtaining a larger elliptic-corrected chi-square value</td>
</tr>
<tr>
<td>Z-test of Wilson and Hilferty</td>
<td>Z-test of Wilson and Hilferty</td>
</tr>
<tr>
<td>Hoelter Critical N</td>
<td>N value that makes a significant chi-square when multiplied to the fit function value</td>
</tr>
<tr>
<td>Root Mean Square Residual (RMR)</td>
<td>Root mean square residual</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>Standardized root mean square residual</td>
</tr>
<tr>
<td>Goodness of Fit Index (GFI)</td>
<td>Jöreskog and Sörbom goodness-of-fit index</td>
</tr>
</tbody>
</table>

### Possible Values of FitIndex When _TYPE_ =Parsimony

<table>
<thead>
<tr>
<th>Value of FitIndex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjusted GFI (AGFI)</td>
<td>Goodness-of-fit index adjusted for the degrees of freedom of the model</td>
</tr>
<tr>
<td>Parsimonious GFI</td>
<td>Mulaik et al. (1989) modification of the GFI</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>Steiger and Lind (1980) root mean square error of approximation</td>
</tr>
<tr>
<td>RMSEA Lower r% Confidence Limit</td>
<td>Lower r% confidence limit for RMSEA</td>
</tr>
<tr>
<td>RMSEA Upper r% Confidence Limit</td>
<td>Upper r% confidence limit for RMSEA</td>
</tr>
<tr>
<td>Probability of Close Fit</td>
<td>Browne and Cudeck (1993) test of close fit</td>
</tr>
<tr>
<td>ECVI Estimate</td>
<td>Expected cross-validation index</td>
</tr>
<tr>
<td>ECVI Lower r% Confidence Limit</td>
<td>Lower r% confidence limit for ECVI</td>
</tr>
<tr>
<td>ECVI Upper r% Confidence Limit</td>
<td>Upper r% confidence limit for ECVI</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>Akaike information criterion</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>Bozdogan (1987) consistent AIC</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>Schwarz (1978) Bayesian criterion</td>
</tr>
<tr>
<td>McDonald Centrality</td>
<td>McDonald and Marsh (1988) measure of centrality</td>
</tr>
</tbody>
</table>

1. The value of r is one minus the ALPHARMS= value. By default, r=90.
2. The value of r is one minus the ALPHAECV= value. By default, r=90.
Possible Values of FitIndex When _TYPE_=Incremental

<table>
<thead>
<tr>
<th>Value of FitIndex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bentler Comparative Fit Index</td>
<td>Bentler (1985) comparative fit index</td>
</tr>
<tr>
<td>Bentler-Bonett NFI</td>
<td>Bentler and Bonett (1980) normed fit index</td>
</tr>
<tr>
<td>Bentler-Bonett Non-normed Index</td>
<td>Bentler and Bonett (1980) nonnormed fit index</td>
</tr>
<tr>
<td>Bollen Normed Index Rho1</td>
<td>Bollen normed $\rho_1$</td>
</tr>
<tr>
<td>Bollen Non-normed Index Delta2</td>
<td>Bollen nonnormed $\delta_2$</td>
</tr>
<tr>
<td>James et al. Parsimonious NFI</td>
<td>James, Mulaik, and Brett (1982) parsimonious normed fit index</td>
</tr>
</tbody>
</table>

Default Analysis Type and Default Parameterization

This section describes the default analysis type and default parameterization of PROC CALIS. Because various types of models supported by PROC CALIS have their own specific features, the following table outlines only those major default settings of PROC CALIS in SAS/STAT 9.22 or later:

<table>
<thead>
<tr>
<th>Moment Structures Analyzed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariance structures</td>
</tr>
<tr>
<td>Mean structures</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Default Parameterization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance parameters</td>
</tr>
<tr>
<td>Covariance parameters</td>
</tr>
<tr>
<td>Mean parameters</td>
</tr>
<tr>
<td>Intercept parameters</td>
</tr>
</tbody>
</table>

Role of the VAR Statement in Model Specification

Inclusion of Variables

If the VAR statement is used, observed variables that are listed in the VAR statement but not mentioned in corresponding model specification are included as exogenous variables in the model. Default variance, covariance, and mean parameterization apply to these exogenous variables.
Exclusion of Variables

If the VAR statement is used, observed variables that are not listed in the VAR statement are not recognized as observed variables in model specification. They might be treated as latent variables instead.

In general, the default settings of PROC CALIS in SAS/STAT 9.22 and later are more consistent with conventional practice of structural equation modeling. For example, because statistical theory of structural equation modeling is based mainly on the analysis of covariance structures, the default analysis type of PROC CALIS in SAS/STAT 9.22 or later is for covariance structures. This means that the CORR option must be specified if you want to analyze correlation matrices. Mean structures in PROC CALIS have also been parameterized more naturally since SAS/STAT 9.22 so that users can make statistical inferences directly on the mean and intercept estimates. Default variance and covariance parameters for all independent factors and observed variables have been implemented since SAS/STAT 9.22 to reflect the common belief that no exogenous variables are absolutely uncorrelated (except for the unrotated factors in the initial solution of exploratory FACTOR models).

For comparisons of these and other default settings before and after SAS/STAT 9.22, see the section “Compatibility with the CALIS Procedure in SAS/STAT 9.2 or Earlier” on page 1162.

For details about the default parameters in specific types of model, see the FACTOR, LINEQS, LISMOD, PATH, and RAM statements. Because there are no explicit latent variables and the exogenous/endogenous variable distinction is not used in the COSAN and MSTRUCT modeling languages, the default parameterization outlined in the preceding table does not apply to these two types of models. See the COSAN and the MSTRUCT statements for details. See the following sections for details about model parameterization in various types of models:

- “The COSAN Model” on page 1390
- “The FACTOR Model” on page 1394
- “The LINEQS Model” on page 1401
- “The LISMOD Model and Submodels” on page 1408
- “The PATH Model” on page 1419
- “The RAM Model” on page 1425
- “The MSTRUCT Model” on page 1416

By default, PROC CALIS in SAS/STAT 9.22 or later treats observed variables specified in the VAR statement as exogenous variables in the FACTOR, PATH, LINEQS, and RAM models. This minimizes the routine specifications in the COVARIANCE, MEAN, PCOV, PVAR, or VARIANCE for those “standalone” exogenous observed variables—observed variables that are not functionally related to any other variables in the model. Including these standalone exogenous observed variables into the model is useful during the model modification process (by the MODIFICATION option) where you can do Lagrange multiplier tests to see whether relating these variables to other variables can improve the model fit.
Notice that the use of the `VAR` statement specification is optional in PROC CALIS. If you have specified all the observed variables that you want in the model specification statements, it is not necessary to use the `VAR` statement specifications for the purpose of specifying the set of observed variable for analysis.

### The COSAN Model

The original COSAN (covariance structure analysis) model is proposed by McDonald (1978, 1980) for analyzing general covariance structure models. PROC CALIS enables you to analyze a generalized form of the original COSAN model. The generalized COSAN model extends the original COSAN model with the inclusion of addition terms in the covariance structure formula and the associated mean structure formula.

The covariance structure formula of the generalized COSAN model is

\[
\Sigma = F_1 P_1 F_1' + \cdots + F_m P_m F_m'
\]

and the corresponding mean structure formula of the generalized COSAN model is

\[
\mu = F_1 v_1 + \cdots + F_m v_m
\]

where \(\Sigma\) is a symmetric correlation or covariance matrix for the observed variables, \(\mu\) is a vector for the observed variable means, each \(P_k\) is a symmetric matrix, each \(v_k\) is a mean vector, and each \(F_k\) \((k = 1, \ldots, m)\) is the product of \(n(k)\) matrices \(F_{k1}, \ldots, F_{kn(k)}\); that is,

\[
F_k = F_{k1} \cdots F_{kn(k)}, \quad k = 1, \ldots, m
\]

The matrices \(F_{kj}\) and \(P_k\) in the model can be one of the forms

\[
F_{kj} = \begin{cases} 
G_{kj} & j = 1, \ldots, n(k) \\
G_{kj}^{-1} & (I - G_{kj})^{-1}
\end{cases}
\]

\[
P_k = \begin{cases} 
Q_k & \text{when mean structures are analyzed} \\
Q_k^{-1} & \text{parameters and fixed values in the model matrices}
\end{cases}
\]

where \(G_{kj}\) and \(Q_k\) are basic model matrices that are not expressed as functions of other matrices. The COSAN model matrices and vectors are \(G_{kj}\), \(Q_k\), and \(v_k\) (when the mean structures are analyzed). The elements of these model matrices and vectors are either parameters (free or constrained) or fixed values. Matrix \(P_k\) is referred to as the central covariance matrix for the \(k\)th term in the covariance structure formula.

Essentially, the COSAN modeling language enables you to define the covariance and mean structure formulas of the generalized COSAN model, the basic COSAN model matrices \(G_{kj}\), \(Q_k\), and \(v_k\), and the parameters and fixed values in the model matrices.

You can also specify a generalized COSAN model without using an explicit central covariance matrix in any term. For example, you can define the \(k\)th term in the covariance structure formula as

\[
F_k F_k' = F_{k1} \cdots F_{kn-1} F_{kn} F_{kn} F_{kn} F_{kn} F_{kn-1} \cdots F_{k1}
\]

The corresponding term for the mean structure becomes

\[
F_{k1} \cdots F_{kn-1} v_m
\]

In the covariance structure formula, \(F_{kn} F_{kn}'\) serves as an implicit central covariance matrix in this term of the covariance structure formula. Because of this, \(F_{kn}\) does not appear in the corresponding mean structure formula.
To take advantage of the modeling flexibility of the COSAN model specifications, you are required to provide the correct covariance and mean structure formulas for the analysis problem. If you are not familiar with the mathematical formulations of structural equation models, you can consider using simpler modeling languages such as PATH or LINEQS.

**An Example: Specifying a Second-Order Factor Model**

This example illustrates how to specify the covariance structures in the COSAN statement. Consider a second-order factor analysis model with the following formula for the covariance structures of observed variables $v_1$–$v_9$

$$
\Sigma = F_1(F_2P_2F'_2 + U_2)F'_1 + U_1
$$

where $F_1$ is a $9 \times 3$ first-order factor matrix, $F_2$ is a $3 \times 2$ second-order factor matrix, $P_2$ is a $2 \times 2$ covariance matrix for the second-order factors, $U_2$ is a $3 \times 3$ diagonal matrix for the unique variances of the first-order factors, and $U_1$ is a $9 \times 9$ diagonal matrix for the unique variances of the observed variables.

To fit this covariance structure model, you first rewrite the covariance structure formula in the form of the generalized COSAN model as

$$
\Sigma = F_1F_2P_2F'_2F'_1 + F_1U_2F'_1 + U_1
$$

You can specify the list of observed variables and the three terms for the covariance structure formula in the following COSAN statement:

```
Cosan var= v1-v9,
    F1(3) * F2(2) * P2(2,SYM) + F1(3) * U2(3,DIA) + U1(9,DIA);
```

The `VAR=` option specifies the nine observed variables in the model. Next, the three terms of the covariance structure formula are specified. Because each term in the covariance structure formula is a symmetric product, you only need to specify each term up to the central covariance matrix. For example, although the first term in the covariance structure formula is $F_1F_2P_2F'_2F'_1$, you only need to specify $F1(3) * F2(2) * P2(2,SYM)$. PROC CALIS generates the redundant information for the term. Similarly, you specify the other two terms of the covariance structure formula.

In each matrix specification of the COSAN statement, you can specify the following three matrix properties as the arguments in the trailing parentheses: the number of columns, the matrix type, and the transformation of the matrix. For example, $F1(3)$ means that the number of columns of $F1$ is 3 (while the number of rows is 9 because this number has to match the number of observed variables specified in the `VAR=` option), $F2(2)$ means that the number of columns of $F2$ is 2 (while the number of rows is 3 because the number has to match the number of columns of the preceding matrix, $F1$). You can specify the type of the matrix in the second argument. For example, $P2(2,SYM)$ means that $P2$ is a symmetric (SYM) matrix and $U2(2,DIA)$ means that $U2$ is a diagonal (DIA) matrix. You can also specify the transformation of the matrix in the third argument. Because there is no transformation needed in the current second-order factor model, this argument is omitted in the specification. See the COSAN statement for details about the matrix types and transformation that are supported by the COSAN modeling language.

Suppose now you also want to analyze the mean structures of the second-order factor model. The corresponding mean structure formula is

$$
\mu = F_1F_2v + u
$$

where $v$ is a $2 \times 1$ mean vector for the second-order factors and $u$ is a $6 \times 1$ vector for the intercepts of the observed variables. To analyze the mean and covariance structures simultaneously, you can use the following COSAN statement:
cosan var= v1-v9,
  F1(3) * F2(2) * P2(2,SYM) [mean = v] + F1(3) * U2(3,DIA)
  + U1(9,DIA) [mean = u];

In addition to the covariance structure specified, you now add the trailing MEAN= options in the first and the third terms. PROC CALIS then generates the mean structure formula by the following steps:

- Remove the last matrix (that is, the central covariance matrix) in each term of the covariance structure formula.
- Append to each term the vector that is specified in the MEAN= option of the term, or if no MEAN= option is specified in a term, that term becomes a zero vector in the mean structure formula.

Following these steps, the mean structure formula generated for the second-order factor model is

$$\mu = F_1 F_2 v + 0 + u$$

which is what you expect for the mean structures of the second-order factor model. To complete the COSAN model specification, you can use MATRIX statements to specify the parameters and fixed values in the COSAN model matrices. See Example 29.29 for a complete example.

### Special Cases of the Generalized COSAN Model

It is illustrative to see how you can view different types of models as a special case of the generalized COSAN model. This section describes two such special cases.

#### The Original COSAN Model

The original COSAN (covariance structure analysis) model (McDonald 1978, 1980) specifies the following covariance structures:

$$\Sigma = F_1 \cdots F_n PP'_n \cdots F'_1$$

This is the generalized COSAN with only one term for the covariance structure model formula. Hence, using the COSAN statement to specify the original COSAN model is straightforward.

#### Reticular Action Model

The RAM (McArdle 1980; McArdle and McDonald 1984) model fits the covariance structures

$$\Sigma_a = (I - A)^{-1} P (I - A)^{-1}$$

where $\Sigma_a$ is the symmetric covariance for all latent and observed variables in the RAM model, $A$ is a square matrix for path coefficients, $I$ is an identity matrix with the same dimensions as $A$, and $P$ is a symmetric covariance matrix. For details about the RAM model, see the section “The RAM Model” on page 1425.

Correspondingly, the RAM model fits the mean structure formula

$$\mu_a = (I - A)^{-1} w$$

where $\mu_a$ is the mean vector for all latent and observed variables in the RAM model and $w$ is a vector for mean or intercepts of the variables.
To extract the covariance and mean structures for the observed variables, a selection matrix $G$ is used. The selection matrix $G$ contains zeros and ones as its elements. Each row of $G$ has exactly one nonzero element at the position that corresponds to the location of a manifest row variable in $\Sigma_a$ or $\mu_a$. The covariance structure formula for the observed variables in the RAM model becomes

$$\Sigma = G(I - A)^{-1}P(I - A)^{-1}G'$$

The mean structure formula for the observed variables in the RAM model becomes

$$\mu = G(I - A)^{-1}w$$

These formulas suggest that the RAM model is a special case of the generalized COSAN model with one term. For example, suppose that there are 10 observed variables ($\text{var1} - \text{var10}$) and 3 latent variables in a RAM model. The following COSAN statement represents the RAM model:

```
cosan var= v1-v10,
    G(13,GEN) * A(13,GEN,IMI) * P(13,SYM) [Mean = w];
```

In the COSAN statement, you define the 10 variables in the VAR= option. Next, you provide the formulas for the mean and covariance structures. $G$ is a $10 \times 13$ general matrix (GEN), $A$ is a $13 \times 13$ general matrix with the IMI transformation (that is, $(I - A)^{-1}$), $P$ is a $13 \times 13$ symmetric matrix (SYM), and $w$ is a $13 \times 1$ vector. With these COSAN statement specifications, your mean and covariance structure formulas represent exactly those of the RAM model. To complete the entire model specification, your next step is to use the MATRIX statements to specify the parameters and fixed values in the model matrices $G$, $A$, $P$, and $w$.

Similarly, it is possible to use the COSAN modeling language to represent any other model types such as models defined by the FACTOR, LINEQS, LISMOD, MSTRUCT, PATH, and RAM statements. But this is not an automatic recommendation of using the COSAN modeling languages in all situations. When an analysis can be specified by either the COSAN or a more specific modeling language (for example, PATH), you should consider using the specific modeling language because the specific modeling language can exploit specific model features so that it does the following:

- enables more supplemental analysis (effect analysis, standardized solutions, and so on), which COSAN has no general way to display
- supports better initial estimation methods (the COSAN model can only set initial estimates to certain default or random values)
- leads to more efficient computations due to the availability of more specific formulas and algorithms

Certainly, the COSAN modeling language is still very useful when you fit some nonstandard model structures that cannot be handled otherwise by the more specific modeling languages.

### Naming Variables in the COSAN Model

Although you can define the list of observed (manifest) variables in the VAR= option of the COSAN statement, the COSAN modeling language does not support a direct specification of the latent or error variables in the model. In the COSAN statement, you can define the model matrices and how they multiply together to form the covariance and mean structures. However, except for the row variables of the first matrix in each term, you do not need to identify the row and column variables in all other matrices. However, you can use
the **VARNAMES** statement to *label* the column variables of the matrices. The names in the **VARNAMES** statement follow the general naming rules required by the general SAS system. They should not contain special characters and cannot be longer than 32 characters. Also, they do not need to use certain prefixes like what the LINEQS modeling language requires. It is important to realize that the **VARNAME** statement only *labels*, but does not *identify*, the column variables (and the row variables, by propagation). This means that while keeping all other things equal, changing the names in the **VARNAMES** statements does not change the mathematical model or the estimation of the model. For example, you can label all columns of a COSAN matrix with the same name but it does not mean that these columns refer to the same variable in the model. See the section “Naming Variables and Parameters” on page 1433 for the general rules about naming variables and parameters.

**Default Parameters in the COSAN Model**

The default parameters of the COSAN model matrices depend on the types of the matrices. Each element of the IDE or ZID matrix (identity matrix with or without an additional zero matrix) is either a fixed one or a fixed zero. You cannot override the default parameter values of these fixed matrices. For COSAN model matrices with types other than IDE or ZID, all elements are fixed zeros by default. You can override these default zeros by specifying them explicitly in the **MATRIX** statements.

**The FACTOR Model**

The FACTOR modeling language is used for specifying exploratory and confirmatory factor analysis models. You can use other general modeling languages such as LINEQS, LISMOD, PATH, and RAM to specify a factor model. But the FACTOR modeling language is more convenient for specifying factor models and is more specialized in displaying factor-analytic results. For convenience, models specified by the FACTOR modeling language are called FACTOR models.

**Types of Variables in the FACTOR Model**

Each variable in the FACTOR model is either manifest or latent. Manifest variables are those variables that are measured in the research. They must be present in the input data set. Latent variables are not directly observed. Each latent variable in the FACTOR model can be either a factor or an error term.

Factors are unmeasured hypothetical constructs for explaining the covariances among manifest variables, while errors are the unique parts of the manifest variables that are not explained by the (common) factors.

In the FACTOR model, all manifest variables are endogenous, which means that they are predicted from the latent variables. In contrast, all latent variables in the FACTOR model are exogenous, which means that they serve as predictors only.

**Naming Variables in the FACTOR Model**

Manifest variables in the FACTOR model are referenced in the input data set. In the FACTOR model specification, you use their names as they appear in the input data set. Manifest variable names must not be longer than 32 characters. There are no further restrictions on these names beyond those required by the SAS System.

Error variables in the FACTOR model are not named explicitly, although they are assumed in the model. You can name latent factors only in confirmatory FACTOR models. Factor names must not be longer than 32
The FACTOR Model

Characters and must be distinguishable from the manifest variable names in the same analysis. You do not need to name factors in exploratory FACTOR models, however. Latent factors named Factor1, Factor2, and so on are generated automatically in exploratory FACTOR models.

Model Matrices in the FACTOR Model

Suppose in the FACTOR model that there are \( p \) manifest variables and \( n \) factors. The FACTOR model matrices are described in the following subsections.

**Matrix \( \mathbf{F} \) \((p \times n)\) : Factor Loading Matrix**

The rows of \( \mathbf{F} \) represent the \( p \) manifest variables, while the columns represent the \( n \) factors. Each row of \( \mathbf{F} \) contains the factor loadings of a variable on all factors in the model.

**Matrix \( \mathbf{P} \) \((n \times n)\) : Factor Covariance Matrix**

The \( \mathbf{P} \) matrix is a symmetric matrix for the variances of and covariances among the \( n \) factors.

**Matrix \( \mathbf{U} \) \((p \times p)\) : Error Covariance Matrix**

The \( \mathbf{U} \) matrix represents a \( p \times p \) diagonal matrix for the error variances for the manifest variables. Elements in this matrix are the parts of variances of the manifest variables that are not explained by the common factors. Note that all off-diagonal elements of \( \mathbf{U} \) are fixed zeros in the FACTOR model.

**Vector \( \mathbf{a} \) \((p \times 1)\) : Intercepts**

If the mean structures are analyzed, vector \( \mathbf{a} \) represents the intercepts of the manifest variables.

**Vector \( \mathbf{v} \) \((n \times 1)\) : Factor Means**

If the mean structures are analyzed, vector \( \mathbf{v} \) represents the means of the factors.

Matrix Representation of the FACTOR Model

Let \( \mathbf{y} \) be a \( p \times 1 \) vector of manifest variables, \( \mathbf{\xi} \) be an \( n \times 1 \) vector of latent factors, and \( \mathbf{e} \) be a \( p \times 1 \) vector of errors. The factor model is written as

\[
\mathbf{y} = \mathbf{a} + \mathbf{F}\mathbf{\xi} + \mathbf{e}
\]

With the model matrix definitions in the previous section, the covariance matrix \( \mathbf{\Sigma} \) \((p \times p)\) of manifest variables is structured as

\[
\mathbf{\Sigma} = \mathbf{FPFP'} + \mathbf{U}
\]

The mean vector \( \mathbf{\mu} \) \((p \times p)\) of manifest variables is structured as

\[
\mathbf{\mu} = \mathbf{a} + \mathbf{Fv}
\]

Exploratory Factor Analysis Models

Traditionally, exploratory factor analysis is applied when the relationships of manifest variables with factors have not been well-established in research. All manifest variables are allowed to have nonzero loadings on the factors in the model. First, factors are extracted and an initial solution is obtained. Then, for ease
of interpretation a final factor solution is usually derived by rotating the factor space. Factor-variable relationships are determined by interpreting the final factor solution. This is different from the confirmatory factor analysis in which the factor-variable relationships are prescribed and to be confirmed.

So far, confirmatory and exploratory models are not distinguished in deriving the covariance and mean structures. These two types of models are now distinguished in terms of the required structures or restrictions in model matrices.

In PROC CALIS, the initial exploratory factor solution is obtained from a specific confirmatory factor model with restricted model matrices, which are described as follows:

- The factor loading matrix $F$ has $n \times (n - 1)/2$ fixed zeros at the upper triangle portion of the matrix.
- The factor covariance matrix $P$ is an identity matrix, which means that factors are not correlated.
- The error covariance matrix $U$ is a diagonal matrix.
- Except for METHOD=FIML or METHOD=LSFIML and robust methods with the ROBUST option, the mean structures are not modeled by default—that is, the intercept vector $a$ or the factor mean vector $v$ are not parameterized in the model.
- With METHOD=FIML or METHOD=LSFIML, the application of the robust methods with the ROBUST option, the use of the MEAN statement, or the specification of the MEANSTR option, the mean structures are modeled. The intercept vector $a$ contains $p$ free parameters, and the factor mean vector $v$ is a zero vector.

The intercept vector $a$ is parameterized in the FIML method because the first-order moments (that is, the variable means) of the data have to be analyzed with the FIML treatment of the incomplete observations. For robust methods, the observations are reweighted during estimation, so the intercept vector $a$ must also be parameterized to accommodate the use of robust means in computing robust covariances. Estimation that is done using methods other than FIML or without the robust methods usually ignores the analysis of the mean structures because they are saturated and do not affect the fitting of covariance structures.

With the exploratory factor specification, you do not need to specify the patterns of the model matrices. PROC CALIS automatically sets up the correct patterns for the model matrices. For example, for an analysis with nine variables and three factors, the relevant model matrices of an exploratory FACTOR model have the following patterns, where * denotes free parameters in the model matrices:

$$
F = \begin{pmatrix}
* & 0 & 0 \\
* & * & 0 \\
* & * & * \\
* & * & * \\
* & * & * \\
* & * & * \\
* & * & * \\
* & * & * \\
* & * & *
\end{pmatrix}
$$

$$
P = \begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}
$$
and

\[
U = \begin{pmatrix}
* & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & * & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & * & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & * & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & * & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & * & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & * & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & *
\end{pmatrix}
\]

If METHOD=FIML or METHOD=LSFIML, the elements of the intercept vector \( a \) are all free parameters, as shown in the following:

\[
a = \begin{pmatrix}
* \\
* \\
* \\
* \\
* \\
*
\end{pmatrix}
\]

The factor mean vector \( v \) is a fixed zero vector.

If an initial factor solution is rotated afterward, some of these matrix patterns are changed. In general, rotating a factor solution eliminates the fixed zero pattern in the upper triangle of the factor loading matrix \( F \). If you apply an orthogonal rotation, the factor covariance matrix \( P \) does not change. It is an identity matrix before and after rotation. However, if you apply an oblique rotation, in general the rotated factor covariance matrix \( P \) is not an identity matrix and the off-diagonal elements are not zeros.

The error covariance matrix \( U \) remains unchanged after rotation. That is, it would still be a diagonal matrix. For the FIML estimation, the rotation does not affect the estimation of the intercept vector \( a \) and the fixed factor mean vector \( v \).

**Confirmatory Factor Analysis Models**

In confirmatory FACTOR models, there are no imposed patterns on the \( F, P, a, \) and \( v \) model matrices. All elements in these model matrices can be specified. However, for model identification, you might need to specify some factor loadings or factor variances as constants.

The only model restriction in confirmatory FACTOR models is placed on \( U \), which must be a diagonal matrix, as in exploratory FACTOR models too.
For example, for a confirmatory factor analysis with nine variables and three factors, you might specify the following patterns for the model matrices, where * denotes free parameters in the model matrices:

\[
F = \begin{pmatrix}
1 & 0 & 0 \\
* & 0 & 0 \\
* & 0 & 0 \\
0 & 1 & 0 \\
0 & * & 0 \\
0 & * & 0 \\
0 & 0 & 1 \\
0 & 0 & * \\
0 & 0 & *
\end{pmatrix}
\]

\[
P = \begin{pmatrix}
* & * & * \\
* & * & * \\
* & * & *
\end{pmatrix}
\]

and

\[
U = \begin{pmatrix}
* & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & * & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & * & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & * & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & * & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & * & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & * & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & *
\end{pmatrix}
\]

In this confirmatory factor model, mean structures are not modeled. In addition, there are some distinctive features that underscore the differences between confirmatory and exploratory models:

- Factor loading matrix \( F \) contains mostly zero elements and few nonzero free parameters, a pattern which is seen in most confirmatory factor models. In contrast, in exploratory factor models most elements in the \( F \) matrix are nonzero parameters.

- Factor loading matrix \( F \) contains fixed values of ones. These fixed values are used for model identification purposes (that is, identifying the scales of the latent variables). In general, you always have to make sure that your confirmatory factor models are identified by putting fixed values in appropriate parameter locations in the model matrices. However, this is not a concern in exploratory FACTOR models because identification has been ensured by imposing certain patterns on the model matrices.

- The nonzero off-diagonal parameters in the factor covariance matrix \( P \) indicate that correlated factors are hypothesized in the confirmatory factor model. This cannot be the case with the initial model of exploratory FACTOR models, where the \( P \) matrix must be an identity matrix before rotation.
**Summary of Matrices in the FACTOR Model**

Let \( p \) be the number of manifest variables and \( n \) be the number of factors in the FACTOR model. The names, roles, and dimensions of the FACTOR model matrices are shown in the following table.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Name</th>
<th>Description</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td><em>FACTLOAD</em></td>
<td>Factor loading matrix</td>
<td>( p \times n )</td>
</tr>
<tr>
<td>P</td>
<td><em>FACTFCOV</em></td>
<td>Factor covariance matrix</td>
<td>( n \times n )</td>
</tr>
<tr>
<td>U</td>
<td><em>FACTERRV</em></td>
<td>Error covariance matrix</td>
<td>( p \times p )</td>
</tr>
<tr>
<td>a</td>
<td><em>FACTINTE</em></td>
<td>Intercepts</td>
<td>( p \times 1 )</td>
</tr>
<tr>
<td>v</td>
<td><em>FACTMEAN</em></td>
<td>Factor means</td>
<td>( n \times 1 )</td>
</tr>
</tbody>
</table>

**Specification of the Exploratory Factor Model**

Because all initial model matrices of exploratory FACTOR models are predefined in PROC CALIS, you do not need to specify any other parameters in the model matrices. To obtain desired factor solutions, you can use various options for exploratory factor analysis in the FACTOR statement. These options are the *EFA-options* in the FACTOR statement. Two main types of *EFA-options* are shown as follows:

- options for factor extraction: COMPONENT, HEYWOOD, and N=.
- options for factor rotation: GAMMA=, NORM=, RCONVERGE=, RITER=, ROTATE=, and TAU=.

For example, the following statement requests that three factors be extracted, followed by a varimax rotation of the initial factor solution:

```plaintext
factor n=3 rotate=varimax;
```

See the FACTOR statement on page 1258 for details about the *EFA-options*.

**Specification of the Confirmatory Factor Model**

To specify a confirmatory FACTOR model, you specify the factor-variable relationships in the FACTOR statement, the factor variances and error variances in the PVAR statement, the factor covariances in the COV statement, and the means and intercepts in the MEAN statement.

**Specification of Factor-Variable Relationships**

The *CFA-spec* in the FACTOR statement is for specifying the factor-variables relationships. For example, in the following statement you specify three factors \( F1 \), \( F2 \), and \( F3 \) that are related to different clusters of observed variables \( V1 \)–\( V9 \):

```plaintext
factor
  F1 ===> V1-V3 = 1. parm1 (.4) parm2 (.4),
  F2 ===> V4-V6 = 1. parm3 parm4,
  F3 ===> V7-V9 = 1. parm5 parm6 (.3);
```

In the specification, variable \( V1 \) has a fixed loading of 1.0 on \( F1 \). Variables \( V2 \) and \( V3 \) have loadings on \( F1 \) also. These two loadings are free parameters named \( \text{parm1} \) and \( \text{parm2} \), respectively. Initial estimates can be set in parentheses after the free parameters. For example, both \( \text{parm1} \) and \( \text{parm2} \) have initial values at 0.4. Similarly, relationships of factor \( F2 \) with \( V4 \)–\( V6 \) and of factor \( F3 \) with \( V7 \)–\( V9 \) are defined in the
same FACTOR statement. Providing initial estimates for parameters is optional. In this example, parm3, parm4, and parm5 are all free parameters without initial values provided. PROC CALIS can determine appropriate initial estimates for these parameters. See the descriptions of CFA-spec in the FACTOR statement on page 1258 for more details about the syntax.

**Specification of Factor Variances and Error Variances**
You can specify the factor variances and error variances in the PVAR statement. For example, consider the following statement:

```
pvar F1-F3 = fvar1-fvar3,
     V1-V9 = evar1-evar9 (9*10.);
```

In the PVAR statement, you specify the variances of factors F1, F2, and F3 as free parameters fvar1, fvar2, and fvar3, respectively, and the error variances for manifest variables V1–V9 as free parameters evar1–evar9, respectively. Each of the error variance parameters is given a starting value at 10. See the PVAR statement on page 1345 for more details about the syntax.

**Specification of Factor Covariances**
You can specify the factor covariances in the COV statement. For example, you specify the covariances among factors F1, F2, and F3 in the following statement:

```
cov F1 F2 = cov12,
       F1 F3 = cov13,
       F2 F3 = cov23;
```

The covariance parameters are named cov12, cov13, and cov23, respectively. They represent the lower triangular elements of the factor covariance matrix P. See the COV statement on page 1252 for more details about the syntax.

**Specification of Means and Intercepts**
If mean structures are of interest, you can also specify the factor means and the intercepts for the manifest variables in the MEAN statement. For example, consider the following statement:

```
mean F1-F3 = fmean1-fmean3,
     V1-V9 = 9*12.;
```

In this statement, you specify the factor means of F1, F2, and F3 as free parameters fmean1, fmean2, and fmean3, respectively, and the intercepts for variables V1–V9 as fixed parameters at 12. See the MEAN statement on page 1309 for more details about the syntax.

**Naming the Factors**
For the exploratory FACTOR model, PROC CALIS generates the names for the factors automatically. For the confirmatory FACTOR model, you can specify the names for the factors. Unlike the LINEQS model, in the confirmatory FACTOR model you do not need to use the ‘F’ or ‘f’ prefix to denote factors in the model. You can use any valid SAS variable names for the factors, especially those names that reflect the nature of the factors. To avoid confusions with other names in the model, some general rules are recommended. See the section “Naming Variables and Parameters” on page 1433 for these general rules about naming variables and parameters.
Default Parameters in the FACTOR Model

Default parameters in the FACTOR model are different for exploratory and confirmatory factor models.

For the exploratory FACTOR model, all fixed and free parameters of the model are prescribed. These prescribed parameters include a fixed pattern for the factor loading matrix $F$, a diagonal pattern for the error variance matrix $U$, and an identity matrix for factor covariance matrix $P$. This means that factors are uncorrelated in the estimation. However, if you specify an oblique rotation after the estimation of the factor solution, the factors could become correlated. See the section “Exploratory Factor Analysis Models” on page 1395 for more details about the patterns of the exploratory FACTOR model. Because all these patterns are prescribed, you cannot override any of these parameters for the exploratory FACTOR model.

For the confirmatory FACTOR model, the set of default free parameters of the confirmatory FACTOR model includes the following:

- the error variances of the observed variables; these correspond to the diagonal elements of the uniqueness matrix $U$
- the variances and covariances among the factors; these correspond to all elements of the factor covariance matrix $P$
- the intercepts of the observed variables if the mean structures are modeled; these correspond to all elements of the intercept vector $a$

PROC CALIS names the default free parameters with the _Add prefix (or the _a prefix for the case of default intercept parameters in $a$ for exploratory factor models), followed by a unique integers for each parameter. Except for the exploratory factor model, you can override the default free parameters by explicitly specifying them as free, constrained, or fixed parameters in the COV, MEAN, or PVAR statement.

In addition to default free parameters, another type of default parameter is the fixed zeros applied to the unspecified parameters in the loading matrix $F$ and the factor means in the $v$ vector. Certainly, you use the FACTOR and MEAN specifications to override those default zero loadings or factor means and set them to free, constrained, or fixed parameters. Notice that the uniqueness matrix $U$ in the confirmatory factor model is a diagonal element. You cannot specify any of its off-diagonal elements—they are always fixed zeros by the model restriction.

The LINEQS Model

The LINEQS modeling language is adapted from the EQS (equations) program by Bentler (1995). The statistical models that LINEQS or EQS analyzes are essentially the same as other general modeling languages such as LISMOD, RAM, and PATH. However, the terminology and approach of the LINEQS or EQS modeling language are different from other languages. They are based on the theoretical model developed by Bentler and Weeks (1980). For convenience, models that are analyzed using the LINEQS modeling language are called LINEQS models. Note that these so-called LINEQS models can also be analyzed by other general modeling languages in PROC CALIS.
In the LINEQS (or the original EQS) model, relationships among variables are represented by a system of equations. For example:

\[ Y_1 = a_0 + a_1 X_1 + a_2 X_2 + E_1 \]
\[ Y_2 = b_0 + b_1 X_1 + b_2 Y_1 + E_2 \]

On the left-hand side of each equation, an outcome variable is hypothesized to be a linear function of one or more predictor variables and an error, which are all specified on the right-hand side of the equation. The parameters specified in an equation are the effects (or regression coefficients) of the predictor variables. For example, in the preceding equations, \( Y_1 \) and \( Y_2 \) are outcome variables; \( E_1 \) and \( E_2 \) are error variables; \( a_1 \), \( a_2 \), \( b_1 \), and \( b_2 \) are effect parameters (or regression coefficients); and \( a_0 \) and \( b_0 \) are intercept parameters. Variables \( X_1 \) and \( X_2 \) serve as predictors in the first equation, while variables \( X_1 \) and \( Y_1 \) serve as predictors in the second equation.

This is almost the same representation as in multiple regression models. However, the LINEQS model entails more. It supports a system of equations that can also include latent variables, measurement errors, and correlated errors.

**Types of Variables in the LINEQS Model**

The distinction between dependent and independent variables is important in the LINEQS model.

A variable is dependent if it appears on the left-hand side of an equation in the model. A dependent variable might be observed (manifest) or latent. It might or might not appear on the right-hand side of other equations, but it cannot appear on the left-hand sides of two or more equations. Error variables cannot be dependent in the LINEQS model.

A variable in the LINEQS model is independent if it is not dependent. Independent variables can be observed (manifest) or latent. All error variables must be independent in the LINEQS model.

Dependent variables are also referred to as endogenous variables; these names are interchangeable. Similarly, independent variables are interchangeable with exogenous variables.

Whereas an outcome variable in any equation must be a dependent variable, a predictor variable in an equation is not necessarily an independent variable in the entire LINEQS model. For example, \( Y_1 \) is a predictor variable in the second equation of the preceding example, but it is a dependent variable in the LINEQS model.

In summary, the predictor-outcome nature of a variable is determined within a single equation, while the exogenous-endogenous (independent-dependent) nature of variable is determined within the entire system of equations.

In addition to the dependent-independent variable distinction, variables in the LINEQS model are distinguished according to whether they are observed in the data. Variables that are observed in research are called observed or manifest variables. Hypothetical variables that are not observed in the LINEQS model are latent variables.

Two types of latent variables should be distinguished: one is error variables; the other is non-error variables. An error variable is unique to an equation. It serves as the unsystematic source of effect for the outcome variable in an equation. If the outcome variable in the equation is latent, the corresponding error variable is also called disturbance. In contrast, non-error or systematic latent variables are called factors. Factors are unmeasured hypothetical constructs in your model. They are systematic sources that explain or describe functional relationships in your model.

Both manifest variables and latent factors can be dependent or independent. However, error or disturbance terms must be independent (or exogenous) variables in your model.
Naming Variables in the LINEQS Model

Whether a variable in each equation is an outcome or a predictor variable is prescribed by the modeler. Whether a variable is independent or dependent can be determined by analyzing the entire system of equations in the model. Whether a variable is observed or latent can be determined if it is referenced in your data set. However, whether a latent variable serves as a factor or an error can be determined only if you provide the specific information.

To distinguish latent factors from errors and both from manifest variables, the following rules for naming variables in the LINEQS model are followed:

- Manifest variables are referenced in the input data set. You use their names in the LINEQS model specification directly. There is no additional naming rule for the manifest variables in the LINEQS model beyond those required by the SAS System.

- Latent factor variables must start with letter F or f (for factor).

- Error variables must start with letter E or e (for error), or D or d (for disturbance). Although you might enforce the use of D- (or d-) variables for disturbances, it is not required. For flexibility, disturbance variables can also start with letter E or e in the LINEQS model.

- The names of latent variables, errors, and disturbances (F-, E-, and D-variables) should not coincide with the names of manifest variables.

- You should not use Intercept as a name for any variable. This name is reserved for the intercept specification in LINEQS model equations.

See the section “Naming Variables and Parameters” on page 1433 for the general rules about naming variables and parameters.

Matrix Representation of the LINEQS Model

As a programming language, the LINEQS model uses equations to describes relationships among variables. But as a mathematical model, the LINEQS model is more conveniently described by matrix terms. In this section, the LINEQS matrix model is described.

Suppose in a LINEQS model that there are \( n_i \) independent variables and \( n_d \) dependent variables. The vector of the independent variables is denoted by \( \xi \), in the order of manifest variables, latent factors, and error variables. The vector of dependent variables is denoted by \( \eta \), in the order of manifest variables and latent factors. The LINEQS model matrices are defined as follows:

\[
\begin{align*}
\alpha & (n_d \times 1) : \text{intercepts of dependent variables} \\
\beta & (n_d \times n_d) : \text{effects of dependent variables (in columns) on dependent variables (in rows)} \\
\gamma & (n_d \times n_i) : \text{effects of independent variables (in columns) on dependent variables (in rows)} \\
\Phi & (n_i \times n_i) : \text{covariance matrix of independent variables} \\
\nu & (n_i \times 1) : \text{means of independent variables}
\end{align*}
\]
Chapter 29: The CALIS Procedure

The model equation of the LINEQS model is

\[ \eta = \alpha + \beta \eta + \gamma \xi \]

Assuming that \((I - \beta)\) is invertible, under the model the covariance matrix of all variables \((\eta', \xi')'\) is structured as

\[
\Sigma_a = \begin{pmatrix}
(I - \beta)^{-1} \gamma \Phi y' (I - \beta)^{-1}' & (I - \beta)^{-1} \gamma \Phi \\
\Phi y' (I - \beta)^{-1}' & \Phi
\end{pmatrix}
\]

The mean vector of all variables \((\eta', \xi')'\) is structured as

\[
\mu_a = \begin{pmatrix}
(I - \beta)^{-1} (\alpha + \gamma v) \\
v + \gamma v
\end{pmatrix}
\]

As is shown in the structured covariance and mean matrices, the means \(\mathbf{G}\) and covariances of independent variables are direct model parameters in \(v\) and \(\Phi\); whereas the means and covariances of dependent variables are functions of various model matrices and hence functions of model parameters.

The covariance and mean structures of all observed variables are obtained by selecting the elements in \(\Sigma_a\) and \(\mu_a\). Mathematically, define a selection matrix \(\mathbf{G}\) of dimensions \(n \times (n_d + n_i)\), where \(n\) is the number of observed variables in the model. The selection matrix \(\mathbf{G}\) contains zeros and ones as its elements. Each row of \(\mathbf{G}\) has exactly one nonzero element at the position that corresponds to the location of an observed row variable in \(\Sigma_a\) or \(\mu_a\). With each row of \(\mathbf{G}\) selecting a distinct observed variable, the structured covariance matrix of all observed variables is represented by

\[
\Sigma = \mathbf{G} \Sigma_a \mathbf{G}'
\]

The structured mean vector of all observed variables is represented by

\[
\mu = \mathbf{G} \mu_a
\]

Partitions of Some LINEQS Model Matrices and Their Restrictions

There are some restrictions in some of the LINEQS model matrices. Although these restrictions do not affect the derivation of the covariance and mean structures, they are enforced in the LINEQS model specification.

**Model Restrictions on the \(\beta\) Matrix**

The diagonal of the \(\beta\) matrix must be zeros. This prevents the direct regression of dependent variables on themselves. Hence, in the LINEQS statement you cannot specify the same variable on both the left-hand and the right-hand sides of the same equation.

**Partitions of the \(\gamma\) Matrix and the Associated Model Restrictions**

The columns of the \(\gamma\) matrix refer to the variables in \(\xi\), in the order of manifest variables, latent factors, and error variables. In the LINEQS model, the following partition of the \(\gamma\) matrix is assumed:

\[
\gamma = \begin{pmatrix}
\gamma_0 & \mathbf{E}
\end{pmatrix}
\]

where \(\gamma_0\) is an \(n_d \times (n_i - n_d)\) matrix for the effects of independent manifest variables and latent factors on the dependent variables and \(\mathbf{E}\) is an \(n_d \times n_d\) permutation matrix for the effects of errors on the dependent variables.
The dimension of submatrix $E$ is $n_d \times n_d$ because in the LINEQS model each dependent variable signifies an equation with an error term. In addition, because $E$ is a permutation matrix (which is formed by exchanging rows of an identity matrix of the same order), the partition of the $\gamma$ matrix ensures that each dependent variable is associated with a unique error term and that the effect of each error term on its associated dependent variable is 1.

As a result of the error term restriction, in the LINEQS statement you must specify a unique error term in each equation. The coefficient associated with the error term can only be a fixed value at one, either explicitly (with 1.0 inserted immediately before the error term) or implicitly (with no coefficient specified).

**Partitions of the $\nu$ Vector and the Associated Model Restrictions**

The $\nu$ vector contains the means of independent variables, in the order of the manifest, latent factor, and error variables. In the LINEQS model, the following partition of the $\nu$ vector is assumed:

$$
\nu = \begin{pmatrix}
\nu_0 \\
0
\end{pmatrix}
$$

where $\nu_0$ is an $(n_i - n_d) \times 1$ vector for the means of independent manifest variables and latent factors and 0 is a null vector of dimension $n_d$ for the means of errors or disturbances. Again, the dimension of the null vector is $n_d$ because each dependent variable is associated uniquely with an error term. This partition restricts the means of errors or disturbances to zeros.

Hence, when specifying a LINEQS model, you cannot specify the means of errors (or disturbances) as free parameter or fixed values other than zero in the MEAN statement.

**Partitions of the $\Phi$ matrix**

The $\Phi$ matrix is for the covariances of the independent variables, in the order of the manifest, latent factor, and error variables. The following partition of the $\Phi$ matrix is assumed:

$$
\Phi = \begin{pmatrix}
\Phi_{11} & \Phi_{21}' \\
\Phi_{21} & \Phi_{22}
\end{pmatrix}
$$

where $\Phi_{11}$ is an $(n_i - n_d) \times (n_i - n_d)$ covariance matrix for the independent manifest variables and latent factors, $\Phi_{22}$ is an $n_d \times n_d$ covariance matrix for the errors, and $\Phi_{21}$ is an $n_d \times (n_i - n_d)$ covariance matrix for the errors with other independent variables in the LINEQS model. Because $\Phi$ is symmetric, $\Phi_{11}$ and $\Phi_{22}$ are also symmetric.

There are actually no model restrictions placed on the submatrices of the partition. However, in most statistical applications, errors represent unsystematic sources of effects and therefore they are not to be correlated with other systematic sources. This implies that submatrix $\Phi_{21}$ is a null matrix. However, $\Phi_{21}$ being null is not enforced in the LINEQS model specification. If you ever specify a covariance between an error variable and a non-error independent variable in the COV statement, as a workaround trick or otherwise, you should provide your own theoretical justifications.

**Summary of Matrices and Submatrices in the LINEQS Model**

Let $n_d$ be the number of dependent variables and $n_i$ be the number of independent variables. The names, roles, and dimensions of the LINEQS model matrices and submatrices are summarized in the following table.
### Model Matrices

<table>
<thead>
<tr>
<th>Matrix Name</th>
<th>Description</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha )</td>
<td><em>EQUALPHA</em></td>
<td>( n_d \times 1 )</td>
</tr>
<tr>
<td>( \beta )</td>
<td><em>EQUIS</em></td>
<td>( n_d \times n_d )</td>
</tr>
<tr>
<td>( \gamma )</td>
<td><em>EQUISGAMMA</em></td>
<td>( n_d \times n_i )</td>
</tr>
<tr>
<td>( \nu )</td>
<td><em>EQUISNU</em></td>
<td>( n_i \times 1 )</td>
</tr>
<tr>
<td>( \Phi )</td>
<td><em>EQUISPHI</em></td>
<td>( n_i \times n_i )</td>
</tr>
</tbody>
</table>

### Submatrices

<table>
<thead>
<tr>
<th>Submatrix Name</th>
<th>Description</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \gamma_0 )</td>
<td><em>EQUISGAMMA_SUB</em></td>
<td>( n_d \times (n_i - n_d) )</td>
</tr>
<tr>
<td>( \nu_0 )</td>
<td><em>EQUISNU_SUB</em></td>
<td>( (n_i - n_d) \times 1 )</td>
</tr>
<tr>
<td>( \Phi_{11} )</td>
<td><em>EQUISPHI11</em></td>
<td>( (n_i - n_d) \times (n_i - n_d) )</td>
</tr>
<tr>
<td>( \Phi_{21} )</td>
<td><em>EQUISPHI21</em></td>
<td>( n_d \times (n_i - n_d) )</td>
</tr>
<tr>
<td>( \Phi_{22} )</td>
<td><em>EQUISPHI22</em></td>
<td>( n_d \times n_d )</td>
</tr>
</tbody>
</table>

### Specification of the LINEQS Model

#### Specification in Equations

In the LINEQS statement, you specify intercepts and effect parameters (or regression coefficients) along with the variable relationships in equations. In terms of model matrices, you specify the \( \alpha \) vector and the \( \beta \) and \( \gamma \) matrices in the LINEQS statement without using any matrix language.

For example:

\[
Y = b_0 + b_1 \times X_1 + b_2 \times F_2 + E_1
\]

In this equation, you specify \( Y \) as an outcome variable, \( X_1 \) and \( F_2 \) as predictor variables, and \( E_1 \) as an error variable. The parameters in the equation are the intercept \( b_0 \) and the path coefficients (or effects) \( b_1 \) and \( b_2 \).

This kind of model equation is specified in the LINEQS statement. For example, the previous equation translates into the following LINEQS statement specification:

```plaintext
lineqs Y = b0 * Intercept + b1 * X1 + b2 * F2 + E1;
```
If the mean structures of the model are not of interest, the intercept term can be omitted. The specification becomes:

```plaintext
lineqs Y = b1 * X1 + b2 * F2 + E1;
```

See the LINEQS statement on page 1275 for the details about the syntax.

Because of the LINEQS model restrictions (see the section “Partitions of Some LINEQS Model Matrices and Their Restrictions” on page 1404), you must also follow these rules when specifying LINEQS model equations:

- A dependent variable can appear only on the left-hand side of an equation once. In other words, you must put all predictor variables for a dependent variable in one equation. This is different from some econometric models where a dependent variable can appear on the left-hand sides of two equations to represent an equilibrium point. However, this limitation can be resolved by reparameterization in some cases. See Example 29.18.

- A dependent variable that appears on the left-hand side of an equation cannot appear on the right-hand side of the same equation. If you measure the same characteristic at different time points and the previous measurement serves as a predictor of the next measurement, you should use different variable names for the measurements so as to comply with this rule.

- An error term must be specified in each equation and must be unique. The same error name cannot appear in two or more equations. When an equation is truly intended to have no error term, it should be represented equivalently in the LINEQS equation by introducing an error term with zero variance (specified in the VARIANCE statement).

- The regression coefficient (effect) that is associated with an error term must be fixed at one (1.0). This is done automatically by omitting any fixed constants or parameters that are associated with the error terms. Inserting a parameter or a fixed value other than 1 immediately before an error term is not allowed.

**Mean, Variance, and Covariance Parameter Specification**

In addition to the intercept and effect parameters that are specified in equations, the means, variances, and covariances among all independent variables are parameters in the LINEQS model. An exception is that the means of all error variables are restricted to fixed zeros in the LINEQS model. To specify the mean, variance, and covariance parameters, you use the MEAN, VARIANCE, and the COV statements, respectively.

The means, variances, and covariances among dependent variables are not parameters themselves in the model. Rather, they are complex functions of the model parameters. See the section “Matrix Representation of the LINEQS Model” on page 1403 for mathematical details.

**Default Parameters in the LINEQS Model**

There are two types of default parameters of the LINEQS model, as implemented in PROC CALIS. One is the free parameters; the other is the fixed constants.

The following sets of parameters are free parameters by default:

- the variances of all exogenous (independent) observed or latent variables (including error and disturbance variables)
• the covariances among all exogenous (independent) manifest or latent variables (excluding error and disturbance variances)

• the means of all exogenous (independent) observed variables if the mean structures are modeled

• the intercepts of all endogenous (dependent) manifest variables if the mean structures are modeled

PROC CALIS names the default free parameters with the _Add prefix and a unique integer suffix. You can override the default free parameters by explicitly specifying them as free, constrained, or fixed parameters in the COV, LINEQS, MEAN, or VARIANCE statement.

Parameters that are not default free parameters in the LINEQS model are fixed constants by default. You can override almost all of the default fixed constants of the LINEQS model by using the COV, LINEQS, MEAN, or VARIANCE statement. You cannot override the following two sets of fixed constants:

• fixed zero parameters for the direct effects (path coefficients) of variables on their own. You cannot have an equation in the LINEQS statement that has the same variable specified on the left-hand and the right-hand sides.

• fixed one effects from the error or disturbance variables. You cannot set the path coefficient (effect) of the error or disturbance term to any value other than 1 in the LINEQS statement.

These two sets of fixed parameters reflect the LINEQS model restrictions so that they cannot be modified. Other than these two sets of default fixed parameters, all other default fixed parameters are zeros. You can override these default zeros by explicitly specifying them as free, constrained, or fixed parameters in the COV, LINEQS, MEAN, or VARIANCE statement.

The LISMOD Model and Submodels

As a statistical model, the LISMOD modeling language is derived from the LISREL model proposed by Jöreskog and others (see Keesling 1972; Wiley 1973; Jöreskog 1973). But as a computer language, the LISMOD modeling language is quite different from the LISREL program. To maintain the consistence of specification syntax within the CALIS procedure, the LISMOD modeling language departs from the original LISREL programming language. In addition, to make the programming a little easier, some terminological changes from LISREL are made in LISMOD.

For brevity, models specified by the LISMOD modeling language are called LISMOD models, although you can also specify these LISMOD models by other general modeling languages that are supported in PROC CALIS.

The following descriptions of LISMOD models are basically the same as those of the original LISREL models. The main modifications are the names for the model matrices.

The LISMOD model is described by three component models. The first one is the structural equation model that describes the relationships among latent constructs or factors. The other two are measurement models that relate latent factors to manifest variables.
The LISMOD Model and Submodels

Structural Equation Model

The structural equation model for latent factors is

\[ \eta = \alpha + \beta \eta + \Gamma \xi + \zeta \]

where:
- \( \eta \) is a random vector of \( n_\eta \) endogenous latent factors
- \( \xi \) is a random vector of \( n_\xi \) exogenous latent factors
- \( \zeta \) is a random vector of errors
- \( \alpha \) is a vector of intercepts
- \( \beta \) is a matrix of regression coefficients of \( \eta \) variables on other \( \eta \) variables
- \( \Gamma \) is a matrix of regression coefficients of \( \eta \) on \( \xi \)

There are some assumptions in the structural equation model. To prevent a random variable in \( \eta \) from regressing directly on itself, the diagonal elements of \( \beta \) are assumed to be zeros. Also, \( (I - \beta)^{-1} \) is assumed to be nonsingular, and \( \zeta \) is uncorrelated with \( \xi \).

The covariance matrix of \( \xi \) is denoted by \( \Psi \) and its expected value is a null vector. The covariance matrix of \( \xi \) is denoted by \( \Phi \) and its expected value is denoted by \( \kappa \).

Because variables in the structural equation model are not observed, to analyze the model these latent variables must somehow relate to the manifest variables. The measurement models, which are discussed in the subsequent sections, provide such relations.

Measurement Model for \( y \)

\[ y = \nu_y + \Lambda_y \eta + \epsilon \]

where:
- \( y \) is a random vector of \( n_y \) manifest variables
- \( \epsilon \) is a random vector of errors for \( y \)
- \( \nu_y \) is a vector of intercepts for \( y \)
- \( \Lambda_y \) is a matrix of regression coefficients of \( y \) on \( \eta \)

It is assumed that \( \epsilon \) is uncorrelated with either \( \eta \) or \( \zeta \). The covariance matrix of \( \epsilon \) is denoted by \( \Theta_y \) and its expected value is the null vector.

Measurement Model for \( x \)

\[ x = \nu_x + \Lambda_x \xi + \delta \]

where:
- \( x \) is a random vector of \( n_x \) manifest variables
- \( \delta \) is a random vector of errors for \( x \)
- \( \nu_x \) is a vector of intercepts for \( x \)
- \( \Lambda_x \) is a matrix of regression coefficients of \( x \) on \( \xi \)

It is assumed that \( \delta \) is uncorrelated with \( \xi \), \( \epsilon \), or \( \zeta \). The covariance matrix of \( \delta \) is denoted by \( \Theta_x \) and its expected value is a null vector.
Covariance and Mean Structures

Under the structural and measurement equations and the model assumptions, the covariance structures of the manifest variables \((y', x')\) are expressed as

\[
\Sigma = \begin{pmatrix}
\Lambda_y(I - \beta)^{-1}(\Gamma \Phi' + \Psi)(I - \beta')^{-1}\Lambda_y' + \Theta_y \\
\Lambda_x \Phi'(I - \beta')^{-1}\Lambda_x'
\end{pmatrix}
\]

The mean structures of the manifest variables \((y', x')\) are expressed as

\[
\mu = \begin{pmatrix}
v_y + \Lambda_y(I - \beta)^{-1}(\alpha + \Gamma \kappa) \\
v_x + \Lambda_x \kappa
\end{pmatrix}
\]

Model Matrices in the LISMOD Model

The parameters of the LISMOD model are elements in the model matrices, which are summarized as follows.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Name</th>
<th>Description</th>
<th>Dimensions</th>
<th>Row Variables</th>
<th>Column Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\alpha)</td>
<td><em>ALPHA</em></td>
<td>Intercepts for (\eta)</td>
<td>(n_\eta \times 1)</td>
<td>(\eta)</td>
<td>N/A (ETAVAR=)</td>
</tr>
<tr>
<td>(\beta)</td>
<td><em>BETA</em></td>
<td>Effects of (\eta) on (\eta)</td>
<td>(n_\eta \times n_\eta)</td>
<td>(\eta)</td>
<td>(\eta) (ETAVAR=)</td>
</tr>
<tr>
<td>(\Gamma)</td>
<td><em>GAMMA</em></td>
<td>Effects of (\xi) on (\eta)</td>
<td>(n_\eta \times n_\xi)</td>
<td>(\eta)</td>
<td>(\xi) (ETAVAR=)</td>
</tr>
<tr>
<td>(\Psi)</td>
<td><em>PSI</em></td>
<td>Error covariance matrix for (\eta)</td>
<td>(n_\eta \times n_\eta)</td>
<td>(\eta)</td>
<td>(\eta) (ETAVAR=)</td>
</tr>
<tr>
<td>(\Phi)</td>
<td><em>PHI</em></td>
<td>Covariance matrix for (\xi)</td>
<td>(n_\xi \times n_\xi)</td>
<td>(\xi)</td>
<td>(\xi) (ETAVAR=)</td>
</tr>
<tr>
<td>(\kappa)</td>
<td><em>KAPPA</em></td>
<td>Mean vector for (\xi)</td>
<td>(n_\xi \times 1)</td>
<td>(\xi)</td>
<td>N/A (XIVAR=)</td>
</tr>
<tr>
<td>(v_y)</td>
<td><em>NUY</em></td>
<td>Intercepts for (y)</td>
<td>(n_y \times 1)</td>
<td>(y)</td>
<td>N/A (YVAR=)</td>
</tr>
<tr>
<td>(\Lambda_y)</td>
<td><em>LAMBDAY</em></td>
<td>Effects of (\eta) on (y)</td>
<td>(n_y \times n_\eta)</td>
<td>(y)</td>
<td>(\eta) (YVAR=)</td>
</tr>
<tr>
<td>(\Theta_y)</td>
<td><em>THETAY</em></td>
<td>Error covariance matrix for (y)</td>
<td>(n_y \times n_y)</td>
<td>(y)</td>
<td>(y) (YVAR=)</td>
</tr>
<tr>
<td>(v_x)</td>
<td><em>NUX</em></td>
<td>Intercepts for (x)</td>
<td>(n_x \times 1)</td>
<td>(x)</td>
<td>N/A (XVAR=)</td>
</tr>
<tr>
<td>(\Lambda_x)</td>
<td><em>LAMBDAX</em></td>
<td>Effects of (\xi) on (x)</td>
<td>(n_x \times n_\xi)</td>
<td>(x)</td>
<td>(\xi) (XVAR=)</td>
</tr>
<tr>
<td>(\Theta_x)</td>
<td><em>THETAX</em></td>
<td>Error covariance matrix for (x)</td>
<td>(n_x \times n_x)</td>
<td>(x)</td>
<td>(x) (XVAR=)</td>
</tr>
</tbody>
</table>

There are twelve model matrices in the LISMOD model. Not all of them are used in all situations. See the section “LISMOD Submodels” on page 1413 for details. In the preceding table, each model matrix is given a name in the column Name, followed by a brief description of the parameters in the matrix, the dimensions,
and the row and column variables being referred to. In the second column of the table, the LISMOD matrix names are used in the MATRIX statements when specifying the LISMOD model. In the last two columns of the table, following the row or column variables is the variable list (for example, ETAVAR=, YVAR=, and so on) in parentheses. These lists are used in the LISMOD statement for defining variables.

Specification of the LISMOD Model

The LISMOD specification consists of two tasks. The first task is to define the variables in the model. The second task is to specify the parameters in the LISMOD model matrices.

Specifying Variables

The first task is accomplished in the LISMOD statement. In the LISMOD statement, you define the lists of variables of interest: YVAR=, XVAR=, ETAVAR=, and XIVAR= lists, respectively for the \( y \)-variables, \( x \)-variables, \( \eta \)-variables, and the \( \xi \)-variables. While you provide the names of variables in these lists, you also define implicitly the numbers of four types of variables: \( n_y \), \( n_x \), \( n_\eta \), and \( n_\xi \). The variables in the YVAR= and XVAR= lists are manifest variables and therefore must be present in the analyzed data set. The variables in the ETAVAR= and XIVAR= lists are latent factors, the names of which are assigned by the researcher to represent their roles in the substantive theory. After these lists are defined, the dimensions of the model matrices are also defined by the number of variables on various lists. In addition, the variable orders in the lists are referred to by the row and column variables of the model matrices.

Unlike the LINEQS model, in the LISMOD model you do not need to use the ‘F’ or ‘f’ prefix to denote factors in the ETAVAR= or XIVAR= list. You can use any valid SAS variable names for the factors, especially those names that reflect the nature of the factors. To avoid confusion with other names in the model, some general rules are recommended. See the section “Naming Variables and Parameters” on page 1433 for these general rules about naming variables and parameters.

Specifying Parameters in Model Matrices

The second task is accomplished by the MATRIX statements. In each MATRIX statement, you specify the model matrix by using the matrix names described in the previous table. Then you specify the parameters (free or fixed) in the locations of the model matrix. You can use as many MATRIX statements as needed for defining your model. But each model matrix can be specified only in one MATRIX statement, and each MATRIX statement is used for specifying only one model matrix.

An Example

In the section “LISMOD Model” on page 1189, the LISMOD modeling language is used to specify the model described in the section “A Structural Equation Example” on page 1184. In the LISMOD statement, you define four lists of variables, as shown in the following statement:

```sas
lismod
  yvar  = Anomie67 Powerless67 Anomie71 Powerless71,
  xvar  = Education SEI,
  etav  = Alien67 Alien71,
  xivar = SES;
```
Endogenous latent factors are specified in the ETAVAR= list. Exogenous latent factors are specified in the XIVAR= list. In this case, Alien67 and Alien71 are the $\eta$-variables, and SES is the only $\xi$-variable in the model. Manifest variables that are indicators of endogenous latent factors in $\eta$ are specified in the YVAR= list. In this case, they are the Anomie and Powerless variables, measured at two different time points. Manifest variables that are indicators of exogenous latent factors in $\xi$ are specified in the XVAR= list. In this case, they are the Education and the SEI variables. Implicitly, the dimensions of the model matrices are defined by these lists already; that is, $n_y = 4$, $n_x = 2$, $n_\eta = 2$, and $n_\xi = 1$.

The MATRIX statements are used to specify parameters in the model matrices. For example, in the following statement you define the $\_LAMBDAX_\_ (A_x)$ matrix with two nonzero entries:

\[
\text{matrix } \_LAMBDAX_\_ [1,1] = 1.0, \\
[2,1] = \text{lambda};
\]

The first parameter location is for $[1,1]$, which is the effect of SES (the first variable in the XIVAR= list) on Education (the first element in the XVAR= list). A fixed value of 1.0 is specified there. The second parameter location is for $[2,1]$, which is the effect of SES (the first variable in the XIVAR= list) on SEI (the second variable in the XVAR= list). A parameter named lambda without an initial value is specified there.

Another example is shown as follows:

\[
\text{matrix } \_THETAY_\_ [1,1] = \text{theta1}, \\
[2,2] = \text{theta2}, \\
[3,3] = \text{theta1}, \\
[4,4] = \text{theta2}, \\
[3,1] = \text{theta5}, \\
[4,2] = \text{theta5};
\]

In this MATRIX statement, the error variances and covariances (that is, the $\Theta_y$ matrix) for the $y$-variables are specified. The diagonal elements of the $\_THETAY_\_ $ matrix are specified by parameters theta1, theta2, theta1, and theta2, respectively, for the four $y$-variables Anomie67, Powerless67, Anomie71, and Powerless71. By using the same parameter name theta1, the error variances for Anomie67 and Anomie71 are implicitly constrained. Similarly, the error variances for Powerless67 and Powerless71 are also implicitly constrained. Two more parameter locations are specified. The error covariance between Anomie67 and Anomie71 and the error covariance between Powerless67 and Powerless71 are both represented by the parameter theta5. Again, this is an implicit constraint on the covariances. All other unspecified elements in the $\_THETAY_\_ $ matrix are treated as fixed zeros.

In this example, no parameters are specified for matrices $\_ALPHA_\_, \_KAPPA_\_, \_NUY_\_, \_NUX_\$. Therefore, mean structures are not modeled.
LISMOD Submodels

It is not necessary to specify all four lists of variables in the LISMOD statement. When some lists are unspecified in the LISMOD statement, PROC CALIS analyzes submodels derived logically from the specified lists of variables. For example, if only \( y \)- and \( x \)-variable lists are specified, the submodel being analyzed would be a multivariate regression model with manifest variables only. Not all combinations of lists lead to meaningful submodels, however. To determine whether and how a submodel (which is formed by a certain combination of variable lists) can be analyzed, the following three principles in the LISMOD modeling language are applied:

- Submodels with at least one of the YVAR= and XVAR= lists are required.
- Submodels that have an ETAVAR= list but no YVAR= list cannot be analyzed.
- When a submodel has a YVAR= (or an XVAR=) list but without an ETAVAR= (or a XIVAR=) list, it is assumed that the set of \( y \)-variables (\( x \)-variables) is equivalent to the \( \eta \)-variables (\( \xi \)-variables). Hereafter, this principle is referred to as an equivalence interpretation.

Apparently, the third principle is the same as the situation where the latent factors \( \eta \) (or \( \xi \)) are perfectly measured by the manifest variables \( y \) (or \( x \)). That is, in such a perfect measurement model, \( \Lambda_y \) (\( \Lambda_x \)) is an identity matrix and \( \Theta_y \) (\( \Theta_x \)) and \( \nu_y \) (\( \nu_x \)) are both null. This can be referred to as a perfect measurement interpretation. However, the equivalence interpretation stated in the last principle presumes that there are actually no measurement equations at all. This is important because under the equivalence interpretation, matrices \( \Lambda_y \) (\( \Lambda_x \)), \( \Theta_y \) (\( \Theta_x \)) and \( \nu_y \) (\( \nu_x \)) are nonexistent rather than fixed quantities, which is assumed under the perfect measurement interpretation. Hence, the \( x \)-variables are treated as exogenous variables with the equivalence interpretation, but they are still treated as endogenous with the perfect measurement interpretation. Ultimately, whether \( x \)-variables are treated as exogenous or endogenous affects the default or automatic parameterization. See the section “Default Parameters” on page 1284 for more details.

By using these three principles, the models and submodels that PROC CALIS analyzes are summarized in the following table, followed by detailed descriptions of these models and submodels.
<table>
<thead>
<tr>
<th>Presence of Lists</th>
<th>Description</th>
<th>Model Equations</th>
<th>Nonfixed Model Matrices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presence of Both x- and y-variables</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 YVAR=, ETAVAR=, XVAR=, XIVAR=</td>
<td>Full model</td>
<td>[ y = v_y + \Lambda_y \eta + \epsilon ] [ x = v_x + \Lambda_x \xi + \delta ] [ \eta = \alpha + \beta \eta + \Gamma \xi + \xi ]</td>
<td>[ \nu_y, \Lambda_y, \Theta_y ] [ \nu_x, \Lambda_x, \Theta_x, \kappa, \Phi ] [ \alpha, \beta, \Gamma, \Psi ]</td>
</tr>
<tr>
<td>2 YVAR=, XVAR=, XIVAR=</td>
<td>Full model with ( y \equiv \eta )</td>
<td>[ x = v_x + \Lambda_x \xi + \delta ] [ y = \alpha + \beta y + \Gamma x + \xi ]</td>
<td>[ \nu_x, \Lambda_x, \Theta_x, \kappa, \Phi ] [ \alpha, \beta, \Gamma, \Psi ]</td>
</tr>
<tr>
<td>3 YVAR=, ETAVAR=, XVAR=</td>
<td>Full model with ( y \equiv \eta )</td>
<td>[ y = v_y + \Lambda_y \eta + \epsilon ] [ x = \xi ] [ \eta = \alpha + \beta \eta + \Gamma x + \xi ]</td>
<td>[ \nu_y, \Lambda_y, \Theta_y ] [ \alpha, \beta, \Gamma, \Psi, \kappa, \Phi ]</td>
</tr>
<tr>
<td>4 YVAR=, XVAR=</td>
<td>Regression ( (y \equiv \eta) ) ( (x \equiv \xi) ) [ (I - \beta)^{-1} y = \alpha + \Gamma x + \xi ]</td>
<td></td>
<td>[ \alpha, \beta, \Gamma, \Psi, \kappa, \Phi ]</td>
</tr>
<tr>
<td>Presence of x-variables and Absence of y-variables</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 XVAR=, XIVAR=</td>
<td>Factor model ( x \equiv \xi ) ( \text{for } x )</td>
<td>[ x = v_x + \Lambda_x \xi + \delta ]</td>
<td>[ \nu_x, \Lambda_x, \Theta_x, \kappa, \Phi ]</td>
</tr>
<tr>
<td>6 XVAR=</td>
<td>x-structures ( (x \equiv \xi) )</td>
<td>( \kappa, \Phi )</td>
<td></td>
</tr>
<tr>
<td>Presence of y-variables and Absence of x-variables</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 YVAR=, ETAVAR=</td>
<td>Factor model ( y \equiv \eta ) ( \text{for } y )</td>
<td>[ y = v_y + \Lambda_y \eta + \epsilon ] [ \eta = \alpha + \beta \eta + \xi ]</td>
<td>[ \nu_y, \Lambda_y, \Theta_y ] [ \alpha, \beta, \Psi ]</td>
</tr>
<tr>
<td>8 YVAR=</td>
<td>y-structures ( (y \equiv \eta) ) ( (I - \beta)^{-1} y = \alpha + \xi )</td>
<td>[ y = \alpha + \beta y + \xi ] ( \text{or } \alpha, \beta, \Psi )</td>
<td></td>
</tr>
<tr>
<td>9 YVAR=, ETAVAR=, XIVAR=</td>
<td>Second-order factor model ( (y \equiv \eta) )</td>
<td>[ y = v_y + \Lambda_y \eta + \epsilon ] [ \eta = \alpha + \beta \eta + \Gamma \xi + \xi ]</td>
<td>[ \nu_y, \Lambda_y, \Theta_y ] [ \alpha, \beta, \Gamma, \Psi, \kappa, \Phi ]</td>
</tr>
<tr>
<td>10 YVAR=, XIVAR=</td>
<td>Factor model ( (y \equiv \eta) ) ( (I - \beta)^{-1} y = \alpha + \Gamma \xi + \xi )</td>
<td>[ y = \alpha + \beta y + \Gamma \xi + \xi ] ( \text{or } \alpha, \beta, \Gamma, \Psi, \kappa, \Phi )</td>
<td></td>
</tr>
</tbody>
</table>

**Models 1, 2, 3, and 4—Presence of Both x- and y-Variables**

Submodels 1, 2, 3, and 4 are characterized by the presence of both x- and y-variables in the model. In fact, Model 1 is the full model with the presence of all four types of variables. All twelve model matrices are parameter matrices in this model.

Depending on the absence of the latent factor lists, manifest variables can replace the role of the latent factors in Models 2–4. For example, the absence of the ETAVAR= list in Model 2 means \( y \) is equivalent to \( \eta \) \( (y \equiv \eta) \). Consequently, you cannot, nor do you need to, use the MATRIX statement to specify parameters in the _LAMBDAY_, _THETAY_, or _NUY_ matrices under this model. Similarly, because \( x \) is equivalent to \( \xi \) \( (x \equiv \xi) \) in Model 3, you cannot, nor do you need to, use the MATRIX statement to specify the parameters in the _LAMBDAX_, _THETAX_, or _NUX_ matrices. In Model 4, \( y \) is equivalent to \( \eta \) \( (y \equiv \eta) \) and \( x \) is equivalent to \( \xi \) \( (x \equiv \xi) \). None of the six model matrices in the measurement equations are defined in the
model. Matrices in which you can specify parameters by using the MATRIX statement are listed in the last column of the table.

Describing Model 4 as a regression model is a simplification. Because \( y \) can regress on itself in the model equation, the regression description is not totally accurate for Model 4. Nonetheless, if \( \beta \) is a null matrix, the equation describes a multivariate regression model with outcome variables \( y \) and predictor variables \( x \). This model is the TYPE 2A model in LISREL VI (Jöreskog and Sörbom 1985).

You should also be aware of the changes in meaning of the model matrices when there is an equivalence between latent factors and manifest variables. For example, in Model 4 the \( \Phi \) and \( \kappa \) are now the covariance matrix and mean vector, respectively, of manifest variables \( x \), while in Model 1 (the complete model) these matrices are of the latent factors \( \xi \).

**Models 5 and 6 — Presence of x-Variables and Absence of y-Variables**

Models 5 and 6 are characterized by the presence of the x-variables and the absence of y-variables.

Model 5 is simply a factor model for measured variables \( x \), with \( \Lambda_x \) representing the factor loading matrix, \( \Theta_x \) the error covariance matrix, and \( \Phi \) the factor covariance matrix. If mean structures are modeled, \( \kappa \) represents the factor means and \( \nu_x \) is the intercept vector. This is the TYPE 1 submodel in LISREL VI (Jöreskog and Sörbom 1985).

Model 6 is a special case where there is no model equation. You specify the mean and covariance structures (in \( \kappa \) and \( \Phi \), respectively) for the manifest variables \( x \) directly. The x-variables are treated as exogenous variables in this case. Because this submodel uses direct mean and covariance structures for measured variables, it can also be handled more easily by the MSTRUCT modeling language. See the MSTRUCT statement and the section “The MSTRUCT Model” on page 1416 for more details.

Note that because \( \eta \)-variables cannot exist in the absence of y-variables (see one of the three aforementioned principles for deriving submodels), adding the ETAVAR= list alone to these two submodels does not generate new submodels that can be analyzed by PROC CALIS.

**Models 7, 8, 9 and 10—Presence of y-Variables and Absence of x-Variables**

Models 7–10 are characterized by the presence of the y-variables and the absence of x-variables.

Model 7 is a factor model for y-variables (TYPE 3B submodel in LISREL VI). It is similar to Model 5, but with regressions among latent factors allowed. When \( \beta \) is null, Model 7 functions the same as Model 5. It becomes a factor model for y-variables, with \( \Lambda_y \) representing the factor loading matrix, \( \Theta_y \) the error covariance matrix, \( \Psi \) the factor covariance matrix, \( \alpha \) the factor means, and \( \nu_y \) the intercept vector.

Model 8 (TYPE 2B submodel in LISREL VI) is a model for studying the mean and covariance structures of y-variables, with regression among y-variables allowed. When \( \beta \) is null, the mean structures of y are specified in \( \alpha \) and the covariance structures are specified in \( \Psi \). This is similar to Model 6. However, there is an important distinction. In Model 6, the x-variables are treated as exogenous (no model equation at all). But the y-variables are treated as endogenous in Model 8 (with or without \( \beta = 0 \)). Consequently, the default parameterization would be different for these two submodels. See the section “Default Parameters” on page 1284 for details about the default parameterization.

Model 9 represents a modified version of the second-order factor model for y. It would be a standard second-order factor model when \( \beta \) is null. This is the TYPE 3A submodel in LISREL VI. With \( \beta \) being null, \( \eta \) represents the first-order factors and \( \xi \) represents the second-order factors. The first- and second-order factor loading matrices are \( \Lambda_y \) and \( \Gamma \), respectively.
Chapter 29: The CALIS Procedure

Model 10 is another form of factor model when $\beta$ is null, with factors represented by $\xi$ and manifest variables represented by $y$. However, if $\beta$ is indeed a null matrix in applications, you might want to use Model 5, in which the factor model specification is more direct and intuitive.

**Default Parameters in the LISMOD Model**

When a model matrix is defined in a LISMOD model, you can specify fixed values or free parameters for the elements of the matrix by the `MATRIX` statement. All other unspecified elements in the matrix are set by default. There are two types of default parameters for the LISMOD model matrices: one is free parameters; the other is fixed zeros.

The following sets of parameters are free parameters by default:

- the diagonal elements of the `_THETAX_`, `_THETAY_`, and `_PSI_` matrices; these elements represent the error variances in the model
- all elements of the `_PHI_` matrix; these elements represent the variances and covariance among exogenous variables in the model
- all elements in the `_NUX_` and `_NUY_` vectors if the mean structures are modeled; these elements represent the intercepts of the observed variables
- all elements in the `_ALPHA_` vector if a `YVAR=` list is specified but an `ETAVAR=` list is not specified and the mean structures are modeled; these elements represent the intercepts of the $y$-variables
- all elements in the `_KAPPA_` vector if an `XVAR=` list is specified but an `XIVAR=` list is not specified and the mean structures are modeled; these elements represent the means of the $x$-variables

PROC CALIS names the default free parameters with the `_Add` prefix and a unique integer suffix. You can override the default free parameters by explicitly specifying them as free, constrained, or fixed parameter in the `MATRIX` statements for the matrices.

Parameters that are not default free parameters in the LISMOD model are fixed zeros by default. You can override almost all of these default fixed zeros of the LISMOD model by using the `MATRIX` statements for the matrices. The only set of default fixed zeros that you cannot override is the set of the diagonal elements of the `_BETA_` matrix. These fixed zeros reflect a model restriction that precludes variables from having direct effects on themselves.

**The MSTRUCT Model**

In contrast to other modeling languages where the mean and covariance structures are implied from the specification of equations, paths, variable-factor relations, mean parameters, variance parameters, or covariance parameters, the MSTRUCT modeling language is supported in PROC CALIS for modeling mean and covariance structures directly.

A simple example for using the MSTRUCT modeling language is the testing of a covariance model with equal variances and covariances. Suppose that a variable was measured five times in an experiment. The covariance matrix of these five measurements is hypothesized to have the structure

$$ \Sigma = \Sigma(\theta) $$
where

$$\theta = (\phi, \tau)$$

and

$$\Sigma(\theta) = \begin{pmatrix} \phi & \tau & \tau & \tau & \tau \\ \tau & \phi & \tau & \tau & \tau \\ \tau & \tau & \phi & \tau & \tau \\ \tau & \tau & \tau & \phi & \tau \\ \tau & \tau & \tau & \tau & \phi \end{pmatrix}$$

For model structures that are hypothesized directly in the covariance matrix, the MSTRUCT modeling language is the most convenient to use. You can also use other general modeling languages such as LINEQS, PATH, or RAM to fit the same model structures, but the specification is less straightforward and more error-prone. For convenience, models that are specified using the MSTRUCT modeling language are called MSTRUCT models. For applications of MSTRUCT modeling, see Yung, Browne, and Zhang (2014).

Model Matrices in the MSTRUCT Model

Suppose that there are $p$ observed variables. The two model matrices, their names, their roles, and their dimensions are summarized in the following table.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Name</th>
<th>Description</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma$</td>
<td><em>COV</em> or <em>MSTRUCTCOV</em></td>
<td>Structured covariance matrix</td>
<td>$p \times p$</td>
</tr>
<tr>
<td>$\mu$</td>
<td><em>MEAN</em> or <em>MSTRUCTMEAN</em></td>
<td>Structured mean vector</td>
<td>$p \times 1$</td>
</tr>
</tbody>
</table>

Specification of the MSTRUCT Model

**Specifying Variables**

In the MSTRUCT statement, you specify the list of $p$ manifest variables of interest in the VAR= list. For example, you specify $v1$–$v5$ as the variables to be analyzed in your MSTRUCT model by this statement:

```plaintext
mstruct VAR= v1 v2 v3 v4 v5;
```

See the MSTRUCT statement on page 1313 for details about the syntax.

The manifest variables in the VAR= list must be referenced in the input set. The number of variables in the VAR= list determines the dimensions of the _COV_ and the _MEAN_ matrices in the model. In addition, the order of variables determines the order of row and column variables in the model matrices.

**Specifying Parameters in Model Matrices**

Denote the parameter vector in the MSTRUCT model as $\theta$. The dimension of $\theta$ depends on your hypothesized model. In the preceding example, $\theta$ contains two parameters in $\phi$ and $\tau$. You can use the MATRIX statement to specify these parameters in the _COV_ matrix:

```plaintext
matrix _COV_ [1,1] = 5*phi, /* phi for all diagonal elements */
              [2, ] = tau,  /* tau for all off-diagonal elements */
              [3, ] = 2*tau,
              [4, ] = 3*tau,
              [5, ] = 4*tau;
```
In this MATRIX statement, the five diagonal elements, starting from the \([1,1]\) element of the covariance matrix, are fitted by the \(\phi\) parameter. The specification \(5*\phi\) is a shorthand for specifying \(\phi\) five times, once for each of the five diagonal elements in the covariance matrix. All other lower triangular elements are fitted by the \(\tau\) parameter, as shown in the MATRIX statement. For example, with \([3,1]\) the elements starting from the first element of the third row of the \(_{COV}\_\) matrix are parameterized by the \(\tau\) parameter. The specification \(2*\tau\) repeats the specification two times, meaning that the \([3,1]\) and \([3,2]\) elements are both fitted by the same parameter \(\tau\). Similarly, all lower triangular elements (not including the diagonal elements) of the \(_{COV}\_\) matrix are fitted by the \(\tau\) parameter. The specification of the upper triangular elements (diagonal excluded) of the \(_{COV}\_\) matrix is not needed because the \(_{COV}\_\) matrix is symmetric. The specification in the lower triangular elements is transferred automatically to the upper triangular elements. See the MATRIX statement on page 1295 for details about the syntax.

**Default Parameters in the MSTRUCT Model**

By using the MATRIX statements, you can specify either fixed values or free parameters (with or without initial values) for the elements in the \(_{COV}\_\) and \(_{MEAN}\_\) model matrices. If some or all elements are not specified, default parameters are applied to the MSTRUCT. There are two types of default parameters: one is free parameters; the other is fixed zeros. They are applied in different situations.

If you do not specify *any* elements of the \(_{COV}\_\) matrix with the MATRIX statement, all elements of the \(_{COV}\_\) matrix are free parameters by default. PROC CALIS names the default free parameters with the \_Add prefix and a unique integer suffix. However, if you specify *at least one* fixed or free parameter of the \(_{COV}\_\) matrix with the MATRIX statement, then all other unspecified elements of the \(_{COV}\_\) matrix are fixed zeros by default.

If the mean structures are modeled, the same treatment applies to the \(_{MEAN}\_\) vector. That is, if you do not specify any elements of the \(_{MEAN}\_\) vector with the MATRIX statement, all elements of the \(_{MEAN}\_\) vector are free parameters by default. However, if you specify *at least one* fixed or free parameter of the \(_{MEAN}\_\) vector with the MATRIX statement, then all other unspecified elements of the \(_{MEAN}\_\) vector are fixed zeros by default.

**How and Why the Default Parameters Are Treated Differently in the MSTRUCT Model**

Notice that the default parameter treatment of the MSTRUCT model is quite different from other types of models such as FACTOR, LINEQS, LISMOD, RAM, or PATH. For these models, unspecified variances and covariances among exogenous variables are all free parameters by default. However, for the MSTRUCT model, either default free parameters or fixed zeros are generated depending on whether at least one element of the covariance matrix is specified. The reason for this different treatment is that you fit the covariance structure directly by using the MSTRUCT modeling language. Hence, in an MSTRUCT model there is no information regarding the functional relationships among the variables that indicates whether the variables are exogenous or endogenous in the model. Hence, PROC CALIS cannot assume default free parameters based on the exogenous or endogenous variable types.

Because of this special default parameter treatment, when fitting an MSTRUCT model you must make sure that each diagonal element in your \(_{COV}\_\) matrix is set as a free, constrained, or fixed parameter, in accordance with your theoretical model. If you specify some elements in the model matrix but omit the specification of other diagonal elements, the default fixed zero variances would lead to a nonpositive definite \(_{COV}\_\) model matrix, making the model fitting problematic.
The PATH Model

The PATH modeling language is supported in PROC CALIS as a more intuitive modeling tool. It is designed so that specification by using the PATH modeling language translates effortlessly from the path diagram. For example, consider the following simple path diagram:

```
A
|-->
|   
C
|-->

B
```

You can use the following PATH statement to specify the paths easily:

```
path A ===> B ,
     C ===> B ;
```

There are two path entries in the PATH statement: one is for the path $A \rightarrow B$, and the other is for the path $C \rightarrow B$. Sometimes you might want to name the effect parameters in the path diagram, as shown in the following:

```
A
|-->
|   
C
|-->

B
```

You can specify the paths and the parameters together in the following statement:

```
path A ===> B = effect1,
     C ===> B = effect2;
```

In the first entry of the PATH statement, the path $A \rightarrow B$ is specified together with the path coefficient (effect) $effect1$. Similarly, in the second entry, the $C \rightarrow B$ path is specified together with the effect parameter $effect2$. In addition to the path coefficients (effects) in the path diagram, you can also specify other types of parameters by using the PVAR and PCOV statements. See the section “A Structural Equation Example” on page 1184 for a more detailed example of the PATH model specification.

Despite its simple representation of the path diagram, the PATH modeling language is general enough to handle a wide class of structural models that can also be handled by other general modeling languages such as LINEQS, LISMOD, or RAM. For brevity, models specified by the PATH modeling language are called PATH models.

Types of Variables in the PATH Model

When you specify the paths in the PATH model, you typically use arrows (such as $\leftarrow$ or $\rightarrow$) to denote “causal” paths. For example, in the preceding path diagram or the PATH statement, you specify that B is an outcome variable with predictors A and C, respectively, in two paths. An outcome variable is the variable being pointed to in a path specification, while the predictor variable is the one where the arrow starts from.

Whereas the outcome–predictor relationship describes the roles of variables in each single path, the endogenous–exogenous relationship describes the roles of variables in the entire system of paths. In a
system of path specification, a variable is endogenous if it is pointed to by at least one single-headed arrow or it serves as an outcome variable in at least one path. Otherwise, it is exogenous. In the preceding path diagram, for example, variable B is endogenous and both variables A and C are exogenous. Note that although any variable that serves as an outcome variable at least in one path must be endogenous, it does not mean that all endogenous variables must serve only as outcome variables in all paths. An endogenous variable in a model might also serve as a predictor variable in a path. For example, variable B in the following PATH statement is an endogenous variable, and it serves as an outcome variable in the first path but as a predictor variable in the second path.

```
path  A ===> B  = effect1,
     B ===> C  = effect2;
```

A variable is a manifest or observed variable in the PATH model if it is measured and exists in the input data set. Otherwise, it is a latent variable. Because error variables are not explicitly defined in the PATH modeling language, all latent variables that are named in the PATH model are factors, which are considered to be the systematic source of effects in the model. Each manifest variable in the PATH model can be endogenous or exogenous. The same is true for any latent factor in the PATH model.

Because you do not name error variables in the PATH model, you do not need to specify paths from errors to any endogenous variables. Error terms are implicitly assumed for all endogenous variables in the PATH model. Although error variables are not named in the PATH model, the error variances are expressed equivalently as partial variances of the associated endogenous variables. These partial variances are set by default in the PATH modeling language. Therefore, you do not need to specify error variance parameters explicitly unless constraints on these parameters are desirable in the model. You can use the PVAR statement to specify the error variance or partial variance parameters explicitly.

### Naming Variables in the PATH Model

Manifest variables in the PATH model are referenced in the input data set. Their names must not be longer than 32 characters. There are no further restrictions beyond those required by the SAS System. You use the names of manifest variables directly in the PATH model specification.

Because you do not name error variables in the PATH model, all latent variables named in the PATH model specification are factors (non-errors). Factor names in the PATH model must not be longer than 32 characters, and they should be different from the manifest variables. Unlike the LINEQS model, you do not need to use ‘F’ or ‘f’ prefix to denote latent factors in the PATH model. As a general naming convention, you should not use **Intercept** as either a manifest or latent variable name. See the section “Naming Variables and Parameters” on page 1433 for these general rules about naming variables and parameters.

### Specification of the PATH Model

**(1) Specification of Effects or Paths**

You specify the “causal” paths or linear functional relationships among variables in the PATH statement. For example, if there is a path from v2 to v1 in your model and the effect parameter is named parm1 with a starting value at 0.5, you can use either of these specifications:

```
path  v1 <= v2  = parm1(0.5);

path  v2 => v1  = parm1(0.5);
```

If you have more than one path in your model, path specifications should be separated by commas, as shown in the following PATH statement:
The PATH Model

```
path
v1 <= v2 = parm1(0.5),
v2 <= v3 = parm2(0.3);
```

Because the PATH statement can be used only once in each model specification, all paths in the model must be specified together in a single PATH statement. See the PATH statement on page 1322 for more details about the syntax.

(2) Specification of Variances and Partial (Error) Variances

If $$v_2$$ is an exogenous variable in the PATH model and you want to specify its variance as a parameter named parm2 with a starting value at 10, you can use the following PVAR statement specification:

```
pvar v2 = parm2(10.);
```

If $$v_1$$ is an endogenous variable in the PATH model and you want to specify its partial variance or error variance as a parameter named parm3 with a starting value at 5.0, you can also use the following PVAR statement specification:

```
pvar v1 = parm3(5.0);
```

Therefore, the PVAR statement can be used for both exogenous and endogenous variables. When a variable in the statement is exogenous (which can be automatically determined by PROC CALIS), you are specifying the variance parameter of the variable. Otherwise, you are specifying the partial or error variance for an endogenous variable.

You do not need to supply the parameter names for the variances or partial variances if these parameters are not constrained. For example, the following statement specifies the unnamed free parameters for variances or partial variances of $$v_1$$ and $$v_2$$:

```
pvar v1 v2;
```

If you have more than one variance or partial variance parameter to specify in your model, you can put a variable list on the left-hand side of the equal sign, and a parameter list on the right-hand side, as shown in the following PVAR statement specification:

```
pvar v1 v2 v3 = parm1(0.5) parm2 parm3;
```

In the specification, variance or partial variance parameters for variables $$v_1$$–$$v_3$$ are $$\text{parm1}$$, $$\text{parm2}$$, and $$\text{parm3}$$, respectively. Only $$\text{parm1}$$ is given an initial value at 0.5. The initial values for other parameters are generated by PROC CALIS.

You can also separate the specifications into several entries in the PVAR statement. Entries should be separated by commas. For example, the preceding specification is equivalent to the following specification:

```
pvar v1 = parm1 (0.5),
v2 = parm2,
v3 = parm3;
```

Because the PVAR statement can be used only once in each model specification, all variance and partial variance parameters in the model must be specified together in a single PVAR statement. See the PVAR statement on page 1345 for more details about the syntax.
(3) Specification of Covariances and Partial Covariances

If you want to specify the (partial) covariance between two variables v3 and v4 as a parameter named parm4 with a starting value at 3, you can use the following PCOV statement specification:

```
pcov v3 v4 = parm4 (5.);
```

Whether parm4 is a covariance or partial covariance parameter depends on the variable types of v3 and v4. If both v3 and v4 are exogenous variables (manifest or latent), parm4 is a covariance parameter between v3 and v4. If both v3 and v4 are endogenous variables (manifest or latent), parm4 is a parameter for the covariance between the errors for v3 and v4. In other words, it is a partial covariance or error covariance parameter for v3 and v4.

A less common case is when one of the variables is exogenous and the other is endogenous. In this case, parm4 is a parameter for the partial covariance between the endogenous variable and the exogenous variable, or the covariance between the error for the endogenous variable and the exogenous variable. Fortunately, such covariances are relatively uncommon in statistical modeling. Their uses confuse the roles of systematic and unsystematic sources in the model and lead to difficulties in interpretations. Therefore, you should almost always avoid this kind of partial covariance.

Like the syntax of the PVAR statement, you can specify a list of (partial) covariance parameters in the PCOV statement. For example, consider the following statement:

```
pcov v1 v2 = parm4, v1 v3 = parm5, v2 v3 = parm6;
```

In the specification, three (partial) covariance parameters parm4, parm5, and parm6 are specified, respectively, for the variable pairs (v1,v2), (v1,v3), and (v2,v3). Entries for (partial) covariance specification are separated by commas.

Again, if all these covariances are not constrained, you can omit the names for the parameters. For example, the preceding specification can be specified as the following statement when the three covariances are free parameters in the model:

```
pcov v1 v2, v1 v3, v2 v3;
```

Or, you can simply use the following within-list covariance specification:

```
pcov v1 v2 v3;
```

Three covariance parameters are generated by this specification.

Because the PCOV statement can be used only once in each model specification, all covariance and partial covariance parameters in the model must be specified together in a single PCOV statement. See the PCOV statement on page 1343 for more details about the syntax.

(4) Specification of Means and Intercepts

Means and intercepts are specified when the mean structures of the model are of interest. You can specify mean and intercept parameters in the MEAN statement. For example, consider the following statement:
mean V5 = parm5(11.);  

If V5 is an exogenous variable (which is determined by PROC CALIS automatically), you are specifying parm5 as the mean parameter of V5. If V5 is an endogenous variable, you are specifying parm5 as the intercept parameter for V5.

Because each named variable in the PATH model is either exogenous or endogenous (exclusively), each variable in the PATH model would have either a mean or an intercept parameter (but not both) to specify in the MEAN statement. Like the syntax of the PVAR statement, you can specify a list of mean or intercept parameters in the MEAN statement. For example, in the following statement you specify a list of mean or intercept parameters for variables v1–v4:

```
mean v1-v4 = parm6-parm9;
```

This specification is equivalent to the following specification with four entries of parameter specifications:

```
mean
  v1 = parm6,
  v2 = parm7,
  v3 = parm8,
  v4 = parm9;
```

Again, entries in the MEAN statement must be separated by commas, as shown in the preceding statement.

Because the MEAN statement can be used only once in each model specification, all mean and intercept parameters in the model must be specified together in a single MEAN statement. See the MEAN statement on page 1309 for more details about the syntax.

**Specifying Parameters without Initial Values**

If you do not have any knowledge about the initial value for a parameter, you can omit the initial value specification and let PROC CALIS compute it. For example, you can provide just the parameter locations and parameter names as in the following specification:

```
path v1 <=== v2 = parm1;
  pvar v2 = parm2,
   v1 = parm3;
```

**Specifying Fixed Parameter Values**

If you want to specify a fixed parameter value, you do not need to provide a parameter name. Instead, you provide the fixed value (without parentheses) in the specification.

For example, in the following statement the path coefficient for the path is fixed at 1.0 and the (partial) variance of F1 is also fixed at 1.0:

```
path v1 <=== F1 = 1.0;
  pvar
    F1 = 1.0;
```

**A Complete PATH Model Specification Example**

The following specification shows a more complete PATH model specification:
The two paths specified in the PATH statement have unnamed free effect parameters. These parameters are named by PROC CALIS with the _Parm prefix and unique integer suffixes. The error variance of v1 is an unnamed parameter, while the variances of v2 and v3 are constrained by using the same parameter parm3. The covariance between v2 and v3 is a free parameter named parm5, with a starting value of 5.0.

**Default Parameters in the PATH Model**

There are two types of default parameters of the PATH model. One is the free parameters; the other is the fixed constants.

The following sets of parameters are free parameters by default:

- the variances or partial (or error) variances of all variables, manifest or latent
- the covariances among all exogenous (independent) manifest or latent variables
- the means of all exogenous (independent) manifest variables if the mean structures are modeled
- the intercepts of all endogenous (dependent) manifest variables if the mean structures are modeled

For each of the default free parameters, PROC CALIS generates a parameter name with the _Add prefix and a unique integer suffix. Parameters that are not default free parameters in the PATH model are fixed zeros by default. You can override almost all of the default zeros of the PATH model by using the MEAN, PATH, PCOV, and MEAN statements. The only exception is the single-headed path that has the same variable on both sides. That is, the following specification is not accepted by PROC CALIS:

```
path v1 <= v1 = parm;
```

This path should always have a zero coefficient, which is treated as a model restriction that prevents a variable from having a direct effect on itself.

**Relating the PATH Model to the RAM Model**

Mathematically, the PATH model is essentially the RAM model. You can consider the PATH model to share exactly the same set of model matrices as in the RAM model. See the section “Model Matrices in the RAM Model” on page 1425 and the section “Summary of Matrices and Submatrices in the RAM Model” on page 1428 for details about the RAM model matrices. In the RAM model, the A matrix contains effects or path coefficients for describing relationships among variables. In the PATH model, you specify these effect or coefficient parameters in the PATH statement. The P matrix in the RAM model contains (partial) variance and (partial) covariance parameters. In the PATH model, you use the PVAR and PCOV statements to specify these parameters. The W vector in the RAM model contains the mean and intercept parameters, while in the PATH model you use the MEAN statement to specify these parameters. By using these model matrices in the PATH model, the covariance and mean structures are derived in the same way as they are derived in the RAM model. See the section “The RAM Model” on page 1425 for derivations of the model structures.
The RAM Model

The RAM modeling language is adapted from the basic RAM model developed by McArdle (1980). For brevity, models specified by the RAM modeling language are called RAM models. You can also specify these so-called RAM models by other general modeling languages that are supported in PROC CALIS.

Types of Variables in the RAM Model

A variable in the RAM model is manifest if it is observed and is defined in the input data set. A variable in the RAM model is latent if it is not manifest. Because error variables are not explicitly named in the RAM model, all latent variables in the RAM model are considered as factors (non-error-type latent variables).

A variable in the RAM model is endogenous if it ever serves as an outcome variable in the RAM model. That is, an endogenous variable has at least one path (or an effect) from another variable in the model. A variable is exogenous if it is not endogenous. Endogenous variables are also referred to as dependent variables, while exogenous variables are also referred to as independent variables.

In the RAM model, distinctions between exogenous and endogenous and between latent and manifest for variables are not essential to the definitions of model matrices, although they are useful for conceptual understanding when the model matrices are partitioned.

Naming Variables in the RAM Model

Manifest variables in the RAM model are referenced in the input data set. Their names must not be longer than 32 characters. There are no further restrictions beyond those required by the SAS System.

Latent variables in the RAM model are those not being referenced in the input data set. Their names must not be longer than 32 characters. Unlike the LINEQS model, you do not need to use any specific prefix (for example, ‘F’ or ‘f’) for the latent factor names. The reason is that error or disturbance variables in the RAM model are not named explicitly in the RAM model. Thus, any variable names that are not referenced in the input data set are for latent factors.

As a general naming convention, you should not use Intercept as either a manifest or latent variable name.

Model Matrices in the RAM Model

In terms of the number of model matrices involved, the RAM model is the simplest among all the general structural equations models that are supported by PROC CALIS. Essentially, there are only three model matrices in the RAM model: one for the interrelationships among variables, one for the variances and covariances, and one for the means and intercepts. These matrices are discussed in the following subsections.

Matrix $A$ ($n_a \times n_a$): Effects of Column Variables on Row Variables

The row and column variables of matrix $A$ are the set of manifest and latent variables in the RAM model. Unlike the LINEQS model, the set of latent variables in the RAM model matrix does not include the error or disturbance variables. Each entry or element in the $A$ matrix represents an effect of the associated column variable on the associated row variable or a path coefficient from the associated column variable to the associated row variable. A zero entry means an absence of a path or an effect.

The pattern of matrix $A$ determines whether a variable is endogenous or exogenous. A variable in the RAM model is endogenous if its associated row in the $A$ matrix has at least one nonzero entry. Any other variable in the RAM model is exogenous.
Mathematically, you do not need to arrange the set of variables for matrix $A$ in a particular order, as long as the order of variables is the same for the rows and the columns. However, arranging the variables according to whether they are endogenous or exogenous is useful for showing the partitions of the model matrices and certain mathematical properties. See the section “Partitions of the RAM Model Matrices and Some Restrictions” on page 1427 for details.

**Matrix $P$ ($n_a \times n_a$): Variances, Covariances, Partial Variances, and Partial Covariances**

The row and column variables of matrix $P$ refer to the same set of manifest and latent variables that are defined in the RAM model matrix $A$. The diagonal entries of $P$ contain variances or partial variances of variables. If a variable is exogenous, then the corresponding diagonal element in the $P$ matrix represents its variance. Otherwise, the corresponding diagonal element in the $P$ matrix represents its partial variance. This partial variance is an unsystematic source of variance that is not explained by the interrelationships of variables in the model. In most cases, you can interpret a partial variance as the error variance for an endogenous variable.

The off-diagonal elements of $P$ contain covariances or partial covariances among variables. An off-diagonal element in $P$ that is associated with exogenous row and column variables represents covariance between the two exogenous variables. An off-diagonal element in $P$ that is associated with endogenous row and column variables represents partial covariance between the two variables. This partial covariance is unsystematic, in the sense that it is not explained by the interrelationships of variables in the model. In most cases, you can interpret a partial covariance as the error covariance between the two endogenous variables involved. An off-diagonal element in $P$ that is associated with one exogenous variable and one endogenous variable in the row and column represents the covariance between the exogenous variable and the error of the endogenous variable. While this interpretation sounds a little awkward and inelegant, this kind of covariance, fortunately, is rare in most applications.

**Vector $W$ ($n_a \times 1$): Intercepts and Means**

The row variables of vector $W$ refer to the same set of manifest and latent variables that are defined in the RAM model matrix $A$. Elements in $W$ represent either intercepts or means. An element in $W$ that is associated with an exogenous row variable represents the mean of the variable. An element in $W$ that is associated with an endogenous row variable represents the intercept term for the variable. An element in $W$ that is associated with an endogenous row variable represents the intercept term for the variable.

**Covariance and Mean Structures**

Assuming that $(I - A)$ is invertible, where $I$ is an identity matrix of the same dimension as $A$, the structured covariance matrix of all variables (including latent variables) in the RAM model is shown as follows:

$$\Sigma_a = (I - A)^{-1}P(I - A)^{-1'}$$

The structured mean vector of all variables is shown as follows:

$$\mu_a = (I - A)^{-1}W$$

The covariance and mean structures of all manifest variables are obtained by selecting the elements in $\Sigma_a$ and $\mu_a$. This can be achieved by defining a selection matrix $G$ of dimensions $n \times n_a$, where $n$ is the number of manifest variables in the model. The selection matrix $G$ contains zeros and ones as its elements. Each row of $G$ has exactly one nonzero element at the position that corresponds to the location of a manifest row variable in $\Sigma_a$ or $\mu_a$. With each row of $G$ selecting a distinct manifest variable, the structured covariance matrix of all manifest variables is expressed as the following:

$$\Sigma = GG'$$
The structured mean vector of all observed variables is expressed as the following:

\[ \mu = G \mu_a \]

**Partitions of the RAM Model Matrices and Some Restrictions**

There are some model restrictions in the RAM model matrices. Although these restrictions do not affect the derivation of the covariance and mean structures, they are enforced in the RAM model specification.

For convenience, it is useful to assume that \( n_a \) variables are arranged in the order of \( n_d \) endogenous (or dependent) variables and the \( n_i \) exogenous (independent) variables in the rows and columns of the model matrices.

**Model Restrictions on the A Matrix**

The A matrix is partitioned as

\[
A = \begin{pmatrix} \beta & \gamma \\ 0 & 0 \end{pmatrix}
\]

where \( \beta \) is an \( n_d \times n_d \) matrix for paths or effects from (column) endogenous variables to (row) endogenous variables and \( \gamma \) is an \( n_d \times n_i \) matrix for paths (effects) from (column) exogenous variables to (row) endogenous variables.

As shown in the matrix partitions, there are four submatrices. The two submatrices at the lower parts are seemingly structured to zeros. However, this should not be interpreted as restrictions imposed by the model. The zero submatrices are artifacts created by the exogenous-endogenous arrangement of the row and column variables. The only restriction on the A matrix is that the diagonal elements must all be zeros. This implies that the diagonal elements of the submatrix \( \beta \) are also zeros. This restriction prevents a direct path from any endogenous variable to itself. There are no restrictions on the pattern of \( \gamma \).

It is useful to denote the lower partitions of the A matrix by \( A_{LL} \) (lower left) and \( A_{LR} \) (lower right) so that

\[
A = \begin{pmatrix} \beta & \gamma \\ A_{LL} & A_{LR} \end{pmatrix}
\]

Although they are zero matrices in the initial model specification, their entries could become non-zero (paths) in an improved model when you modify your model by using the Lagrange multiplier statistics (see the section “Modification Indices” on page 1501 or the MODIFICATION option). Hence, you might need to reference these two submatrices when you apply the customized LM tests on them during the model modification process (see the LMTESTS statement).

For the purposes of defining specific parameter regions in customized LM tests, you might also partition the A matrix in other ways. First, you can partition A into the left and right portions,

\[
A = \begin{pmatrix} A_{Left} & A_{Right} \end{pmatrix}
\]

where \( A_{Left} \) is top-down concatenation of the \( \beta \) and \( A_{LL} \) matrices and \( A_{Right} \) is the top-down concatenation of the \( \gamma \) and \( A_{LR} \) matrices. Second, you can partition A into the upper and lower portions,

\[
A = \begin{pmatrix} A_{Upper} \\ A_{Lower} \end{pmatrix}
\]
where $A_{Upper}$ is the side-by-side concatenation of the $\beta$ and $\gamma$ matrices and $A_{Lower}$ is the side-by-side concatenation of the $A_{LL}$ and $A_{LR}$ matrices.

In your initial model, because of the arrangement of the endogenous and exogenous variables $A_{Lower}$ is a null matrix. But if you improve your model by applying the LM tests on the entries in $A_{Lower}$, some of these entries might become free paths in your improved model. Hence, some exogenous variables in your initial model now become endogenous variables in your improved model. For this reason, $A_{Lower}$ is also designated as a parameter region for new endogenous variables, which is exactly what the NEWENDO region means in the LMTESTS statement.

**Partition of the $P$ Matrix**

The $P$ matrix is partitioned as

$$P = \begin{pmatrix} P_{11} & P'_{21} \\ P_{21} & P_{22} \end{pmatrix}$$

where $P_{11}$ is an $n_d \times n_d$ partial covariance matrix for the endogenous variables, $P_{22}$ is an $n_i \times n_i$ covariance matrix for the exogenous variables, and $P_{21}$ is an $n_i \times n_d$ covariance matrix between the exogenous variables and the error terms for the endogenous variables. Because $P$ is symmetric, $P_{11}$ and $P_{22}$ are also symmetric.

There are virtually no model restrictions placed on these submatrices. However, in most statistical applications, errors for endogenous variables represent unsystematic sources of effects and therefore they are not to be correlated with other systematic sources such as the exogenous variables in the RAM model. This means that in most practical applications $P_{21}$ would be a null matrix, although this is not enforced in PROC CALIS.

**Partition of the $W$ Vector**

The $W$ vector is partitioned as

$$W = \begin{pmatrix} \alpha \\ \nu \end{pmatrix}$$

where $\alpha$ is an $n_d \times 1$ vector for intercepts of the endogenous variables and $\nu$ is an $n_i \times 1$ vector for the means of the exogenous variables. There is no model restriction on these subvectors.

**Summary of Matrices and Submatrices in the RAM Model**

Let $n_a$ be the total number of manifest and latent variables in the RAM model. Of these $n_a$ variables, $n_d$ are endogenous and $n_i$ are exogenous. Suppose that the rows and columns of the RAM model matrices $A$ and $P$ and the rows of $W$ are arranged in the order of $n_d$ endogenous variables and then $n_i$ exogenous variables. The names, roles, and dimensions of the RAM model matrices and submatrices are summarized in the following table.
### The RAM Model

In PROC CALIS, the RAM model specification is a matrix-oriented modeling language. That is, you have to define the row and column variables for the model matrices and specify the parameters in terms of matrix entries. The VAR= option specifies the variables (including manifest and latent) in the model. For example, the following statement specifies five variables in the model:

```plaintext
RAM
  var= v1 v2 v3;
```

The order of variables in the VAR= option is important. The same order is used for the row and column variables in the model matrices. After you specify the variables in the model, you can specify three types of parameters, which correspond to the elements in the three model matrices. The three types of RAM entries are described in the following.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Name</th>
<th>Description</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Matrices</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td><em>A</em> or <em>RAMA</em></td>
<td>Effects of column variables on row variables, or paths from the column variables to the row variables</td>
<td>$n_a \times n_a$</td>
</tr>
<tr>
<td>P</td>
<td><em>P</em> or <em>RAMP</em></td>
<td>(Partial) variances and covariances</td>
<td>$n_a \times n_a$</td>
</tr>
<tr>
<td>W</td>
<td><em>W</em> or <em>RAMW</em></td>
<td>Intercepts and means</td>
<td>$n_a \times 1$</td>
</tr>
<tr>
<td>Submatrices</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td><em>RAMBETA</em></td>
<td>Effects of endogenous variables on endogenous variables</td>
<td>$n_d \times n_d$</td>
</tr>
<tr>
<td>$\gamma$</td>
<td><em>RAMGAMMA</em></td>
<td>Effects of exogenous variables on endogenous variables</td>
<td>$n_d \times n_i$</td>
</tr>
<tr>
<td>$A_{LL}$</td>
<td><em>RAMA_LL</em></td>
<td>The null submatrix at the lower left portion of <em>A</em></td>
<td>$n_i \times n_d$</td>
</tr>
<tr>
<td>$A_{LR}$</td>
<td><em>RAMA_LR</em></td>
<td>The null submatrix at the lower right portion of <em>A</em></td>
<td>$n_i \times n_i$</td>
</tr>
<tr>
<td>$A_{Left}$</td>
<td><em>RAMA_LEFT</em></td>
<td>The left portion of <em>A</em>, including $\beta$ and $A_{LL}$</td>
<td>$n_a \times n_d$</td>
</tr>
<tr>
<td>$A_{Right}$</td>
<td><em>RAMA_RIGHT</em></td>
<td>The right portion of <em>A</em>, including $\gamma$ and $A_{LR}$</td>
<td>$n_a \times n_i$</td>
</tr>
<tr>
<td>$A_{Upper}$</td>
<td><em>RAMA_UPPER</em></td>
<td>The upper portion of <em>A</em>, including $\beta$ and $\gamma$</td>
<td>$n_d \times n_a$</td>
</tr>
<tr>
<td>$A_{Lower}$</td>
<td><em>RAMA_LOWER</em></td>
<td>The lower portion of <em>A</em>, including $A_{LL}$ and $A_{LR}$</td>
<td>$n_i \times n_a$</td>
</tr>
<tr>
<td>$P_{11}$</td>
<td><em>RAMP11</em></td>
<td>Error variances and covariances for endogenous variables</td>
<td>$n_d \times n_d$</td>
</tr>
<tr>
<td>$P_{21}$</td>
<td><em>RAMP21</em></td>
<td>Covariances between exogenous variables and error terms for endogenous variables</td>
<td>$n_d \times n_i$</td>
</tr>
<tr>
<td>$P_{22}$</td>
<td><em>RAMP22</em></td>
<td>Variances and covariances for exogenous variables</td>
<td>$n_i \times n_i$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td><em>RAMALPHA</em></td>
<td>Intercepts for endogenous variables</td>
<td>$n_d \times 1$</td>
</tr>
<tr>
<td>$\nu$</td>
<td><em>RAMNU</em></td>
<td>Means for exogenous variables</td>
<td>$n_i \times 1$</td>
</tr>
</tbody>
</table>
Chapter 29: The CALIS Procedure

(1) Specification of Effects or Paths in Model Matrix A

If there is a path from V2 to V1 in your model and the associated effect parameter is named parm1 with 0.5 as the starting value, you can use the following RAM statement:

```
RAM
  var= v1 v2 v3,
  _A_ 1 2 parm1(0.5);
```

The `ram-entry` that starts with `_A_` means that an element of the ram matrix A is being specified. The row number and the column number of this element are 1 and 2, respectively. With reference to the VAR= list, the row number 1 refers to variable v1, and the column number 2 refers to variable v2. Therefore, the effect of V2 on V1 is a parameter named parm1, with an initial value of 0.5.

You can specify fixed values in the `ram-entries` too. Suppose the effect of v3 on v1 is fixed at 1.0. You can use the following specification:

```
RAM
  var= v1 v2 v3,
  _A_ 1 2 parm1(0.5),
  _A_ 1 3 1.0;
```

(2) Specification of the Latent Factors in the Model

In the RAM model, you specify the list of variables in VAR= list of the RAM statement. The list of variables can include the latent variables in the model. Because observed variables have references in the input data sets, those variables that do not have references in the data sets are treated as latent factors automatically. Unlike the LINEQS model, you do not need to use ‘F’ or ‘f’ prefix to denote latent factors in the RAM model. It is recommended that you use meaningful names for the latent factors. See the section “Naming Variables and Parameters” on page 1433 for the general rules about naming variables and parameters.

For example, suppose that SES_Factor and Education_Factor are names that are not used as variable names in the input data set. These two names represent two latent factors in the model, as shown in the following specification:

```
RAM
  var= v1 v2 v3 SES_FACTOR Education_Factor,
  _A_ 1 4 b1,
  _A_ 2 5 b2,
  _A_ 3 5 1.0;
```

This specification shows that the effect of SES_Factor on v1 is a free parameter named b1, and the effects of Education_Factor on v2 and v3 are a free parameter named b2 and a fixed value of 1.0, respectively.

However, naming latent factors is not compulsory. The preceding specification is equivalent to the following specification:

```
RAM
  var= v1 v2 v3,
  _A_ 1 4 b1,
  _A_ 2 5 b2,
  _A_ 3 5 1.0;
```

Although you do not name the fourth and the fifth variables in the VAR= list, PROC CALIS generates the names for these two latent variables. In this case, the fourth variable is named _Factor1 and the fifth variable is named _Factor2.
(3) Specification of (Partial) Variances and (Partial) Covariances in Model Matrix \( \mathbf{P} \)

Suppose now you want to specify the variance of \( v_2 \) as a free parameter named \( \text{parm2} \). You can add a new *ram-entry* for this variance parameter, as shown in the following statement:

\[
\text{RAM} \quad \text{var=} \ v_1 \ v_2 \ v_3, \\
\_A\_ \ 1 \ 2 \ \text{parm1}(0.5), \\
\_A\_ \ 1 \ 3 \ 1.0, \\
\_P\_ \ 2 \ 2 \ \text{parm2};
\]

The *ram-entry* that starts with \( \_P\_ \) means that an element of the RAM matrix \( \mathbf{P} \) is being specified. The (2,2) element of \( \mathbf{P} \), which is the variance of \( v_2 \), is a parameter named \( \text{parm2} \). You do not specify an initial value for this parameter.

You can also specify the error variance of \( v_1 \) similarly, as shown in the following statement:

\[
\text{RAM} \quad \text{var=} \ v_1 \ v_2 \ v_3, \\
\_A\_ \ 1 \ 2 \ \text{parm1}(0.5), \\
\_A\_ \ 1 \ 3 \ 1.0, \\
\_P\_ \ 2 \ 2 \ \text{parm2}, \\
\_P\_ \ 1 \ 1;
\]

In the last *ram-entry*, the (1,1) element of \( \mathbf{P} \), which is the error variance of \( v_1 \), is an unnamed free parameter.

Covariance parameters are specified in the same manner. For example, the following specification adds a *ram-entry* for the covariance parameter between \( v_2 \) and \( v_3 \):

\[
\text{RAM} \quad \text{var=} \ v_1 \ v_2 \ v_3, \\
\_A\_ \ 1 \ 2 \ \text{parm1}(0.5), \\
\_A\_ \ 1 \ 3 \ 1.0, \\
\_P\_ \ 2 \ 2 \ \text{parm2}, \\
\_P\_ \ 2 \ 3 \ (.5);
\]

The covariance between \( v_2 \) and \( v_3 \) is an unnamed parameter with an initial value of 0.5.

(4) Specification of Means and Intercepts in Model Matrix \( \_W\_ \)

To specifying means or intercepts, you need to start the *ram-entries* with \( \_W\_ \) keyword. For example, the last two entries of following statement specify the intercept of \( v_1 \) and the mean of \( v_2 \), respectively:

\[
\text{RAM} \quad \text{var=} \ v_1 \ v_2 \ v_3, \\
\_A\_ \ 1 \ 2 \ \text{parm1}(0.5), \\
\_A\_ \ 1 \ 3 \ 1.0, \\
\_P\_ \ 2 \ 2 \ \text{parm2}, \\
\_P\_ \ 1 \ 1, \\
\_P\_ \ 2 \ 3 \ (.5), \\
\_W\_ \ 1 \ 1 \ \text{int}_v1, \\
\_W\_ \ 2 \ 2 \ \text{mean}_v2;
\]

The intercept of \( v_1 \) is a free parameter named \( \text{int}_v1 \), and the mean of \( v_2 \) is a free parameter named \( \text{mean}_v2 \).
Default Parameters in the RAM Model

There are two types of default of parameters of the RAM model in PROC CALIS. One is the free parameters; the other is the fixed zeros.

By default, certain sets of model matrix elements in the RAM model are free parameters. These parameters are set automatically by PROC CALIS, although you can also specify them explicitly in the **ram-entries**. In general, default free parameters enable you to specify only what are absolutely necessary for defining your model. PROC CALIS automatically sets those commonly assumed free parameters so that you do not need to specify them routinely. The sets of default free parameters of the RAM model are as follows:

- Diagonal elements of the \( P \) matrix—this includes the variance of exogenous variables (latent or observed) and error variances of all endogenous variables (latent or observed)
- The off-diagonal elements that pertain to the exogenous variables of the \( P \) matrix—this includes all the covariances among exogenous variables, latent or observed
- If the mean structures are modeled, the elements that pertain to the observed variables (but not the latent variables) in the \( W \) vector—this includes all the means of exogenous observed variables and the intercepts of all endogenous observed variables

For example, suppose you are fitting a RAM model with three observed variables \( x_1, x_2, \) and \( y_3 \), you specify a simple multiple-regression model with \( x_1 \) and \( x_2 \) predicting \( y_3 \) by the following statements:

```plaintext
proc calis meanstr;
   ram var= x1-x2 y3,
       _A_ 3 1 ,
       _A_ 3 2 ;
```

In the RAM statement, you specify that path coefficients represented by \( _A_[3,1] \) and \( _A_[3,2] \) are free parameters in the model. In addition to these free parameters, PROC CALIS sets several other free parameters by default. \( _P_[1,1], _P_[2,2], \) and \( _P_[3,3] \) are set as free parameters for the variance of \( x_1 \), the variance of \( x_2 \), and the error variance of \( x_3 \), respectively. \( _P_[2,1] \) (and hence \( _P_[1,2] \)) is set as a free parameter for the covariance between the exogenous variables \( x_1 \) and \( x_2 \). Because the mean structures are also analyzed by the **MEANSTR** option in the PROC CALIS statement, \( _W_[1,1], _W_[2,1], \) and \( _W_[3,1] \) are also set as free parameters for the mean of \( x_1 \), the mean of \( x_2 \), and the intercept of \( x_3 \), respectively. In the current situation, this default parameterization is consistent with using PROC REG for multiple regression analysis, where you only need to specify the functional relationships among variables.

If a matrix element is not a default free parameter in the RAM model, then it is a fixed zero by default. You can override almost all default fixed zeros in the RAM model matrices by specifying the **ram-entries**. The diagonal elements of the \( A \) matrix are exceptions. These elements are always fixed zeros. You cannot set these elements to free parameters or other fixed values—this reflects a model restriction that prevents a variable from having a direct effect on itself.
Naming Variables and Parameters

Follow these rules when you name your variables:

- Use the usual naming conventions of the SAS System.
- Variable names are not more than 32 characters.
- When you create latent variable names, make sure that they are not used in the input data set that is being analyzed.
- For the LINEQS model, error or disturbance variables must start with ‘E’, ‘e’, ‘D’, or ‘d’.
- For the LINEQS model, non-error-type latent variables (that is, factors) must start with ‘F’ or ‘f’.
- For modeling languages other than LINEQS, names for errors or disturbances are not needed. As a result, you do not need to distinguish latent factors from errors or disturbances by using particular prefixes. Variable names that are not referenced in the analyzed data set are supposed to be latent factors.
- You should not use Intercept (case-insensitive) as a variable name in your data set or as a latent variable name in your model.

Follow these rules when you name your parameters:

- Use the usual naming conventions of the SAS System.
- Parameter names are not more than 32 characters.
- Use a prefix-name when you want to generate new parameter names automatically. A prefix-name contains a short string of characters called a “root,” followed by double underscores ‘__’. Each occurrence of such a prefix-name generates a new parameter name by replacing the two trailing underscores with a unique integer. For example, occurrences of Gen__ generate Gen1, Gen2, and so on.
- A special prefix-name is the one without a root—that is, it contains only double underscores ‘__’. Occurrences of ‘__’ generate _Parm1, _Parm2, and so on.
- PROC CALIS generates parameter names for default parameters to safeguard ill-defined models. These generated parameter names start with the _Add prefix and unique integer suffixes. For example, _Add1, _Add2, and so on.
- Avoid using parameter names that start with either __, _Add, or _Parm. These names might get confused with the names generated by PROC CALIS. The confusion might lead to unintended constraints to your model if the parameter names that you use match those generated by PROC CALIS.
- Avoid using parameter names that are roots of prefix-names. For example, you should not use Gen as a parameter name if Gen__ is also used in the same model specification. Although violation of this rule might not distort the model specification, it might cause ambiguities and confusion.
Finally, parameter names and variable names in PROC CALIS are not distinguished by explicit declarations. That is, a valid SAS name can be used as a parameter name or a variable name in any model that is supported by PROC CALIS. Whether a name in a model specification is for a parameter or a variable is determined by the syntactic structure. For example, consider the following path specification:

```sas
proc calis;
    path
        a ===> b   = c;
    run;
```

PROC CALIS parses the path specification according to the syntactic structure of the PATH statement and determines that a and b are variable names and c is a parameter name. Consider another specification as follows:

```sas
proc calis;
    path
        a ===> b   = b;
    run;
```

This is a syntactically correct specification. Variables a and b are defined in a path relationship with a path coefficient parameter also named b. While such a name conflict between parameter and variable names would not confuse PROC CALIS in terms of model specification and fitting, it would create unnecessary confusion in programming and result interpretations. Hence, using parameter names that match variable names exactly is a bad practice and should be avoided.

---

**Setting Constraints on Parameters**

The CALIS procedure offers a very flexible way to constrain parameters. There are two main methods for specifying constraints. One is explicit specification by using specially designed statements for constraints. The other is implicit specification by using the SAS programming statements.

**Explicit Specification of Constraints**

Explicit constraints can be set in the following ways:

- specifying boundary constraints on independent parameters in the **BOUNDS** statement
- specifying general linear equality and inequality constraints on independent parameters in the **LINCON** statement
- specifying general nonlinear equality and inequality constraints on parametric functions in the **NLINCON** statement

**BOUNDS Statement**

You can specify one-sided or two-sided boundaries on independent parameters in the **BOUNDS** statement. For example, in the following statement you constrain parameter `var1` to be nonnegative and parameter `effect` to be between 0 and 5.

```sas
proc calis;
    bounds
        var1 >= 0;
        effect >= 0; effect <= 5;
    run;
```
Setting Constraints on Parameters

**bounds**

```plaintext
  var1 >= 0,
  0. <= effect <= 5.;
```

Note that if your upper and lower bounds are the same for a parameter, it effectively sets a fixed value for that parameter. As a result, PROC CALIS will reduce the number of independent parameters by one automatically. Note also that only independent parameters are allowed to be bounded in the **BOUNDS** statement.

**LINCON**

You can specify equality or inequality linear constraints on independent parameters in the **LINCON** statement. For example, in the following statement you specify a linear inequality constraint on parameters beta1, beta2, and beta3 and an equality constraint on parameters gamma1 and gamma2.

```plaintext
  lincon beta1 - .5 * beta2 - .5 * beta3 >= 0.,
  gamma1 - gamma2 = 0.;
```

In the inequality constraint, beta1 is set to be at least as large as the average of beta2 and beta3. In the equality constraint, gamma1 and gamma2 are set to be equal. Note that in PROC CALIS a nonredundant linear equality constraint on independent parameters effectively reduces the number of parameters by one.

**NLINCON**

You can specify equality or inequality nonlinear constraints for parameters in the **NLINCON** statement. While you can only constrain the independent parameters in the **BOUNDS** and the **LINCON** statements, you can constrain any of the following in the **NLINCON** statement:

- independent parameters
- dependent parameters
- parametric functions computed by the **SAS** programming statements

For example, consider the following statements:

```plaintext
  nlcon
  IndParm >= 0,     /* constraint 1 */
  0 <= DepParm <= 10,  /* constraint 2 */
  ParmFunc1 >= 3, /* constraint 3 */
  0 <= ParmFunc2 <= 8;  /* constraint 4 */
  
  /* SAS Programming statements in the following */
  DepParm = IndParm1 + IndParm5;
  ParmFunc1 = IndParm1 - .5 * IndParm2 - .5 * IndParm3;
  ParmFunc2 = (IndParm1 - 7.)**2 + SQRT(DepParm * IndParm4) * ParmFunc1;
```

You specify four nonlinear constraints by using the **NLINCON** statement. Labeled in a comment as “constraint 1” is a one-sided boundary constraint for independent parameter IndParm. Labeled in a comment as “constraint 2” is a two-sided boundary constraint for dependent parameter DepParm. Labeled in a comment as “constraint 3” is a one-sided inequality constraint on parametric function named ParmFunc1. Finally, labeled in a comment as “constraint 4” is a two-sided inequality constraint on parametric function named ParmFunc2. Parametric functions ParmFunc1 and ParmFunc2 are defined and computed in the **SAS** programming statements after the **NLINCON** statement specification.
Constraint 1 could have been set in the **BOUNDS** statement because it is just a simple boundary constraint on an independent parameter. Constraint 3 could have been set in the **LINCON** statement because the definition of ParmFunc1 in a SAS programming statement shows that it is a linear function of independent parameters. The purpose of including these special cases of “nonlinear constraints” in this example is to show the flexibility of the **NLINCON** statement. However, whenever possible, the **BOUNDS** or the **LINCON** statement specification should be considered first because computationally they are more efficient than the equivalent specification in the **NLINCON** statement.

Specification in the **NLINCON** statement becomes necessary when you want to constrain dependent parameters or nonlinear parametric functions. For example, constraint 2 is a two-sided boundary constraint on the dependent parameter DepParm, which is defined as a linear function of independent parameters in a SAS programming statement. Constraints on dependent parameters are not allowed in the **BOUNDS** statement. Constraint 4 is a two-sided inequality constraint on the nonlinear parametric function ParmFunc2, which is defined as a nonlinear function of other parametric functions and parameters in the SAS programming statements. Again, you cannot use the **LINCON** statement to specify nonlinear constraints.

**Implicit Constraint Specification**

An implicit way to specify constraints is to use your own SAS programming statements together with the **PARAMETERS** statement to express special properties of the parameter estimates. This kind of reparameterization tool is also present in McDonald’s COSAN implementation (McDonald 1978) but is considerably easier to use in the CALIS procedure. PROC CALIS is able to compute analytic first- and second-order derivatives that you would have to specify using the COSAN program.

Some traditional ways to enforce parameter constraints by using reparameterization or parameter transformation (McDonald 1980) are considered in the following:

- **one-sided boundary constraints of the form:**
  
  \[ q \geq a \quad \text{or} \quad q \leq b \]

  where the parameter of interest is \( q \), which should be at least as large as (or at most as small as) a given constant value \( a \) (or \( b \)). This inequality constraint can be expressed as an equality constraint:

  \[ q = a + x^2 \quad \text{or} \quad q = b - x^2 \]

  in which the new parameter \( x \) is unconstrained.

  For example, inequality constraint \( q \geq 7 \) can be accomplished by the following statements:

  ```sas
  parameters  x (0.);
  q = 7 + x * x;
  ```

  In this specification, you essentially redefine \( q \) as a parametric function of \( x \), which is not constrained and has a starting value at 0.

- **two-sided boundary constraints of the form:**

  \[ a \leq q \leq b \]
where the parameter of interest is $q$, which should be located between two given constant values $a$ and $b$, with $a < b$. This inequality constraint can be expressed as the following equality constraint:

$$q = a + (b-a) \frac{\exp(x)}{1 + \exp(x)}$$

where the new parameter $x$ is unconstrained.

For example, to implement $1 \leq q \leq 5$ in PROC CALIS, you can use the following statements:

```sas
parameters x (0.);
  u = exp(x);
  q = 1 + 4 * u / (1 + u);
```

In this specification, $q$ becomes a dependent parameter which is nonlinearly related to independent parameter $x$, which is an independent parameter defined in the PARAMETERS statement with a starting value of 0.

• One-sided order constraints of the form:

$$q_1 \leq q_2, \quad q_1 \leq q_3, \quad \ldots, \quad q_1 \leq q_k$$

where $q_1, \ldots, q_k$ are the parameters of interest. These inequality constraints can be expressed as the following set of equality constraints:

$$q_1 = x_1, \quad q_2 = x_1 + x_2^2, \quad \ldots, \quad q_k = x_1 + x_k^2$$

where the new parameters $x_1, \ldots, x_k$ are unconstrained.

For example, to implement $q_1 \leq q_2$, $q_1 \leq q_3$, and $q_1 \leq q_4$ simultaneously, you can use the following statements:

```sas
parameters x1-x4 (4*0.);
  q1 = x1;
  q2 = x1 + x2 * x2;
  q3 = x1 + x3 * x3;
  q4 = x1 + x4 * x4;
```

In this specification, you essentially redefine $q_1 - q_4$ as dependent parameters that are functions of $x_1 - x_4$, which are defined as independent parameters in the PARAMETERS statement with starting values of zeros. No constraints on $x_i$’s are needed. The way that $q_i$’s are computed in the SAS programming statements guarantees the required order constraints on $q_i$’s are satisfied.

• Two-sided order constraints of the form:

$$q_1 \leq q_2 \leq q_3 \leq \ldots \leq q_k$$

These inequality constraints can be expressed as the following set of equality constraints:

$$q_1 = x_1, \quad q_2 = q_1 + x_2^2, \quad \ldots, \quad q_k = q_{k-1} + x_k^2$$

where the new parameters $x_1, \ldots, x_k$ are unconstrained.
For example, to implement \( q_1 \leq q_2 \leq q_3 \leq q_4 \) simultaneously, you can use the following statements:

```sas
parameters x1-x4 (4*0.);
q1 = x1;
q2 = q1 + x2 * x2;
q3 = q2 + x3 * x3;
q4 = q3 + x4 * x4;
```

In this specification, you redefine \( q_1 - q_4 \) as dependent parameters that are functions of \( x_1 - x_4 \), which are defined as independent parameters in the `PARAMETERS` statement. Each \( x_i \) has a starting value of zero without being constrained in estimation. The order relation of \( q_i \)'s are satisfied by the way they are computed in the SAS programming statements.

- **linear equation constraints of the form:**

\[
\sum_{i}^{k} b_i q_i = a
\]

where \( q_i \)'s are the parameters of interest, \( b_i \)'s are constant coefficients, \( a \) is a constant, and \( k \) is an integer greater than one. This linear equation can be expressed as the following system of equations with unconstrained new parameters \( x_1, x_2, \ldots, x_k \):

\[
q_i = \frac{x_i}{b_i} \quad (i < k)
\]

\[
q_k = \frac{(a - \sum_{j}^{k-1} x_j)}{b_k}
\]

For example, consider the following linear constraint on independent parameters \( q_1 - q_3 \):

\[
3q_1 + 2q_2 - 5q_3 = 8
\]

You can use the following statements to implement the linear constraint:

```sas
parameters x1-x2 (2*0.);
q1 = x1 / 3;
q2 = x2 / 2;
q3 = -(8 - x1 - x2) / 5;
```

In this specification, \( q_1 - q_3 \) become dependent parameters that are functions of \( x1 \) and \( x2 \). The linear constraint on \( q_1 \) and \( q_3 \) are satisfied by the way they are computed. In addition, after reparameterization the number of independent parameters drops to two.

See McDonald (1980) and Browne (1982) for further notes on reparameterization techniques.
Explicit or Implicit Specification of Constraints?

Explicit and implicit constraint techniques differ in their specifications and lead to different computational steps in optimizing a solution. The explicit constraint specification that uses the supported statements incurs additional computational routines within the optimization steps. In contrast, the implicit reparameterization method does not incur additional routines for evaluating constraints during the optimization. Rather, it changes the constrained problem to a non-constrained one. This costs more in computing function derivatives and in storing parameters.

If the optimization problem is small enough to apply the Levenberg-Marquardt or Newton-Raphson algorithm, use the BOUNDS and the LINCON statements to set explicit boundary and linear constraints. If the problem is so large that you must use a quasi-Newton or conjugate gradient algorithm, reparameterization techniques might be more efficient than the BOUNDS and LINCON statements.

Automatic Variable Selection

When you specify your model, you use the main and subsidiary model statements to define variable relationships and parameters. PROC CALIS checks the variables mentioned in these statements against the variable list of the input data set. If a variable in your model is also found in your data set, PROC CALIS knows that it is a manifest variable. Otherwise, it is either a latent variable or an invalid variable.

To save computational resources, only manifest variables defined in your model are automatically selected for analysis. For example, even if you have 100 variables in your input data set, only a covariance matrix of 10 manifest variables is computed for the analysis of the model if only 10 variables are selected for analysis.

In some special circumstances, the automatic variable selection performed for the analysis might be a drawback. For example, if you are interested in modification indices connected to some of the variables that are not used in the model, automatic variable selection in the specification stage will exclude those variables from consideration in computing modification indices. Fortunately, a little trick can be done. You can use the VAR statement to include as many exogenous manifest variables as needed. Any variables in the VAR statement that are defined in the input data set but are not used in the main and subsidiary model specification statements are included in the model as exogenous manifest variables.

For example, the first three steps in a stepwise regression analysis of the Werner Blood Chemistry data (Jöreskog and Sörbom 1988, p. 111) can be performed as follows:

```plaintext
proc calis data=dixon method=gls nobs=180 print mod;
  var x1-x7;
  lineqs y = e;
  variance e = var;
run;
proc calis data=dixon method=gls nobs=180 print mod;
  var x1-x7;
  lineqs y = g1 x1 + e;
  variance e = var;
run;
proc calis data=dixon method=gls nobs=180 print mod;
  var x1-x7;
  lineqs y = g1 x1 + g6 x6 + e;
  variance e = var;
run;
```
In the first analysis, no independent manifest variables are included in the regression equation for dependent variable \( y \). However, \( x_1 - x_7 \) are specified in the VAR statement so that in computing the Lagrange multiplier tests these variables would be treated as potential predictors in the regression equation for dependent variable \( y \). Similarly, in the next analysis, \( x_1 \) is already a predictor in the regression equation, while \( x_2 - x_7 \) are treated as potential predictors in the LM tests. In the last analysis, \( x_1 \) and \( x_6 \) are predictors in the regression equation, while other \( x \)-variables are treated as potential predictors in the LM tests.

### Path Diagrams: Layout Algorithms, Default Settings, and Customization

Path diagrams provide visually appealing representations of the interrelationships among variables in structural equation models. However, it is no easy task to determine the “best” layout of the variables in a path diagram. The CALIS procedure provide three algorithms for laying out good-quality path diagrams. Understanding the underlying principles of these algorithms helps you select the right algorithm and fine-tune the features of a path diagram.

The section “The Process-Flow, Grouped-Flow, and GRIP Layout Algorithms” on page 1440 describes the principles of the algorithms. The sections “Handling Destroyers in Path Diagrams” on page 1448 and “Editing of Path Diagrams with the ODS Graphics Editor” on page 1454 show how you can improve the quality of path diagrams in advance or retroactively. The section “Default Path Diagram Settings” on page 1454 details the default settings for path diagram elements and the corresponding options. By using these options, you can customize path diagrams to meet your own requirements. Finally, the sections “Expanding the Definition of the Structural Variables” on page 1460 and “Showing or Emphasizing the Structural Components” on page 1456 show examples of customizations that you can make to the structural components of models.

### The Process-Flow, Grouped-Flow, and GRIP Layout Algorithms

The most important factor in choosing an appropriate algorithm is the nature of the interrelationships among variables in the model. Depending on the pattern of interrelationships, the CALIS procedure provides three basic algorithms: the process-flow algorithm, grouped-flow algorithm, and GRIP algorithm. You can request these algorithms by specifying the ARRANGE= option in the PATHDIAGRAM statement. This section describes these algorithms and the situations in which they work well. The section also describes how the CALIS procedure selects the “best” algorithm by default (by specifying the ARRANGE=AUTOMATIC option).

#### The Process-Flow Algorithm

The process-flow algorithm (which you select by specifying ARRANGE=FLOW) works well when the interrelationships among the variables in the model (not including the error variables) follow an ideal process-flow pattern. In such a pattern, each variable can be placed at a unique level so that all paths between variables exist only between adjacent levels and are aligned in the same direction. Figure 29.3 shows an example of a linear ordering of observed variables based on the directions of the paths. This model clearly exhibits an ideal process-flow pattern: variables can be uniquely assigned to five levels, and the paths can occur only between adjacent levels and in the same direction.
Figure 29.3 Linear Ordering of Observed Variables

Figure 29.4 shows a hierarchical ordering of the observed variables. The pattern of interrelationships of variables in this model is much like an organizational chart. Clearly, this path diagram also depicts an ideal process-flow pattern.

Figure 29.4 Hierarchical Ordering of Observed Variables

Latent variables are often used in structural equation modeling. Some classes of latent variable models have ideal process-flow patterns. For example, for confirmatory factor models in their “pure” form, the process-flow algorithm can be used to show the hierarchical relationships of factors and variables. Figure 29.5 shows an example of a confirmatory factor model.
Other examples are higher-order factor models and hierarchical factor models, as shown in Figure 29.6 and Figure 29.7, respectively.
Figure 29.6 Linearly Ordered Factors with an Ideal Process-Flow Pattern
Chapter 29: The CALIS Procedure

Figure 29.7 Hierarchical Factor Model That Exhibits an Ideal Process-Flow Pattern

The Grouped-Flow Algorithm
The grouped-flow algorithm works well when the interrelationships of the latent factors follow an ideal process-flow pattern. Such a pattern requires that each latent factor in the model be placed at a unique level so that all paths between the latent factors exist only between adjacent levels and in the same direction. This pattern is called an ideal grouped-flow pattern because each latent factor in the model is associated with a cluster of observed variables. In other words, each cluster of observed variables and a latent factor forms a group. You can specify the ARRANGE=GROUPEDFLOW option in the PATHDIAGRAM statement to request the grouped-flow algorithm.

Figure 29.8 illustrates a linear ordering of latent factors, and Figure 29.9 illustrates a hierarchical ordering of latent factors. Both path diagrams illustrate ideal grouped-flow patterns.
A model that exhibits an ideal grouped-flow pattern for factors can also exhibit an ideal process-flow pattern for non-error variables. The question is then which algorithm is better and under what situations. For example, you can characterize the interrelationships among the variables that are shown in Figure 29.8 by an ideal process-flow pattern of all non-error variables in the model. Figure 29.10 is the diagram that results if you use the `ARRANGE=FLOW` option for this model.
Unfortunately, the latent variables and observed variables in Figure 29.10 are now mixed within the two middle levels, burying the important clusters of observed variables and latent factors that are clearly shown in the grouped-flow representation in Figure 29.8. Hence, the grouped-flow algorithm is more preferable here. For this reason, PROC CALIS adds an extra condition for applying the process-flow algorithm when the default automatic layout method is specified. That is, in order for the process-flow algorithm to be used, variables within each level must also be of the same type (observed or latent), and an ideal process-flow pattern is required in the non-error variables.

**The GRIP Algorithm**
The GRIP (Graph dRrawing with Intelligent Placement) algorithm is a general layout method for graphs that display nodes and links (or variables and paths in the context of path diagrams). Unlike the process-flow and grouped-flow algorithms, the GRIP algorithm is not designed to display specific patterns of variable relationships. Rather, the GRIP algorithm uses a graph-theoretic approach to determine an intelligent initial placement of the nodes (variables). Then the algorithm goes through refinement stages in rendering the final path diagram. Essentially, the GRIP algorithm provides a general layout algorithm when variables or factors in the model cannot be ordered uniquely according to the path directions. The algorithm balances the placement of the variables and groups related variables together.

PROC CALIS uses a modified version of the GRIP algorithm that is more suitable for structural equation models. Basically, error variables are ignored in the layout process (but are displayed in path diagrams if requested), and the lengths of the paths are adjusted according to the types of variables that are being
connected by the paths. These modifications ensure that clusters of observed variables and factors are distinct from each other, as illustrated in Figure 29.11.

**Figure 29.11** Path Diagram That Uses the GRIP Layout Algorithm

In Figure 29.11, Factor1 is the only exogenous variable in the model. This means that no directional path points to Factor1, which is considered to be a level-1 variable. At the next level are the variables that have directional paths from Factor1. Therefore, y1, y2, y3, Factor2, and Factor3 are considered to be at level 2. However, because Factor2 has a directional path to Factor3, Factor3 can also be considered to be at level 3. Hence, this model does not show either an ideal process-flow pattern or an ideal grouped-flow pattern. However, the more general GRIP algorithm can still show three recognizable clusters of variables.

**What does the ARRANGE=AUTOMATIC option actually do?**

By default, the CALIS procedure uses an automatic method to determine the layout algorithm for path diagrams. This automatic method is equivalent to specifying the ARRANGE=AUTOMATIC option. Actually, the AUTOMATIC option is not associated with a specific layout algorithm. What it does is determine automatically which algorithm—the process-flow, grouped-flow, or GRIP algorithm—is the best for a given model.
The following steps describe how PROC CALIS automatically determines the best layout algorithm:

1. PROC CALIS checks whether the interrelationships among all non-error variables in the model exhibit an ideal process-flow pattern. In this ideal pattern, all non-error variables in the model can be uniquely ordered according to the directional paths. PROC CALIS then checks whether variables at each level are of the same type (observed variables or latent factors). If they are, PROC CALIS uses the process-flow algorithm as if the ARRANGE=FLOW option were specified. Otherwise, it continues to the next step.

2. PROC CALIS checks whether the interrelationships among all factors in the model exhibit an ideal process-flow pattern. In this ideal pattern, all latent factors in the model can be uniquely ordered according to the directional paths. If they are, PROC CALIS uses the grouped-flow algorithm as if the ARRANGE=GROUPEDFLOW option were specified. Otherwise, it continues to the next step.

3. PROC CALIS uses the GRIP algorithm as if the ARRANGE=GRIP option were specified.

In summary, when the default ARRANGE=AUTOMATIC option is used, PROC CALIS chooses a suitable layout algorithm by examining the interrelationships of variables. The procedure detects ideal process-flow and grouped-flow patterns when they exist in models.

Handling Destroyers in Path Diagrams

No layout algorithm for path diagrams works perfectly in all situations. Chances are that some paths in your model would violate the ideal pattern that each of the basic layout algorithms assumes. In some cases, even if the violations are minor—for example, they occur in just one or two paths—they are sufficient to throw off the layout of the diagram.

There are two ways for you to alleviate the problems that are caused by these minor violations. One way is to proactively identify the paths in your model that violate the ideal pattern that the intended layout algorithm assumes. If these minor violations are ignored during the layout process, then the layout algorithm can work well for the majority of patterns in the path diagram. To accomplish this, the CALIS procedure provides the DESTROYER= option for you to specify these minor violations. This section presents examples to illustrate the use of this option.

The other way to alleviate these problems is to use the ODS Graphics Editor to improve the graphical quality of the path diagram that PROC CALIS produces. The ODS Graphics Editor provides several functionalities to improve your path diagrams manually. The section “Editing of Path Diagrams with the ODS Graphics Editor” on page 1454 describes these functionalities.
To illustrate the use of the `DESTROYER=` option, consider a higher-order factor model that is specified by the following statements:

```sas
proc calis;
  path
    FactorG ===> Factor1-Factor3,
    Factor1 ===> y1-y3 ,
    Factor2 ===> y4-y6 ,
    Factor3 ===> y7-y9 ,
    FactorG ===> y2 y7 ;
run;
```

In this model, Factor1, Factor2, Factor3, and FactorG are all latent factors. All other variables are observed variables. Which would be the best layout algorithm to draw the path diagram for this model?

Does this model exhibit an ideal process-flow pattern? In this model, FactorG is the only exogenous non-error variable and is considered to be at level 1. Factor1, Factor2, Factor3, y2, and y7 all have directional paths from FactorG, and therefore they are considered to be at level 2. However, because y2 and y7 also have directional paths from Factor1 and Factor3, respectively, they can also be considered to be at level 3. Hence, this model does not exhibit an ideal process-flow pattern.

Does this model exhibit an ideal grouped-flow pattern? If you consider only the factors in the model, FactorG is at level 1 and all other latent factors are at level 2 without ambiguities. Hence, this model exhibits an ideal process-flow pattern in factors or an ideal grouped-flow pattern in all non-error variables. You expect the grouped-flow algorithm to be used to draw the path diagram for this model if you specify the following statements:

```sas
proc calis;
  path
    FactorG ===> Factor1-Factor3,
    Factor1 ===> y1-y3 ,
    Factor2 ===> y4-y6 ,
    Factor3 ===> y7-y9 ,
    FactorG ===> y2 y7 = destroyer1 destroyer2;
  pathdiagram diagram=initial notitle;
run;
```
Figure 29.12 shows the output diagram. Although the path diagram still shows the hierarchical ordering of the latent factors, the two paths, “FactorG ===> y2” and “FactorG ===> y7,” weaken the display of the factor-variable clusters for Factor1 and Factor3. Therefore, these two paths can be called destroyer paths, or simply destroyers. For this reason, the effect parameters of these two paths are labeled as “destroyer1” and “destroyer2” in the PATH statement to show their disruptive characteristics.
What if you ignore these destroyer paths in the layout process of the diagram? To do this, you can use the `DESTROYER=` option in the `PATHDIAGRAM` statement, as follows:

```
pathdiagram diagram=initial notitle destroyer=[FactorG ===> y2 y7];
```

Because the two destroyer paths are ignored when `PROC CALIS` determines the layout algorithm, an ideal process-flow pattern is recognized, and the process-flow algorithm is used to lay out all variables in the model. Then, all paths, including the destroyer paths, are simply put back into the picture to produce the final path diagram, as shown in Figure 29.13.

**Figure 29.13** Higher-Order Factor Model with Two Destroyer Paths Identified

The diagram in Figure 29.13 is preferable to the one in Figure 29.12 because it clearly shows the three levels of variables in the higher-order factor model.
Destroyer paths do not necessarily occur only in ideal process-flow or grouped-flow patterns. Even when the GRIP algorithm is used for a general pattern of variable interrelationships, you might be able to identify potential destroyers in the path diagram. Consider the model that is specified by the following statements:

```sas
proc calis;
   path
      Factor1 ===> y1 y2,
      Factor2 ===> y3 y4,
      Factor3 ===> y5 y6,
      Factor1 ===> Factor2 Factor3,
      Factor2 ===> Factor3,
      y2 ===> y5 = destroyer;
   pathdiagram diagram=initial notitle;
run;
```

This model exhibits neither an ideal process-flow pattern nor an ideal grouped-flow pattern. As a result, PROC CALIS uses the GRIP algorithm automatically to draw the path diagram. Figure 29.14 shows the output diagram.

**Figure 29.14** Path Diagram Whose Destroyer Path Is Not Handled by the GRIP Algorithm
Because of the presence of the destroyer path “y2 ===> y5,” the two factor clusters for Factor1 and Factor3 are drawn closer to each other in the path diagram, thus destroying their distinctive identities that are associated with the related measured variables.

You can use the DESTROYER= option to fix this problem, as shown in the following modified PATHDIAGRAM statement:

```
pathdiagram diagram=initial notitle destroyer=[y2 ===> y5];
```

**Figure 29.15** Path Diagram Whose Destroyer Path Is Handled by the GRIP Algorithm

---

**Figure 29.15** now shows a more preferable path diagram. The factor clusters in this diagram are more distinctive than those in **Figure 29.14**, in which the algorithm does not handle the destroyer path.
Editing of Path Diagrams with the ODS Graphics Editor

You can edit the output path diagram in PROC CALIS by using the ODS Graphics Editor. The ODS Graphics Editor provides the following useful functionalities for editing path diagrams:

- moves the variables around without losing the associated paths
- aligns the selected variables vertically or horizontally
- straightens or curves paths
- reshapes or reroutes paths
- relocates the labels of paths

To use the ODS Graphics Editor, you must first enable the creation of editable graphs. For information about the enabling procedures, see the section “Enabling and Disabling ODS Graphics” on page 606 in Chapter 21, “Statistical Graphics Using ODS.” For complete documentation about the ODS Graphics Editor, see the SAS 9.4 ODS Graphics Editor: User’s Guide. For information about editing nodes and links in output path diagrams, see Chapter 8 of the SAS 9.4 ODS Graphics Editor: User’s Guide. Note that different terminology is used in the SAS 9.4 ODS Graphics Editor: User’s Guide. Variables and paths in the current context are referred to as nodes and links in the User’s Guide. For introductory examples of path diagram editing, see Yung (2014).

Default Path Diagram Settings

Taking the conventions and clarity of graphical representations into consideration, the CALIS procedure employs a set of default settings for producing path diagrams. These default settings define the style or scheme for representing the path diagram elements graphically. For example, PROC CALIS displays the error variances as double-headed paths that are attached to the associated variables. In this way, there is no need to display the error variables so that more space is available to present other important graphical elements in path diagrams.

However, researchers do not always agree on the best style or scheme for representing the path diagram elements. Even the same researcher might want to use different representation style or scheme for his or her path diagrams in different situations. In this regard, you can override most of the default settings to customize path diagrams to meet your own requirements. Table 29.9 summarizes the default graphical and nongraphical settings of the path diagram elements. Related options are listed next to these default settings.

<table>
<thead>
<tr>
<th>Graphical Element or Property</th>
<th>Default Settings</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Graphical Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layout method</td>
<td>Automatically determined</td>
<td>ARRANGE=</td>
</tr>
<tr>
<td>Models with path diagram output</td>
<td>All models</td>
<td>MODEL=</td>
</tr>
<tr>
<td>Path diagrams for structural models</td>
<td>Not shown</td>
<td>STRUCTADD=</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STRUCTURAL</td>
</tr>
<tr>
<td>Graphical Element or Property</td>
<td>Default Settings</td>
<td>Options</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| Paths that affect the layout  | All paths        | DESTROYER=  
|                               |                  | OMITPATH=  |
| Solution type of path diagrams| Unstandardized solution | DIAGRAM=  |
| Structural components        | Not emphasized   | EMPHSTRUCT  
|                               |                  | STRUCTADD=  |
| Paths                         |                  |         |
| Covariances between error variables | Shown as double-headed paths | NOCOV  
|                               |                  | NOERRCOV  |
| Covariances between non-error and error variables | Shown as double-headed paths | NOCOV  |
| Covariances between non-error variables | Shown as double-headed paths in models specified using the FACTOR statement; not shown in other types of models | EXOGCOV  
|                               |                  | NOCOV  
|                               |                  | NOEXOGVAR  |
| Directional paths            | Shown as single-headed paths | OMITPATH=  |
| Error variances              | Shown as double-headed paths in covariance models; shown as labels in mean and covariance models | NOERRVAR  
|                               |                  | NOVARIANCE  
|                               |                  | VARPARM=  |
| Means and intercepts         | Not shown as paths; shown as labels in mean and covariance models | MEANPARM=  
|                               |                  | NOMEAN  |
| Variances of non-error variables | Shown as double-headed paths in covariance models; shown as labels in mean and covariance models | VARPARM=  |
| Variables                    |                  |         |
| Error variables              | Not shown        | USEERROR  |
| Relative sizes               | Observed variables:Factors:Errors = 1:1.5:0.5 | ERRORSIZE=  
|                               |                  | FACTORSIZE=  |
| Variable labels              | Original variable names used | LABEL=  |
| Parameter Estimates          |                  |         |
| Covariances between error variables | Shown in all types of models | NOCOV  
|                               |                  | NOERRCOV  |
| Covariances between non-error variables | Shown in models specified using the FACTOR statement; not shown in other types of models | EXOGCOV  
|                               |                  | NOCOV  
|                               |                  | NOEXOGCOV  |
| Initial parameter specifications | Shown with fixed values and user-specified parameter names | NOINITPARM  |
| Means and intercepts         | Shown in models with mean structures | NOMEAN  |
Table 29.9  continued

<table>
<thead>
<tr>
<th>Graphical Element or Property</th>
<th>Default Settings</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variances of error variables</td>
<td>Shown in all types of models</td>
<td>NOERRVAR, NOVARIANCE</td>
</tr>
<tr>
<td>Variances of non-error variables</td>
<td>Shown in all types of models</td>
<td>NOEXOGVAR, NOVARIANCE</td>
</tr>
</tbody>
</table>

Formats of Parameter Estimates

<table>
<thead>
<tr>
<th></th>
<th>Default Settings</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal places</td>
<td>Two decimal places used</td>
<td>DECP=</td>
</tr>
<tr>
<td>Numerical values</td>
<td>Shown in unstandardized and standardized solutions</td>
<td>NOESTIM</td>
</tr>
<tr>
<td>Parameter names</td>
<td>Not shown</td>
<td>PARMNAMES</td>
</tr>
<tr>
<td>Significance flags</td>
<td>Shown in unstandardized and standardized solutions</td>
<td>NOFLAG</td>
</tr>
</tbody>
</table>

Graph Title and Fit Summary

<table>
<thead>
<tr>
<th></th>
<th>Default Settings</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal places in fit summary</td>
<td>Two decimal places used</td>
<td>DECPFIT=</td>
</tr>
<tr>
<td>Fit table</td>
<td>Shown in unstandardized and standardized solutions</td>
<td>NOFITTABLE</td>
</tr>
<tr>
<td>Fit table contents</td>
<td>A default set of available information and statistics</td>
<td>FITINDEX=</td>
</tr>
<tr>
<td>Title</td>
<td>Default title that reflects the identity and solution type of the model</td>
<td>NOTITLE, TITLE=</td>
</tr>
</tbody>
</table>

Showing or Emphasizing the Structural Components

One often-used customization in structural equation modeling is to display only the structural component (or structural model) of the full model, which consists of the measurement and the structural components. Traditionally, the structural component refers to the latent factors and their interrelationships, whereas the measurement component refers to observed indicator variables and their relationships with the latent factors.

For example, consider the full model that is displayed as a path diagram in Figure 29.16. The latent factors are Factor1, Factor2, and Factor3. The remaining variables, y1–y9, are observed variables.
The structural component of this full model includes the three latent factors, their directional relationships (represented by the three directed paths), and their variance parameters (represented by the three double-headed arrows that are attached to the factors).

Showing only the structural component is useful when your research is focused mainly on the interrelationships among the latent factors. The measurement component, which usually contains a relatively large number of observed variables for reflecting or defining the latent factors, is of secondary interest. Eliminating the measurement component leads to a clearer path diagram to represent your model.
To request a path diagram that shows the structural component, you can specify the `STRUCTURAL` option in the `PATHDIAGRAM` statement, as in the following statements:

```plaintext
proc calis;
  path
      Factor1 ===> y1-y3 ,
      Factor2 ===> y4-y6 ,
      Factor3 ===> y7-y9 ,
      Factor1 ===> Factor2 Factor3,
      Factor2 ===> Factor3;
  pathdiagram diagram=initial structural;
run;
```

**Figure 29.17** Path Diagram for the Structural Component

![Path Diagram](image)

Figure 29.17 shows the path diagram for the structural model. For simplicity in presentation, the `DIAGRAM=INITIAL` option is used to show the initial model specifications only.

The preceding specification produces the path diagram for the structural component, in addition to that for the full model. To output the path diagram for the structural component only, use the `STRUCTURAL(ONLY)` option in the `PATHDIAGRAM` statement, as follows:

```plaintext
pathdiagram diagram=initial structural(only);
```
Instead of eliminating the measurement component from the path diagram, PROC CALIS provides the EMPHSTRUCT option to highlight the structural component in the full model. For example, the following PATHDIAGRAM statement produces the path diagram in Figure 29.17:

```
pathdiagram diagram=initial emphstruct;
```

**Figure 29.18**  Emphasizing the Structural Component in the Path Diagram

![Path Diagram](image)

Figure 29.18 shows a complete model, with emphasis on the structural component. The path diagram labels only the latent factors. Observed variables are represented by small rectangles and are not labeled.

Certainly, the EMPHSTRUCT option is not limited to the initial path diagram, such as the one shown in Figure 29.18. For the unstandardized and standardized solutions, the EMPHSTRUCT option puts similar emphasis on the structural variables. In addition, the diagram displays only the parameter estimates of the structural components. Figure 29.19 shows the path diagram of an unstandardized solution that emphasizes the structural components.
The traditional definition of “structural model” or “structural component” might be too restrictive, in the sense that only latent factors and their interrelationships are included. In contemporary structural equation modeling, sometimes observed variables can take the role of “factors” in the structural component. For example, observed variables that have negligible measurement errors and serve as exogenous or predictor variables in a system of functional equations naturally belong to the structural component. For this reason or other reasons, researchers might want to expand the definition of “structural component” to include some designated observed variables. The \texttt{STRUCTADD=} option enables you to do that.

For example, the following statements specify a model that has two latent factors, \texttt{Factor1} and \texttt{Factor2}:

```plaintext
proc calis;
  path
    Factor1 ===> y1-y3,
    Factor2 ===> y4-y6,
    y7 ===> Factor1 Factor2;
  pathdiagram diagram=initial notitle structadd=[y7]
    label=[y7="y7 is Exogenous"] ;
  pathdiagram diagram=initial notitle structadd=[y7] struct(only);
    label=[y7="y7 is Exogenous"] ;
  pathdiagram diagram=initial notitle structadd=[y7] emphstruct;
    label=[y7="y7 is Exogenous"] ;
run;
```
By default, only the latent factors Factor1 and Factor2 are treated as structural variables. However, because the observed variable y7 is exogenous and is a predictor of the two latent variables, it is not unreasonable to also treat it as a structural variable when you are producing path diagrams. Hence, all three PATHDIAGRAM statements use the STRUCTADD= option to include y7 in the set of structural variables. For illustration purposes, variable y7 is labeled as “y7 is Exogenous” in the path diagrams. Although all three PATHDIAGRAM statements produce diagrams for the initial solution, they each display the structural component in a different way.

The first PATHDIAGRAM statement produces a diagram for the full model in Figure 29.20. The structural component is not specifically emphasized. Because y7 is treated as a structural variable, it is larger than the other observed variables, and its size is comparable to that of the default structural variables Factor1 and Factor2.

Figure 29.20 Including an Observed Variable as a Structural Variable: Full Model

The second PATHDIAGRAM statement produces a diagram that emphasizes the structural component of the full model in Figure 29.21. Again, because y7 is treated as a structural variable, it is much larger than the other observed variables, and it is emphasized in the same way as the default structural variables Factor1 and Factor2.
The third PATHDIAGRAM statement produces a diagram for the structural component in Figure 29.22. Because y7 is treated as a structural variable, it is also shown in this path diagram, and its size is comparable to that of Factor1 and Factor2.
If you do not include $y_7$ as a structural variable in this PATHDIAGRAM statement, Figure 29.22 would display Factor1 and Factor2 as isolated variables that do not have any directional paths that connect them, making it an uninteresting path diagram display.

**Estimation Criteria**

The following seven estimation methods are available in PROC CALIS:

- unweighted least squares (ULS)
- full information maximum likelihood (FIML)
- generalized least squares (GLS)
- normal-theory maximum likelihood (ML)
- weighted least squares (WLS, ADF)
• diagonally weighted least squares (DWLS)
• robust estimation (ROBUST) under the normal theory (ML)

Default weight matrices $W$ are computed for GLS, WLS, and DWLS estimation. You can also provide your own weight matrices by using an INWGT= data set. The weight matrices in these estimation methods provide weights for the moment matrices. In contrast, weights that are applied to individual observations are computed in robust estimation. These observation weights are updated during iteration steps of robust estimation. The ULS, GLS, ML, WLS, ADF, and DWLS methods can analyze sample moment matrices as well as raw data, while the FIML and robust methods must analyze raw data.

PROC CALIS does not implement all estimation methods in the field. As mentioned in the section “Overview: CALIS Procedure” on page 1158, partial least squares (PLS) is not implemented. The PLS method is developed under less restrictive statistical assumptions. It circumvents some computational and theoretical problems encountered by the existing estimation methods in PROC CALIS; however, PLS estimates are less efficient in general. When the statistical assumptions of PROC CALIS are tenable (for example, large sample size, correct distributional assumptions, and so on), ML, GLS, or WLS methods yield better estimates than the PLS method. Note that there is a SAS/STAT procedure called PROC PLS that employs the partial least squares technique, but for a different class of models than those of PROC CALIS. For example, in a PROC CALIS model each latent variable is typically associated with only a subset of manifest variables (predictor or outcome variables). However, in PROC PLS latent variables are not prescribed with subsets of manifest variables. Rather, they are extracted from linear combinations of all manifest predictor variables. Therefore, for general path analysis with latent variables you should use PROC CALIS.

**ULS, GLS, and ML Discrepancy Functions**

In each estimation method, the parameter vector is estimated iteratively by a nonlinear optimization algorithm that minimizes a discrepancy function $F$, which is also known as the fit function in the literature. With $p$ denoting the number of manifest variables, $S$ the sample $p \times p$ covariance matrix for a sample with size $N$, $\bar{x}$ the $p \times 1$ vector of sample means, $\Sigma$ the fitted covariance matrix, and $\mu$ the vector of fitted means, the discrepancy function for unweighted least squares (ULS) estimation is:

$$F_{ULS} = 0.5 \text{Tr}[(S - \Sigma)^2] + (\bar{x} - \mu)'(\bar{x} - \mu)$$

The discrepancy function for generalized least squares estimation (GLS) is:

$$F_{GLS} = 0.5 \text{Tr}[(W^{-1}(S - \Sigma))^2] + (\bar{x} - \mu)'W^{-1}(\bar{x} - \mu)$$

By default, $W = S$ is assumed so that $F_{GLS}$ is the normal theory generalized least squares discrepancy function.

The discrepancy function for normal-theory maximum likelihood estimation (ML) is:

$$F_{ML} = \text{Tr}(S\Sigma^{-1}) - p + \ln(|\Sigma|) - \ln(|S|) + (\bar{x} - \mu)'\Sigma^{-1}(\bar{x} - \mu)$$

In each of the discrepancy functions, $S$ and $\bar{x}$ are considered to be given and $\Sigma$ and $\mu$ are functions of model parameter vector $\Theta$. That is:

$$F = F(\Theta, \mu(\Theta); S, \bar{x})$$

Estimating $\Theta$ by using a particular estimation method amounts to choosing a vector $\theta$ that minimizes the corresponding discrepancy function $F$. 


When the mean structures are not modeled or when the mean model is saturated by parameters, the last term of each fit function vanishes. That is, they become:

\[
F_{ULS} = 0.5 \text{Tr}((S - \Sigma)^2) \\
F_{GLS} = 0.5 \text{Tr}((W^{-1}(S - \Sigma))^2) \\
F_{ML} = \text{Tr}(S\Sigma^{-1}) - p + \ln(|\Sigma|) - \ln(|S|)
\]

If, instead of being a covariance matrix, \( S \) is a correlation matrix in the discrepancy functions, \( \Sigma \) would naturally be interpreted as the fitted correlation matrix. Although whether \( S \) is a covariance or correlation matrix makes no difference in minimizing the discrepancy functions, correlational analyses that use these functions are problematic because of the following issues:

- The diagonal of the fitted correlation matrix \( \Sigma \) might contain values other than ones, which violates the requirement of being a correlation matrix.
- Whenever available, standard errors computed for correlation analysis in PROC CALIS are straightforward generalizations of those of covariance analysis. In very limited cases these standard errors are good approximations. However, in general they are not even asymptotically correct.
- The model fit chi-square statistic for correlation analysis might not follow the theoretical distribution, thus making model fit testing difficult.

Despite these issues in correlation analysis, if your primary interest is to obtain the estimates in the correlation models, you might still find PROC CALIS results for correlation analysis useful.

The statistical techniques used in PROC CALIS are primarily developed for the analysis of covariance structures, and hence COVARIANCE is the default option. Depending on the nature of your research, you can add the mean structures in the analysis by specifying mean and intercept parameters in your models. However, you cannot analyze mean structures simultaneously with correlation structures (see the CORRELATION option) in PROC CALIS.

**FIML Discrepancy Function**

The full information maximum likelihood method (FIML) assumes multivariate normality of the data. Suppose that you analyze a model that contains \( p \) observed variables. The discrepancy function for FIML is

\[
F_{FIML} = \frac{1}{N} \sum_{j=1}^{N} \left( \ln(|\Sigma_j|) + (x_j - \mu_j)'\Sigma_j^{-1}(x_j - \mu_j) + K_j \right)
\]

where \( x_j \) is a data vector for observation \( j \), and \( K_j \) is a constant term (to be defined explicitly later) independent of the model parameters \( \Theta \). In the current formulation, \( x_j \)'s are not required to have the same dimensions. For example, \( x_1 \) could be a complete vector with all \( p \) variables present while \( x_2 \) is a \((p - 1) \times 1\) vector with one missing value that has been excluded from the original \( p \times 1 \) data vector. As a consequence, subscript \( j \) is also used in \( \mu_j \) and \( \Sigma_j \) to denote the submatrices that are extracted from the entire \( p \times 1 \) structured mean vector \( \mu \) (\( \mu = \mu(\Theta) \)) and \( p \times p \) covariance matrix \( \Sigma \) (\( \Sigma = \Sigma(\Theta) \)). In other words, in the current formulation \( \mu_j \) and \( \Sigma_j \) do not mean that each observation is fitted by distinct mean and covariance structures (although theoretically it is possible to formulate FIML in such a way). The notation simply
signifies that the dimensions of \( x_j \) and of the associated mean and covariance structures could vary from observation to observation.

Let \( p_j \) be the number of variables without missing values for observation \( j \). Then \( x_j \) denotes a \( p_j \times 1 \) data vector, \( \mu_j \) denotes a \( p_j \times 1 \) vector of means (structured with model parameters), \( \Sigma_j \) is a \( p_j \times p_j \) matrix for variances and covariances (also structured with model parameters), and \( K_j \) is defined by the following formula, which is a constant term independent of model parameters:

\[
K_j = \ln(2\pi) \times p_j
\]

As a general estimation method, the FIML method is based on the same statistical principle as the ordinary maximum likelihood (ML) method for multivariate normal data—that is, both methods maximize the normal theory likelihood function given the data. In fact, \( F_{\text{FIML}} \) used in PROC CALIS is related to the log-likelihood function \( L \) by the following formula:

\[
F_{\text{FIML}} = \frac{-2L}{N}
\]

Because the FIML method can deal with observations with various levels of information available, it is primarily developed as an estimation method that could deal with data with random missing values. See the section “Relationships among Estimation Criteria” on page 1473 for more details about the relationship between FIML and ML methods.

Whenever you use the FIML method, the mean structures are automatically assumed in the analysis. This is due to fact that there is no closed-form formula to obtain the saturated mean vector in the FIML discrepancy function if missing values are present in the data. You can certainly provide explicit specification of the mean parameters in the model by specifying intercepts in the LINEQS statement or means and intercepts in the MEAN or MATRIX statement. However, usually you do not need to do the explicit specification if all you need to achieve is to saturate the mean structures with \( p \) parameters (that is, the same number as the number of observed variables in the model). With METHOD=FIML, PROC CALIS uses certain default parameterizations for the mean structures automatically. For example, all intercepts of endogenous observed variables and all means of exogenous observed variables are default parameters in the model, making the explicit specification of these mean structure parameters unnecessary.

### WLS and ADF Discrepancy Functions

Another important discrepancy function to consider is the weighted least squares (WLS) function. Let \( u = (s, \bar{x}) \) be a \( p(p + 3)/2 \) vector containing all nonredundant elements in the sample covariance matrix \( S \) and sample mean vector \( \bar{x} \), with \( s = \text{vecs}(S) \) representing the vector of the \( p(p + 1)/2 \) lower triangle elements of the symmetric matrix \( S \), stacking row by row. Similarly, let \( \eta = (\sigma, \mu) \) be a \( p(p + 3)/2 \) vector containing all nonredundant elements in the fitted covariance matrix \( \Sigma \) and the fitted mean vector \( \mu \), with \( \sigma = \text{vecs}(\Sigma) \) representing the vector of the \( p(p + 1)/2 \) lower triangle elements of the symmetric matrix \( \Sigma \).

The WLS discrepancy function is:

\[
F_{\text{WLS}} = (u - \eta)'W^{-1}(u - \eta)
\]

where \( W \) is a positive definite symmetric weight matrix with \( (p(p + 3)/2) \) rows and columns. Because \( \eta \) is a function of model parameter vector \( \Theta \) under the structural model, you can write the WLS function as:

\[
F_{\text{WLS}} = (u - \eta(\Theta))'W^{-1}(u - \eta(\Theta))
\]
Suppose that \( u \) converges to \( \eta_o = (\sigma_o, \mu_o) \) with increasing sample size, where \( \sigma_o \) and \( \mu_o \) denote the population covariance matrix and mean vector, respectively. By default, the WLS weight matrix \( W \) in PROC CALIS is computed from the raw data as a consistent estimate of the asymptotic covariance matrix \( \Gamma \) of \( \sqrt{N}(u - \eta_o) \), with \( \Gamma \) partitioned as

\[
\Gamma = \begin{pmatrix}
\Gamma_{ss} & \Gamma_{x\bar{x}}' \\
\Gamma_{x\bar{x}} & \Gamma_{x\bar{x}}
\end{pmatrix}
\]

where \( \Gamma_{ss} \) denotes the \( (p(p + 1)/2) \times (p(p + 1)/2) \) asymptotic covariance matrix for \( \sqrt{N}(s - \sigma_o) \), \( \Gamma_{x\bar{x}} \) denotes the \( p \times p \) asymptotic covariance matrix for \( \sqrt{N} (\bar{x} - \mu_o) \), and \( \Gamma_{x\bar{x}} \) denotes the \( p \times (p(p + 1)/2) \) asymptotic covariance matrix between \( \sqrt{N}(\bar{x} - \mu_o) \) and \( \sqrt{N}(s - \sigma_o) \).

To compute the default weight matrix \( W \) as a consistent estimate of \( \Gamma \), define a similar partition of the weight matrix \( W \) as:

\[
W = \begin{pmatrix}
W_{ss} & W_{x\bar{x}}' \\
W_{x\bar{x}} & W_{x\bar{x}}
\end{pmatrix}
\]

Each of the submatrices in the partition can now be computed from the raw data. First, define the biased sample covariance for variables \( i \) and \( j \) as:

\[
t_{ij} = \frac{1}{N} \sum_{r=1}^{N} (x_{ri} - \bar{x}_i)(x_{rj} - \bar{x}_j)
\]

and the sample fourth-order central moment for variables \( i, j, k, \) and \( l \) as:

\[
t_{ij,kl} = \frac{1}{N} \sum_{r=1}^{N} (x_{ri} - \bar{x}_i)(x_{rj} - \bar{x}_j)(x_{rk} - \bar{x}_k)(x_{rl} - \bar{x}_l)
\]

The submatrices in \( W \) are computed by:

\[
[W_{ss}]_{ij,kl} = t_{ij,kl} - t_{ij} t_{kl}
\]

\[
[W_{x\bar{x}}]_{i,kl} = \frac{1}{N} \sum_{r=1}^{N} (x_{ri} - \bar{x}_i)(x_{rk} - \bar{x}_k)(x_{rl} - \bar{x}_l)
\]

\[
[W_{x\bar{x}}]_{ij} = t_{ij}
\]

Assuming the existence of finite eighth-order moments, this default weight matrix \( W \) is a consistent but biased estimator of the asymptotic covariance matrix \( \Gamma \).

By using the ASYCOV= option, you can use Browne’s unbiased estimator (Browne 1984, formula (3.8)) of \( \Gamma_{ss} \) as:

\[
[W_{ss}]_{ij,kl} = \frac{N(N - 1)}{(N - 2)(N - 3)} (t_{ij,kl} - t_{ij} t_{kl}) - \frac{N}{(N - 2)(N - 3)} (t_{ik} t_{jl} + t_{il} t_{jk} - \frac{2}{N - 1} t_{ij} t_{kl})
\]
There is no guarantee that $W_{ss}$ computed this way is positive semidefinite. However, the second part is of order $O(N^{-1})$ and does not destroy the positive semidefinite first part for sufficiently large $N$. For a large number of independent observations, default settings of the weight matrix $W$ result in asymptotically distribution-free parameter estimates with unbiased standard errors and a correct $\chi^2$ test statistic (Browne 1982, 1984).

With the default weight matrix $W$ computed by PROC CALIS, the WLS estimation is also called as the asymptotically distribution-free (ADF) method. In fact, as options in PROC CALIS, METHOD=WLS and METHOD=ADF are totally equivalent, even though WLS in general might include cases with special weight matrices other than the default weight matrix.

When the mean structures are not modeled, the WLS discrepancy function is still the same quadratic form statistic. However, with only the elements in covariance matrix being modeled, the dimensions of $u$ and $\eta$ are both reduced to $p(p+1)/2 \times 1$, and the dimension of the weight matrix is now $(p(p+1)/2) \times (p(p+1)/2)$. That is, the WLS discrepancy function for covariance structure models is:

$$F_{WLS} = (s - \sigma)' W_{ss}^{-1} (s - \sigma)$$

If $S$ is a correlation rather than a covariance matrix, the default setting of the $W_{ss}$ is a consistent estimator of the asymptotic covariance matrix $\Gamma_{ss}$ of $\sqrt{N}(s - \sigma_o)$ (Browne and Shapiro 1986; De Leeuw 1983), with $s$ and $\sigma_o$ representing vectors of sample and population correlations, respectively. Elementwise, $W_{ss}$ is expressed as:

$$[W_{ss}]_{ij,kl} = r_{ij,kl} - \frac{1}{2} r_{ij} (r_{ii,kl} + r_{jj,kl}) - \frac{1}{2} r_{kl} (r_{kk,ij} + r_{ll,ij})$$

$$+ \frac{1}{4} r_{ij} r_{kl} (r_{ii,kl} + r_{ii,kl}) (r_{jj,kl} + r_{jj,kl})$$

where

$$r_{ij} = \frac{t_{ij}}{\sqrt{t_{ii} t_{jj}}}$$

and

$$r_{ij,kl} = \frac{t_{ij,kl}}{\sqrt{t_{ii} t_{jj} t_{kk} t_{ll}}}$$

The asymptotic variances of the diagonal elements of a correlation matrix are 0. That is,

$$[W_{ss}]_{ii,ii} = 0$$

for all $i$. Therefore, the weight matrix computed this way is always singular. In this case, the discrepancy function for weighted least squares estimation is modified to:

$$F_{WLS} = \sum_{i=2}^{p} \sum_{j=1}^{i-1} \sum_{k=2}^{p} \sum_{l=1}^{k-1} [W_{ss}]_{ij,kl} ([S]_{ij} - [\Sigma]_{ij}) ([S]_{kl} - [\Sigma]_{kl})$$

$$+ r \sum_{i=1}^{p} ([S]_{ii} - [\Sigma]_{ii})^2$$
where \( r \) is the penalty weight specified by the \texttt{WPENALTY=} \( r \) option and the \( W_{ss}^{ij,kl} \) are the elements of the inverse of the reduced \( (p(p-1)/2) \times (p(p-1)/2) \) weight matrix that contains only the nonzero rows and columns of the full weight matrix \( W_{ss} \).

The second term is a penalty term to fit the diagonal elements of the correlation matrix \( S \). The default value of \( r = 100 \) can be decreased or increased by the \texttt{WPENALTY=} option. The often used value of \( r = 1 \) seems to be too small in many cases to fit the diagonal elements of a correlation matrix properly.

Note that when you model correlation structures, no mean structures can be modeled simultaneously in the same model.

**DWLS Discrepancy Functions**

Storing and inverting the huge weight matrix \( W \) in WLS estimation requires considerable computer resources. A compromise is found by implementing the diagonally weighted least squares (DWLS) method that uses only the diagonal of the weight matrix \( W \) from the WLS estimation in the following discrepancy function:

\[
F_{DWLS} = (u - \eta)'[\text{diag}(W)]^{-1}(u - \eta) = \sum_{i=1}^{p} \sum_{j=1}^{p} [W_{ss}^{ij,ij}]^{1/2} (S_{ij} - \Sigma_{ij})^2 + \sum_{i=1}^{p} [W_{ss}^{i,ij}]^{1/2} (\hat{\xi}_i - \mu_i)^2
\]

When only the covariance structures are modeled, the discrepancy function becomes:

\[
F_{DWLS} = \sum_{i=1}^{p} \sum_{j=1}^{p} [W_{ss}^{ij,ij}]^{-1} (S_{ij} - \Sigma_{ij})^2
\]

For correlation models, the discrepancy function is:

\[
F_{DWLS} = \sum_{i=2}^{p} \sum_{j=1}^{i-1} [W_{ss}^{ij,ij}]^{-1} (S_{ij} - \Sigma_{ij})^2 + r \sum_{i=1}^{p} (S_{ii} - \Sigma_{ii})^2
\]

where \( r \) is the penalty weight specified by the \texttt{WPENALTY=} \( r \) option. Note that no mean structures can be modeled simultaneously with correlation structures when using the DWLS method.

As the statistical properties of DWLS estimates are still not known, standard errors for estimates are not computed for the DWLS method.

**Input Weight Matrices**

In GLS, WLS, or DWLS estimation you can change from the default settings of weight matrices \( W \) by using an \texttt{INWGT=} data set. The CALIS procedure requires a positive definite weight matrix that has positive diagonal elements.

**Multiple-Group Discrepancy Function**

Suppose that there are \( k \) independent groups in the analysis and \( N_1, N_2, \ldots, N_k \) are the sample sizes for the groups. The overall discrepancy function \( F(\Theta) \) is expressed as a weighted sum of individual discrepancy functions \( F_i \)'s for the groups:

\[
F(\Theta) = \sum_{i=1}^{k} t_i F_i(\Theta)
\]
where

\[ t_i = \frac{N_i - 1}{N - k} \]

is the weight of the discrepancy function for group \( i \), and

\[ N = \sum_{i=1}^{k} N_i \]

is the total number of observations in all groups. In PROC CALIS, all discrepancy function \( F_i \)'s in the overall discrepancy function must belong to the same estimation method. You cannot specify different estimation methods for the groups in a multiple-group analysis. In addition, the same analysis type must be applied to all groups—that is, you can analyze either covariance structures, covariance and mean structures, and correlation structures for all groups.

Robust Estimation

Two robust estimation methods that are proposed by Yuan and Zhong (2008) and Yuan and Hayashi (2010) are implemented in PROC CALIS. The first method is the two-stage robust method, which estimates robust covariance and mean matrices in the first stage and then feeds the robust covariance and mean matrices (in place of the ordinary sample covariance and mean matrices) for ML estimation in the second stage. Weighting of the observations is done only in the first stage. The ROBUST=SAT option invokes the two-stage robust estimation. The second method is the direct robust method, which iteratively estimates model parameters with simultaneous weightings of the observations. The ROBUST, ROBUST=RES(E), or ROBUST=RES(F) option invokes the direct robust estimation method.

The procedural difference between the two robust methods results in differential treatments of model outliers and leverage observations (or leverage points). In producing the robust covariance and mean matrices in the first stage, the two-stage robust method downweights outlying observations in all variable dimensions without regard to the model structure. This method downweights potential model outliers and leverage observations (which are not necessarily model outliers) in essentially the same way before the ML estimation in the second stage.

However, the direct robust method downweights the model outliers only. “Good” leverage observations (those that are not outliers at the same time) are not downweighted for model estimation. Therefore, it could be argued that the direct robust method is more desirable if you can be sure that the model is a reasonable one. The reason is that the direct robust method can retain the information from the “good” leverage observations for estimation, while the two-stage robust method downweights all leverage observations indiscriminately during its first stage. However, if the model is itself uncertain, the two-stage robust estimation method might be more foolproof.

Both robust methods employ weights on the observations. Weights are functions of the Mahalanobis distances (M-distances) of the observations and are computed differently for the two robust methods. The following two sections describe the weighting scheme and the estimation procedure of the two robust methods in more detail.
Two-Stage Robust Method

For the two-stage robust method, the following conventional M-distance $d_s$ for an observed random vector $x$ is computed as

$$d_s = \sqrt{(x - \mu)'\Sigma^{-1}(x - \mu)}$$

where $\mu$ and $\Sigma$ are the unstructured mean and covariance matrices, respectively.

Two sets of weights are computed as functions of the M-distances of the observations. The weighting functions are essentially the same form as that of Huber (see, for example, Huber 1981). Let $d_i$ be the M-distance of observation $i$, computed from the $d_s$ formula. The first set of weights $w_1(d_i)$ applies to the first moments of the data and is defined as

$$w_1(d_i) = \begin{cases} 
1 & \text{if } d_i \leq \rho \\
\rho / d_i & \text{if } d_i > \rho 
\end{cases}$$

where $\rho$ is the critical value corresponding to the $(1 - \phi) \times 100$ quantile of the $\chi^2_r = \sqrt{\chi^2_r}$ distribution, with $r$ being the degrees of freedom. For the two-stage robust method, $r$ is simply the number of observed variables in the analysis. The tuning parameter $\phi$ controls the approximate proportion of observations to be downweighted (that is, with $w_1(d_i)$ less than 1). The default $\phi$ value is set to 0.05. You can override this value by using the ROBPHI= option.

The second set of weights $w_2(d_i)$ applies to the second moments of the data and is defined as

$$w_2(d_i) = \frac{(w_1(d_i))^2}{\kappa}$$

where $\kappa$ is a constant that adjusts the sum $\sum_{i=1}^{N} w_2(d_i)$ to 1 approximately. After the tuning parameter $\phi$ is determined, the critical value $\rho$ and the adjustment $\kappa$ are computed automatically by PROC CALIS.

With these two sets of weights, the two-stage robust method (ROBUST=SAT) estimates the mean and covariance by the so-called iteratively reweighted least squares (IRLS) algorithm. Specifically, the updating formulas at the $j+1$ iteration are

$$\mu^{(j+1)} = \frac{\sum_{i=1}^{N} w_1(d_i^{(j)})x_i}{\sum_{i=1}^{N} w_1(d_i^{(j)})}$$

$$\Sigma^{(j+1)} = \frac{1}{N} \sum_{i=1}^{N} w_2(d_i^{(j)})(x_i - \mu^{(j)})(x_i - \mu^{(j)})'$$

where $d_i^{(j)}$ is the M-distance evaluated at $\mu^{(j)}$ and $\Sigma^{(j)}$ obtained in the $j$th iteration. Carry out the iterations until $\mu$ and $\Sigma$ converge. The final iteration yields the robust estimates of mean and covariance matrices. PROC CALIS uses the relative parameter convergence criterion for the IRLS algorithm. The default criterion is $1E^{-8}$. See the XCONV= option for the definition of the relative parameter convergence criterion. After the IRLS algorithm converges in the first stage, the two-stage robust method proceeds to treat the robust mean and covariance estimates as if they were sample mean and covariance matrices for a maximum likelihood estimation (METHOD=ML) of the model.
**Direct Robust Method**

The direct robust method computes the following residual M-distance $d_r$ for an observation with residual random vector $\hat{e}$ (say, of dimension $h \times 1$, where $h$ is the number of dependent observed variables $y$):

$$d_r = \sqrt{(L\hat{e})'(L\Omega_\hat{e}L')^{-1}(L\hat{e})}$$

where $L ((h - q) \times h)$ is a loading matrix that reduces $\hat{e}$ to $(h - q)$ independent components and $q$ is the number of independent factors to be estimated from the dependent observed variables $y$. The reduction of the residual vector into independent components is necessary when the number of factors $q$ is not zero in the model. For $q > 0$, the residual covariance matrix $\Omega_\hat{e}$ is not invertible and cannot be used in computing the residual M-distances. Hence, the covariance matrix $L\Omega_\hat{e}L'$ of independent components $L\hat{e}$ is used instead. See Yuan and Hayashi (2010) for details about the computation of the residuals in the context of structural equation modeling.

The direct robust method also computes two sets of weights as functions of the residual M-distances. Let $d_i$ be the M-distance of observation $i$, computed from the $d_r$ formula. The first set of weights $w_1(d_i)$ applies to parameters in the first moments and is defined as

$$w_1(d_i) = \begin{cases} 1 & (\text{if } d_i \leq \rho) \\ \rho/d_i & (\text{if } d_i > \rho) \end{cases}$$

The second set of weights $w_2(d_i)$ applies to the parameters in the second moments and is defined as

$$w_2(d_i) = (w_1(d_i))^2/\kappa$$

These are essentially the same Huber-type weighting functions as those for the two-stage robust method. The only difference is that the $d_r$, instead of the $d_s$, formula is used in the weighting functions for the direct robust method. The definition of $\rho$ is also the same as that in the two-stage robust method, but it is now based on a different theoretical chi-square distribution. That is, in the direct robust method, $\rho$ is the critical value that corresponds to the $(1 - \varphi) \times 100$ quantile of the $\chi^2_r = \sqrt{\kappa}r^2$ distribution, with $r = (h - q)$ being the degrees of freedom. Again, $\varphi$ is a tuning parameter and is set to 0.05 by default. You can override this value by using the **ROBPHI=** option. The calculation of the number of “independent factors” $q$ depends on the variants of the direct robust estimation that you choose. With the **ROBUST=RES(E)** option, $q$ is the same as the number of exogenous factors specified in the model. With the **ROBUST=RES(F)** option, the disturbances of the endogenous factors in the model are also treated as “independent factors,” so $q$ is the total number of latent factors specified in the model.

The direct robust method (**ROBUST=RES(E)** or **ROBUST=RES(F)**) employs the IRLS algorithm in model estimation. Let the following expression be a vector of nonredundant first and second moments structured in terms of model parameters $\theta$:

$$m(\theta) = \left( \begin{array}{c} \mu(\theta) \\ \text{vech}(\Sigma(\theta)) \end{array} \right)$$

where vech() extracts the lower-triangular nonredundant elements in $\Sigma(\theta)$. The updating formulas for $\theta$ at the $j+1$ iteration is
\[ \theta^{(j+1)} = \theta^{(j)} + \Delta \theta^{(j)} \]

where \( \theta^{(j)} \) is the parameter values at the \( j \)th iteration and \( \Delta \theta^{(j)} \) is defined by the following formula:

\[
\Delta \theta^{(j)} = (\mathbf{m}^T(\theta^{(j)}) \mathbf{W}(\theta^{(j)}) \mathbf{m}(\theta^{(j)}))^{-1} \mathbf{m}^T(\theta^{(j)}) \mathbf{W}(\theta^{(j)}) \mathbf{g}_j
\]

where \( \mathbf{m}(\theta) = \frac{\partial \mathbf{m}^T}{\partial \theta} \) is the model Jacobian, \( \mathbf{W}(\theta) \) is the (normal theory) weight matrix for the moments (see Yuan and Zhong 2008 for the formula of the weight matrix), and \( \mathbf{g}_j \) is defined as

\[
\mathbf{g}_j = \left( \frac{1}{\sum_{i=1}^{N} w_1(d_i)} \sum_{i=1}^{N} w_1(d_i) x_i - \mu(\theta^{(j)}) \right) - \frac{1}{N} \sum_{i=1}^{N} w_2(d_i) \text{vech}[(x_i - \mu(\theta^{(j)}))(x_i - \mu(\theta^{(j)}))^T] - \Sigma(\theta^{(j)}) \right)
\]

Starting with some reasonable initial estimates for \( \theta \), PROC CALIS iterates the updating formulas until the relative parameter convergence criterion of the IRLS algorithm is satisfied. The default criterion value is 1E–8. This essentially means that the IRLS algorithm converges when \( \Delta \theta \) is sufficiently small. See the XCONV= option for the definition of the relative parameter convergence criterion.

Although the iterative formulas and the IRLS steps for robust estimation have been presented for single-group analysis, they are easily generalizable to multiple-group analysis. For the two-stage robust method, you only need to repeat the robust estimation of the means and covariances for the groups and then apply the obtained robust moments as if they were sample moments for regular maximum likelihood estimation (METHOD=ML). For the direct robust method, you need to expand the dimensions of the model Jacobian \( \mathbf{m}(\theta) \), the weight matrix \( \mathbf{W}(\theta) \), and the vector \( \mathbf{g} \) to include moments from several groups/models. Therefore, the multiple-group formulas are conceptually quite simple but tedious to present. For this reason, they are omitted here.

---

### Relationships among Estimation Criteria

There is always some arbitrariness to classify the estimation methods according to certain mathematical or numerical properties. The discussion in this section is not meant to be a thorough classification of the estimation methods available in PROC CALIS. Rather, classification is done here with the purpose of clarifying the uses of different estimation methods and the theoretical relationships of estimation criteria.

#### Assumption of Multivariate Normality

GLS, ML, and FIML assume multivariate normality of the data, while ULS, WLS, and DWLS do not. Although the ML method with covariance structure analysis alone can also be based on the Wishart distribution of the sample covariance matrix, for convenience GLS, ML, and FIML are usually classified as normal-theory based methods, while ULS, WLS, and DWLS are usually classified as distribution-free methods.

An intuitive or even naive notion is usually that methods without distributional assumptions such as WLS and DWLS are preferred to normal theory methods such as ML and GLS in practical situations where multivariate normality is doubt. This notion might need some qualifications because there are simply more factors to consider in judging the quality of estimation methods in practice. For example, the WLS method might need a very large sample size to enjoy its purported asymptotic properties, while the ML might be robust against the violation of multi-normality assumption under certain circumstances. No recommendations regarding which estimation criterion should be used are attempted here, but you should make your choice based more than the assumption of multivariate normality.
Contribution of the Off-Diagonal Elements to the Estimation of Covariance or Correlation Structures

If only the covariance or correlation structures are considered, the six estimation functions, $F_{ULS}$, $F_{GLS}$, $F_{ML}$, $F_{FIML}$, $F_{WLS}$, and $F_{DWLS}$, belong to the following two groups:

- The functions $F_{ULS}$, $F_{GLS}$, $F_{ML}$, and $F_{FIML}$ take into account all $n^2$ elements of the symmetric residual matrix $S - \Sigma$. This means that the off-diagonal residuals contribute twice to the discrepancy function $F$, as lower and as upper triangle elements.

- The functions $F_{WLS}$ and $F_{DWLS}$ take into account only the $n(n + 1)/2$ lower triangular elements of the symmetric residual matrix $S - \Sigma$. This means that the off-diagonal residuals contribute to the discrepancy function $F$ only once.

The $F_{DWLS}$ function used in PROC CALIS differs from that used by the LISREL 7 program. Formula (1.25) of the LISREL 7 manual (Jöreskog and Sörbom 1985, p. 23) shows that LISREL groups the $F_{DWLS}$ function in the first group by taking into account all $n^2$ elements of the symmetric residual matrix $S - \Sigma$.

- Relationship between DWLS and WLS:
  PROC CALIS: The $F_{DWLS}$ and $F_{WLS}$ discrepancy functions deliver the same results for the special case that the weight matrix $W = W_{ss}$ used by WLS estimation is a diagonal matrix.
  LISREL 7: This is not the case.

- Relationship between DWLS and ULS:
  LISREL 7: The $F_{DWLS}$ and $F_{ULS}$ estimation functions deliver the same results for the special case that the diagonal weight matrix $W = W_{ss}$ used by DWLS estimation is an identity matrix.
  PROC CALIS: To obtain the same results with $F_{DWLS}$ and $F_{ULS}$ estimation, set the diagonal weight matrix $W = W_{ss}$ used in DWLS estimation to:

$$[W_{ss}]_{ik} = \begin{cases} 1, & \text{if } i = k \\ 0.5, & \text{otherwise} \end{cases} \quad (k \leq i)$$

Because the reciprocal elements of the weight matrix are used in the discrepancy function, the off-diagonal residuals are weighted by a factor of 2.

ML and FIML Methods

Both the ML and FIML methods can be derived from the log-likelihood function for multivariate normal data. The preceding section “Estimation Criteria” on page 1463 mentions that $F_{FIML}$ is essentially the same as $-\frac{2L}{N}$, where $L$ is the log-likelihood function for multivariate normal data. For the ML estimation, you can also consider $-\frac{2L}{N}$ as a part of the $F_{ML}$ discrepancy function that contains the information regarding the model parameters (while the rest the $F_{ML}$ function contains some constant terms given the data). That is, with some algebraic manipulations and assuming that there is no missing value in the analysis (so that all $\mu_j$ and $\Sigma_j$ are the same as $\mu$ and $\Sigma$, respectively), it can shown that

$$F_{FIML} = -\frac{2L}{N}$$

$$= \frac{1}{N} \sum_{j=1}^{n} (\ln(|\Sigma|) + (x_j - \mu)\Sigma^{-1}(x_j - \mu) + K)$$

$$= \ln(|\Sigma|) + \text{Tr}(S_{N}\Sigma^{-1}) + (\bar{x} - \mu)\Sigma^{-1}(\bar{x} - \mu) + K$$
Gradient, Hessian, Information Matrix, and Approximate Standard Errors

where $\bar{x}$ is the sample mean and $S_N$ is the biased sample covariance matrix. Compare this FIML function with the ML function shown in the following expression, which shows that both functions are very similar:

$$F_{ML} = \ln(|\Sigma|) + \text{Tr}(S\Sigma^{-1}) + (\bar{x} - \mu)'\Sigma^{-1}(\bar{x} - \mu) - p - \ln(|S|)$$

The two expressions differ only in the constant terms, which are independent of the model parameters, and in the formulas for computing the sample covariance matrix. While the FIML method assumes the biased formula (with $N$ as the divisor, by default) for the sample covariance matrix, the ML method (as implemented in PROC CALIS) uses the unbiased formula (with $N - 1$ as the divisor, by default).

The similarity (or dissimilarity) of the ML and FIML discrepancy functions leads to some useful conclusions here:

- Because the constant terms in the discrepancy functions play no part in parameter estimation (except for shifting the function values), overriding the default ML method with \text{VARDEF=}=N (that is, using $N$ as the divisor in the covariance matrix formula) leads to the same estimation results as that of the FIML method, given that there are no missing values in the analysis.

- Because the FIML function is evaluated at the level of individual observations, it is much more expensive to compute than the ML function. As compared with ML estimation, FIML estimation takes longer and uses more computing resources. Hence, for data without missing values, the ML method should always be chosen over the FIML method.

- The advantage of the FIML method lies solely in its ability to handle data with random missing values. While the FIML method uses the information maximally from each observation, the ML method (as implemented in PROC CALIS) simply throws away any observations with at least one missing value. If it is important to use the information from observations with random missing values, the FIML method should be given consideration over the ML method.

See \texttt{Example 29.15} for an application of the FIML method and \texttt{Example 29.16} for an empirical comparison of the ML and FIML methods. For more examples and details about the FIML method employed by PROC CALIS, see Yung and Zhang (2011); Zhang and Yung (2011).

Gradient, Hessian, Information Matrix, and Approximate Standard Errors

For a single-sample setting with a discrepancy function $F = F(\Theta, \Sigma(\Theta), \mu(\Theta); S, \bar{x})$, the gradient is defined as the first partial derivatives of the discrepancy function with respect to the model parameters $\Theta$:

$$g(\Theta) = \frac{\partial}{\partial \Theta} F(\Theta)$$

The Hessian is defined as the second partial derivatives of the discrepancy function with respect to the model parameters $\Theta$:

$$H(\Theta) = \frac{\partial^2}{\partial \Theta \partial \Theta'} F(\Theta)$$
Suppose that the mean and covariance structures fit perfectly with $\Theta = \Theta_o$ in the population. The information matrix is defined as

$$I(\Theta_o) = \frac{1}{2} E(H(\Theta_o))$$

where the expectation $E(\cdot)$ is taken over the sampling space of $S, \bar{x}$.

The information matrix plays a significant role in statistical theory. Under certain regularity conditions, the inverse of the information matrix $I^{-1}(\Theta_o)$ is the asymptotic covariance matrix for $\sqrt{N}(\hat{\Theta} - \Theta_o)$, where $N$ denotes the sample size and $\hat{\Theta}$ is an estimator.

In practice, $\Theta_o$ is never known and can only be estimated. The information matrix is therefore estimated by the so-called empirical information matrix:

$$I(\hat{\Theta}) = \frac{1}{2} H(\hat{\Theta})$$

which is evaluated at the values of the sample estimates $\hat{\Theta}$. Notice that this empirical information matrix, rather than the unknown $I(\Theta_o)$, is the “information matrix” displayed in PROC CALIS output.

Taking the inverse of the empirical information matrix with sample size adjustment, PROC CALIS approximates the estimated covariance matrix of $\hat{\Theta}$ by:

$$((N - 1)I(\hat{\Theta}))^{-1} = ((N - 1)\frac{1}{2} H(\hat{\Theta}))^{-1} = \frac{2}{N - 1} H^{-1}(\hat{\Theta})$$

Approximate standard errors for $\hat{\Theta}$ can then be computed as the square roots of the diagonal elements of the estimated covariance matrix. The theory about the empirical information matrix, the approximate covariance matrix of the parameter estimates, and the approximate standard errors applies to all but the ULS and DWLS estimation methods. Standard errors are therefore not computed with the ULS and DWLS estimation methods.

If a given Hessian or information matrix is singular, PROC CALIS offers two ways to compute a generalized inverse of the matrix and, therefore, two ways to compute approximate standard errors of implicitly constrained parameter estimates, $t$ values, and modification indices. Depending on the G4= specification, either a Moore-Penrose inverse or a G2 inverse is computed. The expensive Moore-Penrose inverse computes an estimate of the null space by using an eigenvalue decomposition. The cheaper G2 inverse is produced by sweeping the linearly independent rows and columns and zeroing out the dependent ones.

**Multiple-Group Extensions**

In the section “Multiple-Group Discrepancy Function” on page 1469, the overall discrepancy function for multiple-group analysis is defined. The same notation is applied here. To begin with, the overall discrepancy function $F(\Theta)$ is expressed as a weighted sum of individual discrepancy functions $F_i$’s for the groups as follows:

$$F(\Theta) = \sum_{i=1}^{k} t_i F_i(\Theta)$$

where

$$t_i = \frac{N_i - 1}{N - k}$$
is the weight for group $i$, 

$$N = \sum_{i=1}^{k} N_i$$

is the total sample size, and $N_i$ is the sample size for group $i$.

The gradient $g(\Theta)$ and the Hessian $H(\Theta)$ are now defined as weighted sum of individual functions. That is,

$$g(\Theta) = \sum_{i=1}^{k} t_i g_i(\Theta) = \sum_{i=1}^{k} t_i \frac{\partial}{\partial \Theta} F_i(\Theta)$$

and

$$H(\Theta) = \sum_{i=1}^{k} t_i H_i(\Theta) = \sum_{i=1}^{k} t_i \frac{\partial^2}{\partial \Theta \partial \Theta'} F_i(\Theta)$$

Suppose that the mean and covariance structures fit perfectly with $\Theta = \Theta_o$ in the population. If each $t_i$ converges to a fixed constant $\tau_i$ ($\tau_i > 0$) with increasing total sample size, the information matrix can be written as:

$$I(\Theta_o) = \frac{1}{2} \sum_{i=1}^{k} \tau_i E(H_i(\Theta_o))$$

To approximate this information matrix, an empirical counterpart is used:

$$I(\hat{\Theta}) = \frac{1}{2} \sum_{i=1}^{k} t_i H_i(\hat{\Theta})$$

which is evaluated at the values of the sample estimates $\hat{\Theta}$. Again, this empirical information matrix, rather than the unknown $I(\Theta_o)$, is the “information matrix” output in PROC CALIS results.

Taking the inverse of the empirical information matrix with sample size adjustment, PROC CALIS approximates the estimated covariance matrix of $\hat{\Theta}$ in multiple-group analysis by:

$$((N-k)I(\hat{\Theta}))^{-1} = ((N-k) \frac{1}{2} H(\hat{\Theta}))^{-1} = \frac{2}{N-k} \sum_{i=1}^{k} t_i H_i^{-1}(\hat{\Theta})$$

Approximate standard errors for $\hat{\Theta}$ can then be computed as the square roots of the diagonal elements of the estimated covariance matrix. Again, for ULS and DWLS estimation, the theory does not apply and so there are no standard errors computed in these cases.

**Testing Rank Deficiency in the Approximate Covariance Matrix for Parameter Estimates**

When computing the approximate covariance matrix and hence the standard errors for the parameter estimates, inversion of the scaled information matrix or Hessian matrix is involved. The numerical condition of the information matrix can be very poor in many practical applications, especially for the analysis of unscaled covariance data. The following four-step strategy is used for the inversion of the information matrix.
Chapter 29: The CALIS Procedure

1. The inversion (usually of a normalized matrix $D^{-1}HD^{-1}$) is tried using a modified form of the Bunch and Kaufman (1977) algorithm, which allows the specification of a different singularity criterion for each pivot. The following three criteria for the detection of rank loss in the information matrix are used to specify thresholds:

   - **ASING** specifies absolute singularity.
   - **MSING** specifies relative singularity depending on the whole matrix norm.
   - **VSING** specifies relative singularity depending on the column matrix norm.

   If no rank loss is detected, the inverse of the information matrix is used for the covariance matrix of parameter estimates, and the next two steps are skipped.

2. The linear dependencies among the parameter subsets are displayed based on the singularity criteria.

3. If the number of parameters $t$ is smaller than the value specified by the G4= option (the default value is 60), the Moore-Penrose inverse is computed based on the eigenvalue decomposition of the information matrix. If you do not specify the NOPRINT option, the distribution of eigenvalues is displayed, and those eigenvalues that are set to zero in the Moore-Penrose inverse are indicated. You should inspect this eigenvalue distribution carefully.

4. If PROC CALIS did not set the right subset of eigenvalues to zero, you can specify the COVSING= option to set a larger or smaller subset of eigenvalues to zero in a further run of PROC CALIS.

---

**Counting the Degrees of Freedom**

When fitting covariance and mean structure models, the population moments are hypothesized to be functions of model parameters $\Theta$. The population moments refer to the first-order moments (means) and the second-order central moments (variances of and covariances among the variables). Usually, the number of nonredundant population moments is greater than the number of model parameters for a structural model. The difference between the two is the degrees of freedom ($df$) of your model.

Formally, define a multiple-group situation where you have $k$ independent groups in your model. The set of variables in each group might be different so that you have $p_1, p_2, \ldots, p_k$ manifest or observed variables for the $k$ groups. It is assumed that the primary interest is to study the covariance structures. The inclusion of mean structures is optional for each of these groups. Define $\delta_1, \delta_2, \ldots, \delta_k$ as zero-one indicators of the mean structures for the groups. If $\delta_i$ takes the value of one, it means that the mean structures of group $i$ is modeled. The total number of nonredundant elements in the moment matrices is thus computed by:

\[
q = \sum_{i=1}^{k} \left( p_i (p_i + 1)/2 + \delta_i p_i \right)
\]

The first term in the summation represents the number of lower triangular elements in the covariance or correlation matrix, while the second term represents the number of elements in the mean matrix. Let $t$ be the total number of independent parameters in the model. The degrees of freedom is:

\[
df = q - (t - c)
\]

where $c$ represents the number of linear equality constraints imposed on the independent parameters in the model. In effect, the $(t - c)$ expression means that each nonredundant linear equality constraint reduces one independent parameter.
Counting the Number of Independent Parameters

To count the number of independent parameters in the model, first you have to distinguish them from the dependent parameters. Dependent parameters are expressed as functions of other parameters in the SAS programming statements. That is, a parameter is dependent if it appears at the left-hand side of the equal sign in a SAS programming statement.

A parameter is independent if it is not dependent. An independent parameter can be specified in the main or subsidiary model specification statements or the PARAMETERS statement, or it is generated automatically by PROC CALIS as additional parameters. Quite intuitively, all independent parameter specified in the main or subsidiary model specification statements are independent parameters in the model. All automatic parameters added by PROC CALIS are also independent parameters in the model.

Intentionally or not, some independent parameters specified in the PARMS statement might not be counted as independent parameters in the model. Independent parameters in the PARMS statement belong in the model only when they are used to define at least one dependent parameter specified in the main or subsidiary model specification statements. This restriction eliminates the counting of superfluous independent parameters which have no bearing of model specification.

Note that when counting the number of independent parameters, you are counting the number of distinct independent parameter names but not the number of distinct parameter locations for independent parameters. For example, consider the following statement for defining the error variances in a LINEQS model:

\[
\text{variance} \quad \text{E1-E3} = \text{vare1} \text{ vare2} \text{ vare3};
\]

You define three variance parameter locations with three independent parameters \text{vare1}, \text{vare2}, and \text{vare3}. However, in the following specification:

\[
\text{variance} \quad \text{E1-E3} = \text{vare} \text{ vare} \text{ vare};
\]

you still have three variance parameter locations to define, but the number of independent parameter is only one, which is the parameter named \text{vare}.

Counting the Number of Linear Equality Constraints

The linear equality constraints refer to those specified in the BOUNDS or LINCON statement. For example, consider the following specification:

\[
\text{bounds} \quad 3 \leq \text{parm01} \leq 3;
\]
\[
\text{lincon} \quad 3 \times \text{parm02} + 2 \times \text{parm03} = 12;
\]

In the BOUNDS statement, \text{parm01} is constrained to a fixed number 3, and in the LINCON statement, \text{parm02} and \text{parm03} are constrained linearly. In effect, these two statements reduce two independent parameters from the model. In the degrees of freedom formula, the value of \(c\) is 2 for this example.

Adjustment of Degrees of Freedom

In some cases, computing degrees of freedom for model fit is not so straightforward. Two important cases are considered in the following.

The first case is when you set linear inequality or boundary constraints in your model, and these inequality or boundary constraints become “active” in your final solution. For example, you might have set inequality boundary and linear constraints as:
Chapter 29: The CALIS Procedure

```latex
bounds  0 <= var01;
lincon  3 * betal + 2 * beta2 >= 7;
```

The optimal solution occurs at the boundary point so that you observe in the final solution the following two equalities:

```latex
var01 = 0,
3 * betal + 2 * beta2 = 7
```

These two active constraints reduce the number of independent parameters of your original model. As a result, PROC CALIS will automatically increase the degrees of freedom by the number of active linear constraints. Adjusting degrees of freedom not only affects the significance of the model fit chi-square statistic, but it also affects the computation of many fit statistics and indices. See Dijkstra (1992) for a discussion of the validity of statistical inferences with active boundary constraints.

Automatically adjusting \( df \) in such a situation might not be totally justified in all cases. Statistical estimation is subject to sampling fluctuation. Active constraints might not occur when fitting the same model in new samples. If the researcher believes that those linear inequality and boundary constraints have a small chance of becoming active in repeated sampling, it might be more suitable to turn off the automatic adjustment by using the NOADJDF option in the PROC CALIS statement.

Another case where you need to pay attention to the computation of degrees of freedom is when you fit correlation models. The degrees-of-freedom calculation in PROC CALIS applies mainly to models with covariance structures with or without mean structures. When you model correlation structures, the degrees of freedom calculation in PROC CALIS is a straightforward generalization of the covariance structures. It does not take the fixed ones at the diagonal elements of the sample correlation matrix into account. Some might argue that with correlation structures, the degrees of freedom should be reduced by the total number of diagonal elements in the correlation matrices in the model. While PROC CALIS does not do this automatically, you can use the DFREDUCE= option to specify the adjustment, where \( i \) can be any positive or negative integer. The \( df \) value is reduced by the DFREDUCE= value.

A Different Type of Degrees of Freedom

The degrees of freedom for model fitting has to be distinguished from another type of degrees of freedom. In a regression problem, the number of degrees of freedom for the error variance estimate is the number of observations in the data set minus the number of parameters. The NOBS=, DFR= (RDF=), and DFE= (EDF=) options refer to degrees of freedom in this sense. However, these values are not related to the degrees of freedom for the model fit statistic. The NOBS=, DFR=, and DFE= options should be used in PROC CALIS to specify the effective number of observations in the input data set only.
Assessment of Fit

In PROC CALIS, there are three main tools for assessing model fit:

- residuals for the fitted means or covariances
- overall model fit indices
- squared multiple correlations and determination coefficients

This section contains a collection of formulas for these assessment tools. The following notation is used:

- \( N \) for the total sample size
- \( k \) for the total number of independent groups in analysis
- \( p \) for the number of manifest variables
- \( t \) for the number of parameters to estimate
- \( \Theta \) for the \( t \)-vector of parameters, \( \hat{\Theta} \) for the estimated parameters
- \( S = (s_{ij}) \) for the \( p \times p \) input covariance or correlation matrix
- \( \bar{x} = (\bar{x}_i) \) for the \( p \)-vector of sample means
- \( \hat{\Sigma} = \Sigma(\hat{\Theta}) = (\hat{s}_{ij}) \) for the predicted covariance or correlation matrix
- \( \hat{\mu} = (\hat{\mu}_i) \) for the predicted mean vector
- \( \delta \) for indicating the modeling of the mean structures
- \( W \) for the weight matrix
- \( f_{\min} \) for the minimized function value of the fitted model
- \( d_{\min} \) for the degrees of freedom of the fitted model

In multiple-group analyses, subscripts are used to distinguish independent groups or samples. For example, \( N_1, N_2, \ldots, N_r, \ldots, N_k \) denote the sample sizes for \( k \) groups. Similarly, notation such as \( p_r, S_r, \bar{x}_r, \hat{\Sigma}_r, \hat{\mu}_r, \delta_r, \) and \( W_r \) is used for multiple-group situations.
Residuals in the Moment Matrices

Residuals indicate how well each entry or element in the mean or covariance matrix is fitted. Large residuals indicate bad fit.

PROC CALIS computes four types of residuals and writes them to the OUTSTAT= data set when requested.

- **raw residuals**
  \[ s_{ij} - \hat{s}_{ij}, \quad \bar{x}_i - \hat{\mu}_i \]
  for the covariance and mean residuals, respectively. The raw residuals are displayed whenever the PALL, PRINT, or RESIDUAL option is specified.

- **variance standardized residuals**
  \[ \frac{s_{ij} - \hat{s}_{ij}}{\sqrt{s_{ii}s_{jj}}}, \quad \frac{\bar{x}_i - \hat{\mu}_i}{\sqrt{s_{ii}}} \]
  for the covariance and mean residuals, respectively. The variance standardized residuals are displayed when you specify one of the following:
  - the PALL, PRINT, or RESIDUAL option and METHOD=NONE, METHOD=ULS, or METHOD=DWLS
  - RESIDUAL=VARSTAND

The variance standardized residuals are equal to those computed by the EQS 3 program (Bentler 1995).

- **asymptotically standardized residuals**
  \[ \frac{s_{ij} - \hat{s}_{ij}}{\sqrt{v_{ij,ij}}}, \quad \frac{\bar{x}_i - \hat{\mu}_i}{\sqrt{u_{ii}}} \]
  for the covariance and mean residuals, respectively; with
  \[ v_{ij,ij} = (\hat{\Gamma}_1 - \hat{J}_1 \hat{\text{Cov}}(\hat{\Theta}) \hat{J}_1')_{ij,ij} \]
  \[ u_{ii} = (\hat{\Gamma}_2 - \hat{J}_2 \hat{\text{Cov}}(\hat{\Theta}) \hat{J}_2')_{ii} \]
  where \( \hat{\Gamma}_1 \) is the \( p^2 \times p^2 \) estimated asymptotic covariance matrix of sample covariances, \( \hat{\Gamma}_2 \) is the \( p \times p \) estimated asymptotic covariance matrix of sample means, \( \hat{J}_1 \) is the \( p^2 \times t \) Jacobian matrix \( d \Sigma / d \Theta \), \( \hat{J}_2 \) is the \( p \times t \) Jacobian matrix \( d \mu / d \Theta \), and \( \hat{\text{Cov}}(\hat{\Theta}) \) is the \( t \times t \) estimated covariance matrix of parameter estimates, all evaluated at the sample moments and estimated parameter values. See the next section for the definitions of \( \hat{\Gamma}_1 \) and \( \hat{\Gamma}_2 \). Asymptotically standardized residuals are displayed when one of the following conditions is met:
  - The PALL, the PRINT, or the RESIDUAL option is specified, and METHOD=ML, METHOD=GLS, or METHOD=WLS, and the expensive information and Jacobian matrices are computed for some other reason.
  - RESIDUAL=ASYSTAND is specified.
Assessment of Fit

The asymptotically standardized residuals are equal to those computed by the LISREL 7 program (Jöreskog and Sörbom 1988) except for the denominator in the definition of matrix \( \hat{\Gamma}_1 \).

- **normalized residuals**

\[
\frac{s_{ij} - \hat{\theta}_{ij}}{\sqrt{(\hat{\Gamma}_1)_{i,j,j}}} \quad \frac{\bar{x}_i - \hat{\mu}_i}{\sqrt{(\hat{\Gamma}_2)_{i,i}}}
\]

for the covariance and mean residuals, respectively; with \( \hat{\Gamma}_1 \) as the \( p^2 \times p^2 \) estimated asymptotic covariance matrix of sample covariances; and \( \hat{\Gamma}_2 \) as the \( p \times p \) estimated asymptotic covariance matrix of sample means.

Diagonal elements of \( \hat{\Gamma}_1 \) and \( \hat{\Gamma}_2 \) are defined for the following methods:

- **GLS:** \((\hat{\Gamma}_1)_{i,j,j} = \frac{1}{(N-1)} (s_{ii}s_{jj} + s_{ij}^2) \) and \((\hat{\Gamma}_2)_{ii} = \frac{1}{(N-1)} s_{ii} \)
- **ML:** \((\hat{\Gamma}_1)_{i,j,j} = \frac{1}{(N-1)} (\hat{\theta}_{ii}\hat{\sigma}_{jj} + \hat{\sigma}_{ij}^2) \) and \((\hat{\Gamma}_2)_{ii} = \frac{1}{(N-1)} \hat{\theta}_{ii} \)
- **WLS:** \((\hat{\Gamma}_1)_{i,j,j} = \frac{1}{(N-1)} W_{i,j,j} \) and \((\hat{\Gamma}_2)_{ii} = \frac{1}{(N-1)} s_{ii} \)

where \( W \) in the **WLS** method is the weight matrix for the second-order moments.

Normalized residuals are displayed when one of the following conditions is met:

- The **PALL**, **PRINT**, or **RESIDUAL** option is specified, and **METHOD=ML**, **METHOD=GLS**, or **METHOD=WLS**, and the expensive information and Jacobian matrices are not computed for some other reasons.
- **RESIDUAL=NORM** is specified.

The normalized residuals are equal to those computed by the LISREL VI program (Jöreskog and Sörbom 1985) except for the definition of the denominator in computing matrix \( \hat{\Gamma}_1 \).

For estimation methods that are not “best” generalized least squares estimators (Browne 1982, 1984), such as **METHOD=NONE**, **METHOD=ULS**, or **METHOD=DWLS**, the assumption of an asymptotic covariance matrix \( \Gamma_1 \) of sample covariances does not seem to be appropriate. In this case, the normalized residuals should be replaced by the more relaxed variance standardized residuals. Computation of asymptotically standardized residuals requires computing the Jacobian and information matrices. This is computationally very expensive and is done only if the Jacobian matrix has to be computed for some other reasons—that is, if at least one of the following items is true:

- The default, **PRINT**, or **PALL** displayed output is requested, and neither the **NOMOD** nor **NOSTDERR** option is specified.
- Either the **MODIFICATION** (included in **PALL**), **PCOVES**, or **STDERR** (included in default, **PRINT**, and **PALL** output) option is requested or **RESIDUAL=ASYSTAND** is specified.
- The **LEVMAR** or **NEWRAP** optimization technique is used.
- An **OUTMODEL=** data set is specified without using the **NOSTDERR** option.
- An **OUTEST=** data set is specified without using the **NOSTDERR** option.
Since normalized residuals use an overestimate of the asymptotic covariance matrix of residuals (the diagonals of $\Gamma_1$ and $\Gamma_2$), the normalized residuals cannot be greater than the asymptotically standardized residuals (which use the diagonal of the form $\Gamma - J\text{Cov}(\Theta)J'$).

Together with the residual matrices, the values of the average residual, the average off-diagonal residual, and the rank order of the largest values are displayed. The distributions of the normalized and standardized residuals are displayed also.

**Overall Model Fit Indices**

Instead of assessing the model fit by looking at a number of residuals of the fitted moments, an overall model fit index measures model fit by a single number. Although an overall model fit index is precise and easy to use, there are indeed many choices of overall fit indices. Unfortunately, researchers do not always have a consensus on the best set of indices to use in all occasions.

PROC CALIS produces a large number of overall model fit indices in the fit summary table. If you prefer to display only a subset of these fit indices, you can use the ONLIST(ONLY)= option of the FITINDEX statement to customize the fit summary table.

Fit indices are classified into three classes in the fit summary table of PROC CALIS:

- absolute or standalone Indices
- parsimony indices
- incremental indices

**Absolute or Standalone Indices**

These indices are constructed so that they measure model fit without comparing with a baseline model and without taking the model complexity into account. They measure the absolute fit of the model.

- **fit function or discrepancy function**
  The fit function or discrepancy function $F$ is minimized during the optimization. See the section "Estimation Criteria" on page 1463 for definitions of various discrepancy functions available in PROC CALIS. For a multiple-group analysis, the fit function can be written as a weighted average of discrepancy functions for $k$ independent groups as:

$$F = \sum_{r=1}^{k} a_r F_r$$

where $a_r = \frac{(N_r - 1)}{(N - k)}$ and $F_r$ are the group weight and the discrepancy function for the $r$th group, respectively. Notice that although the groups are assumed to be independent in the model, in general $F_r$’s are not independent when $F$ is being minimized. The reason is that $F_r$’s might have shared parameters in $\Theta$ during estimation.

The minimized function value of $F$ will be denoted as $f_{\text{min}}$, which is always positive, with small values indicating good fit.
• **χ² test statistic**
  For the ML, GLS, and the WLS estimation, the overall χ² measure for testing model fit is:

  \[ χ² = (N - k) \times f_{\text{min}} \]

  where \( f_{\text{min}} \) is the function value at the minimum, \( N \) is the total sample size, and \( k \) is the number of independent groups. The associated degrees of freedom is denoted by \( d_{\text{min}} \).

  For the ML estimation, this gives the likelihood ratio test statistic of the specified structural model in the null hypothesis against an unconstrained saturated model in the alternative hypothesis. The χ² test is valid only if the observations are independent and identically distributed, the analysis is based on the unstandardized sample covariance matrix \( S \), and the sample size \( N \) is sufficiently large (Browne 1982; Bollen 1989b; Jöreskog and Sörbom 1985). For ML and GLS estimates, the variables must also have an approximately multivariate normal distribution.

  In the output fit summary table of PROC CALIS, the notation “Prob > Chi-Square” means “the probability of obtaining a greater χ² value than the observed value under the null hypothesis.” This probability is also known as the \( p \)-value of the chi-square test statistic.

• **adjusted χ² value** (Browne 1982)
  If the variables are \( p \)-variate elliptic rather than normal and have significant amounts of multivariate kurtosis (leptokurtic or platykurtic), the χ² value can be adjusted to:

  \[ χ_{\text{ell}}² = \frac{χ²}{η²} \]

  where \( η² \) is the multivariate relative kurtosis coefficient.

• **Z-test** (Wilson and Hilferty 1931)
  The Z-test of Wilson and Hilferty assumes a \( p \)-variate normal distribution:

  \[ Z = \sqrt{\frac{χ²}{d}} - \left(1 - \frac{2}{9d} \right) \sqrt{\frac{2}{9d}} \]

  where \( d \) is the degrees of freedom of the model. See McArdle (1988) and Bishop, Fienberg, and Holland (1975, p. 527) for an application of the Z-test.

• **critical N index** (Hoelter 1983)
  The critical N (Hoelter 1983) is defined as

  \[ CN = \text{int}(\frac{χ_{\text{crit}}²}{f_{\text{min}}}) \]

  where \( χ_{\text{crit}}² \) is the critical chi-square value for the given \( d \) degrees of freedom and probability \( α = 0.05 \), and int() takes the integer part of the expression. See Bollen (1989b, p. 277). Conceptually, the CN value is the largest number of observations that could still make the chi-square model fit statistic insignificant if it were to apply to the actual sample fit function value \( f_{\text{min}} \). Hoelter (1983) suggests that CN should be at least 200; however, Bollen (1989b) notes that the CN value might lead to an overly pessimistic assessment of fit for small samples.

  Note that when you have a perfect model fit for your data (that is, \( f_{\text{min}} = 0 \)) or a zero degree of freedom for your model (that is, \( d = 0 \)), CN is not computable.
Chapter 29: The CALIS Procedure

• root mean square residual (RMR)
  For a single-group analysis, the RMR is the root of the mean squared residuals:

  \[
  RMR = \sqrt{\frac{1}{b} \left( \sum_{i} \sum_{j} (s_{ij} - \hat{s}_{ij})^2 + \delta \sum_{i} (\bar{x}_i - \hat{\mu}_i)^2 \right)}
  \]

  where

  \[
  b = \frac{p(p+1+2\delta)}{2}
  \]

  is the number of distinct elements in the covariance matrix and in the mean vector (if modeled).

  For multiple-group analysis, PROC CALIS uses the following formula for the overall RMR:

  \[
  \text{overall RMR} = \sqrt{\frac{1}{\sum_{r=1}^{k} \frac{w_r}{\sum_{r=1}^{k} w_r}} \left( \sum_{i} \sum_{j} (s_{ij} - \hat{s}_{ij})^2 + \delta \sum_{i} (\bar{x}_i - \hat{\mu}_i)^2 \right)}
  \]

  where

  \[
  w_r = \frac{N_r - 1}{N - k} b_r
  \]

  is the weight for the squared residuals of the \( r \)th group. Hence, the weight \( w_r \) is the product of group size weight \( \frac{N_r - 1}{N - k} \) and the number of distinct moments \( b_r \) in the \( r \)th group.

• standardized root mean square residual (SRMR)

  For a single-group analysis, the SRMR is the root of the mean of the standardized squared residuals:

  \[
  \text{SRMR} = \sqrt{\frac{1}{b} \left( \sum_{i} \sum_{j} \frac{(s_{ij} - \hat{s}_{ij})^2}{s_{ii}^2 s_{jj}} + \delta \sum_{i} \frac{(\bar{x}_i - \hat{\mu}_i)^2}{s_{ii}} \right)}
  \]

  where \( b \) is the number of distinct elements in the covariance matrix and in the mean vector (if modeled). The formula for \( b \) is defined exactly the same way as it appears in the formula for RMR.

  Similar to the calculation of the overall RMR, an overall measure of SRMR in a multiple-group analysis is a weighted average of the standardized squared residuals of the groups. That is,

  \[
  \text{overall SRMR} = \sqrt{\frac{1}{\sum_{r=1}^{k} \frac{w_r}{\sum_{r=1}^{k} w_r}} \left( \sum_{i} \sum_{j} \frac{(s_{ij} - \hat{s}_{ij})^2}{s_{ii}^2 s_{jj}} + \delta \sum_{i} \frac{(\bar{x}_i - \hat{\mu}_i)^2}{s_{ii}} \right)}
  \]

  where \( w_r \) is the weight for the squared residuals of the \( r \)th group. The formula for \( w_r \) is defined exactly the same way as it appears in the formula for SRMR.

• goodness-of-fit index (GFI)

  For a single-group analysis, the goodness-of-fit index for the ULS, GLS, and ML estimation methods is:

  \[
  \text{GFI} = 1 - \frac{\text{Tr}((W^{-1}(S - \hat{\Sigma}))^2) + \delta (\bar{x} - \hat{\mu})^t W^{-1} (\bar{x} - \hat{\mu})}{\text{Tr}((W^{-1}S)^2) + \delta \bar{x}^t W^{-1} \bar{x}}
  \]
with $W = I$ for ULS, $W = S$ for GLS, and $W = \hat{\Sigma}$. For WLS and DWLS estimation,

$$GFI = 1 - \frac{(u - \hat{\eta})'W^{-1}(u - \hat{\eta})}{u'W^{-1}u}$$

where $u$ is the vector of observed moments and $\hat{\eta}$ is the vector of fitted moments. When the mean structures are modeled, vectors $u$ and $\hat{\eta}$ contains all the nonredundant elements $\text{vecs}(S)$ in the covariance matrix and all the means. That is,

$$u = (\text{vecs}'(S), \bar{x})', \quad \hat{\eta} = (\text{vecs}'(\hat{\Sigma}), \hat{\mu}')'$$

and the symmetric weight matrix $W$ is of dimension $p \times (p + 3)/2$. When the mean structures are not modeled, vectors $u$ and $\hat{\eta}$ contains all the nonredundant elements $\text{vecs}(S)$ in the covariance matrix only. That is,

$$u = \text{vecs}(S), \quad \hat{\eta} = \text{vecs}(\hat{\Sigma})$$

and the symmetric weight matrix $W$ is of dimension $p \times (p + 1)/2$. In addition, for the DWLS estimation, $W$ is a diagonal matrix.

For a constant weight matrix $W$, the goodness-of-fit index is $1$ minus the ratio of the minimum function value and the function value before any model has been fitted. The GFI should be between $0$ and $1$. The data probably do not fit the model if the GFI is negative or much greater than $1$.

For a multiple-group analysis, individual $GFI_r$’s are computed for groups. The overall measure is a weighted average of individual $GFI_r$’s, using weight $a_r = \frac{N_r - 1}{N-k}$. That is,

$$\text{overall GFI} = \sum_{r=1}^{k} a_r GFI_r$$

**Parsimony Indices**

These indices are constructed so that the model complexity is taken into account when assessing model fit. In general, models with more parameters (fewer degrees of freedom) are penalized.

- **adjusted goodness-of-fit index (AGFI)**
  The AGFI is the GFI adjusted for the degrees of freedom $d$ of the model,

$$\text{AGFI} = 1 - \frac{c}{d} (1 - \text{GFI})$$

where

$$c = \sum_{r=1}^{k} p_k (p_k + 1 + 2\delta_k)$$

computes the total number of elements in the covariance matrices and mean vectors for modeling. For single-group analyses, the AGFI corresponds to the GFI in replacing the total sum of squares by the mean sum of squares.
CAUTION:

- Large \( p \) and small \( d \) can result in a negative AGFI. For example, GFI=0.90, \( p=19 \), and \( d=2 \) result in an AGFI of \(-8.5\).
- AGFI is not defined for a saturated model, due to division by \( d = 0 \).
- AGFI is not sensitive to losses in \( d \).

The AGFI should be between 0 and 1. The data probably do not fit the model if the AGFI is negative or much greater than 1. For more information, see Mulaik et al. (1989).

- **parsimonious goodness-of-fit index (PGFI)**
  The PGFI (Mulaik et al. 1989) is a modification of the GFI that takes the parsimony of the model into account:
  \[
  \text{PGFI} = \frac{d_{\text{min}}}{d_0} \text{GFI}
  \]
  where \( d_{\text{min}} \) is the model degrees of freedom and \( d_0 \) is the degrees of freedom for the independence model. See the section “Incremental Indices” on page 1490 for the definition of independence model. The PGFI uses the same parsimonious factor as the parsimonious normed Bentler-Bonett index (James, Mulaik, and Brett 1982).

- **RMSEA index (Steiger and Lind 1980; Steiger 1998)**
  The root mean square error of approximation (RMSEA) coefficient is:
  \[
  \epsilon = \sqrt{k} \max \left( \frac{f_{\text{min}}}{d_{\text{min}}} - \frac{1}{(N - k)}, 0 \right)
  \]
  The lower and upper limits of the \((1 - \alpha)\%\)-confidence interval are computed using the cumulative distribution function of the noncentral chi-square distribution \( \Phi(x|\lambda, d) \). With \( x = (N - k) f_{\text{min}} \), \( \lambda_L \) satisfying \( \Phi(x|\lambda_L, d_{\text{min}}) = 1 - \frac{\alpha}{2} \), and \( \lambda_U \) satisfying \( \Phi(x|\lambda_U, d_{\text{min}}) = \frac{\alpha}{2} \):
  \[
  (\epsilon_{\lambda_L}, \epsilon_{\lambda_U}) = (\sqrt{k} \sqrt{\frac{\lambda_L}{(N - k)d_{\text{min}}}}, \sqrt{k} \sqrt{\frac{\lambda_U}{(N - k)d_{\text{min}}}})
  \]
  See Browne and Du Toit (1992) for more details. The size of the confidence interval can be set by the option ALPHARMS=\( \alpha \), \( 0 \leq \alpha \leq 1 \). The default is \( \alpha = 0.1 \), which corresponds to the 90% confidence interval for the RMSEA.

- **probability for test of close fit (Browne and Cudeck 1993)**
  The traditional exact \( \chi^2 \) test hypothesis \( H_0: \epsilon = 0 \) is replaced by the null hypothesis of close fit \( H_0: \epsilon \leq 0.05 \) and the exceedance probability \( P \) is computed as:
  \[
  P = 1 - \Phi(x|\lambda^*, d_{\text{min}})
  \]
  where \( x = (N - k) f_{\text{min}} \) and \( \lambda^* = 0.05^2(N - k)d_{\text{min}}/k \). The null hypothesis of close fit is rejected if \( P \) is smaller than a prespecified level (for example, \( P < 0.05 \)).
ECVI: expected cross validation index (Browne and Cudeck 1993)
The following formulas for ECVI are limited to the case of single-sample analysis without mean structures and with either the GLS, ML, or WLS estimation method. For other cases, ECVI is not defined in PROC CALIS. For GLS and WLS, the estimator $c$ of the ECVI is linearly related to AIC, Akaike's Information Criterion (Akaike 1974, 1987):

$$c = f_{min} + \frac{2t}{N-1}$$

For ML estimation, $c_{ML}$ is used:

$$c_{ML} = f_{min} + \frac{2t}{N-p-2}$$

For GLS and WLS, the confidence interval $(c_L; c_U)$ for ECVI is computed using the cumulative distribution function $\Phi(x|\lambda, d_{min})$ of the noncentral chi-square distribution,

$$(c_L; c_U) = \left( (N-1) \frac{\lambda_L + p(p+1)/2 + t}{(N-1)} ; (N-1) \frac{\lambda_U + p(p+1)/2 + t}{(N-1)} \right)$$

with $x = (N-1)f_{min}$, $\Phi(x|\lambda_U, d_{min}) = \frac{\alpha}{2}$, and $\Phi(x|\lambda_L, d_{min}) = 1 - \frac{\alpha}{2}$. For ML, the confidence interval $(c_{L*}; c_{U*})$ for ECVI is:

$$(c_{L*}; c_{U*}) = \left( (N-p-2) \frac{\lambda_{L*} + p(p+1)/2 + t}{N-p-2} ; (N-p-2) \frac{\lambda_{U*} + p(p+1)/2 + t}{N-p-2} \right)$$

where $x = (N-p-2)f_{min}$, $\Phi(x|\lambda_{U*}, d_{min}) = \frac{\alpha}{2}$ and $\Phi(x|\lambda_{L*}, d_{min}) = 1 - \frac{\alpha}{2}$. See Browne and Cudeck (1993). The size of the confidence interval can be set by the option $\text{ALPHAECV}=\alpha$, $0 \leq \alpha \leq 1$. The default is $\alpha = 0.1$, which corresponds to the 90% confidence interval for the ECVI.

Akaike's information criterion (AIC) (Akaike 1974, 1987)
This is a criterion for selecting the best model among a number of candidate models. The model that yields the smallest value of AIC is considered the best.

$$\text{AIC} = h + 2t$$

where $h$ is the $-2$ times the likelihood function value for the FIML method or the $\chi^2$ value for other estimation methods.

Consistent Akaike's information criterion (CAIC) (Bozdogan 1987)
This is another criterion, similar to AIC, for selecting the best model among alternatives. The model that yields the smallest value of CAIC is considered the best. CAIC is preferred by some people to AIC or the $\chi^2$ test.

$$\text{CAIC} = h + (\ln(N) + 1)t$$

where $h$ is the $-2$ times the likelihood function value for the FIML method or the $\chi^2$ value for other estimation methods. Notice that $N$ includes the number of incomplete observations for the FIML method while it includes only the complete observations for other estimation methods.
• **Schwarz’s Bayesian criterion (SBC)** (Schwarz 1978; Sclove 1987)
  
  This is another criterion, similar to AIC, for selecting the best model. The model that yields the smallest value of SBC is considered the best. SBC is preferred by some people to AIC or the $\chi^2$ test.

  \[
  \text{SBC} = h + \ln(N)t
  \]

  where $h$ is the $-2$ times the likelihood function value for the FIML method or the $\chi^2$ value for other estimation methods. Notice that $N$ includes the number of incomplete observations for the FIML method while it includes only the complete observations for other estimation methods.

  • **McDonald’s measure of centrality** (McDonald and Marsh 1988)

  \[
  \text{CENT} = \exp\left(-\frac{(\chi^2 - d_{\text{min}})}{2N}\right)
  \]

  **Incremental Indices**

  These indices are constructed so that the model fit is assessed through the comparison with a baseline model. The baseline model is usually the independence model where all covariances among manifest variables are assumed to be zeros. The only parameters in the independence model are the diagonals of covariance matrix. If modeled, the mean structures are saturated in the independence model. For multiple-group analysis, the overall independence model consists of component independence models for each group.

  In the following, let $f_0$ and $d_0$ denote the minimized discrepancy function value and the associated degrees of freedom, respectively, for the independence model; and $f_{\text{min}}$ and $d_{\text{min}}$ denote the minimized discrepancy function value and the associated degrees of freedom, respectively, for the model being fitted in the null hypothesis.

  • **Bentler comparative fit index** (Bentler 1995)

  \[
  \text{CFI} = 1 - \frac{\max((N-k)f_{\text{min}} - d_{\text{min}}, 0)}{\max((N-k)f_{\text{min}} - d_{\text{min}}, \max((N-k)f_0 - d_0, 0))}
  \]

  • **Bentler-Bonett normed fit index (NFI)** (Bentler and Bonett 1980)

  \[
  \Delta = \frac{f_0 - f_{\text{min}}}{f_0}
  \]

  Mulaik et al. (1989) recommend the parsimonious weighted form called parsimonious normed fit index (PNFI) (James, Mulaik, and Brett 1982).

  • **Bentler-Bonett nonnormed coefficient** (Bentler and Bonett 1980)

  \[
  \rho = \frac{f_0/d_0 - f_{\text{min}}/d_{\text{min}}}{f_0/d_0 - 1/(N-k)}
  \]


  • **normed index $\rho_1$** (Bollen 1986)

  \[
  \rho_1 = \frac{f_0/d_0 - f_{\text{min}}/d_{\text{min}}}{f_0/d_0}
  \]

  $\rho_1$ is always less than or equal to 1; $\rho_1 < 0$ is unlikely in practice. See the discussion in Bollen (1989a).
• **nonnormed index** $\Delta_2$ (Bollen 1989a)

$$\Delta_2 = \frac{f_0 - f_{\text{min}}}{f_0 - \frac{d_{\text{min}}}{(N-k)}}$$

is a modification of Bentler and Bonett’s $\Delta$ that uses $d$ and “lessens the dependence” on $N$. See the discussion in (Bollen 1989b). $\Delta_2$ is identical to the IFI2 index of Mulaik et al. (1989).

• **parsimonious normed fit index** (James, Mulaik, and Brett 1982)

The PNFI is a modification of Bentler-Bonett’s normed fit index that takes parsimony of the model into account,

$$\text{PNFI} = \frac{d_{\text{min}} (f_0 - f_{\text{min}})}{d_0} \frac{f_{\text{min}}}{f_0}$$

The PNFI uses the same parsimonious factor as the parsimonious GFI of Mulaik et al. (1989).

**Fit Indices and Estimation Methods**

Note that not all fit indices are reasonable or appropriate for all estimation methods set by the METHOD= option of the PROC CALIS statement. The availability of fit indices is summarized as follows:

• Adjusted (elliptic) chi-square and its probability are available only for METHOD=ML or GLS and with the presence of raw data input.

• For METHOD=ULS or DWLS, probability of the chi-square value, RMSEA and its confidence intervals, probability of close fit, ECVI and its confidence intervals, critical N index, Z-test, AIC, CAIC, SBC, and measure of centrality are not appropriate and therefore not displayed.

**Individual Fit Indices for Multiple Groups**

When you compare the fits of individual groups in a multiple-group analysis, you can examine the residuals of the groups to gauge which group is fitted better than the others. While examining residuals is good for knowing specific locations with inadequate fit, summary measures like fit indices for individual groups would be more convenient for overall comparisons among groups.

Although the overall fit function is a weighted sum of individual fit functions for groups, these individual functions are not statistically independent. Therefore, in general you cannot partition the degrees of freedom or $\chi^2$ value according to the groups. This eliminates the possibility of breaking down those fit indices that are functions of degrees of freedom or $\chi^2$ for group comparison purposes. Bearing this fact in mind, PROC CALIS computes only a limited number of descriptive fit indices for individual groups.

• **fit function**

The overall fit function is:

$$F = \sum_{r=1}^{k} a_r F_r$$

where $a_r = \frac{(N_r - 1)}{(N-k)}$ and $F_r$ are the group weight and the discrepancy function for group $r$, respectively. The value of unweighted fit function $F_r$ for the $r$th group is denoted by:

$$f_r$$
This $F_r$ value provides a measure of fit in the $r$th group without taking the sample size into account. The larger the $F_r$, the worse the fit for the group.

**percentage contribution to the chi-square**
The percentage contribution of group $r$ to the chi-square is:

$$\text{percentage contribution} = a_r F_r / f_{\min} \times 100\%$$

where $F_r$ is the value of $F_r$ with $F$ minimized at the value $f_{\min}$. This percentage value provides a descriptive measure of fit of the moments in group $r$, weighted by its sample size. The group with the largest percentage contribution accounts for the most lack of fit in the overall model.

**root mean square residual (RMR)**
For the $r$th group, the total number of moments being modeled is:

$$g = p_r (p_r + 1 + 2 \delta_r) / 2$$

where $p_r$ is the number of variables and $\delta_r$ is the indicator variable of the mean structures in the $r$th group. The root mean square residual for the $r$th group is:

$$\text{RMR}_r = \sqrt{\frac{1}{g} \sum_{i<j}^{p_r} \left( \left( S_r \right)_{ij} - \left( \hat{S}_r \right)_{ij} \right)^2 + 2 \delta_r \sum_{i}^{p_r} \left( \left( \bar{x}_r \right)_i - \left( \hat{\mu}_r \right)_i \right)^2}$$

**standardized root mean square residual (SRMR)**
For the $r$th group, the standardized root mean square residual is:

$$\text{SRMR}_r = \sqrt{\frac{1}{g} \sum_{i<j}^{p_r} \left( \left( S_r \right)_{ij} - \left( \hat{S}_r \right)_{ij} \right)^2 + \delta_r \sum_{i}^{p_r} \left( \left( \bar{x}_r \right)_i - \left( \hat{\mu}_r \right)_i \right)^2 / \left( \left( S_r \right)_{ii} \right)}$$

**goodness-of-fit index (GFI)**
For the ULS, GLS, and ML estimation, the goodness-of-fit index (GFI) for the $r$th group is:

$$\text{GFI}_r = 1 - \frac{\text{Tr}((W_r^{-1}(S_r - \hat{S}_r))^2) + \delta_r (\bar{x}_r - \hat{x}_r)' W_r^{-1}(\bar{x}_r - \hat{x}_r)}{\text{Tr}((W_r^{-1}S_r)^2) + \delta_r \bar{x}_r' W_r^{-1} \bar{x}_r}$$

with $W_r = I$ for ULS, $W_r = S_r$ for GLS, and $W_r = \hat{\Sigma}_r$. For the WLS and DWLS estimation,

$$\text{GFI}_r = 1 - \frac{(u_r - \hat{u}_r)' W_r^{-1}(u_r - \hat{u}_r)}{u_r' W_r^{-1} u_r}$$

where $u_r$ is the vector of observed moments and $\hat{u}_r$ is the vector of fitted moments for the $r$th group ($r = 1, \ldots, k$).

When the mean structures are modeled, vectors $u_r$ and $\hat{u}_r$ contain all the nonredundant elements $\text{vec}(S_r)$ in the covariance matrix and all the means, and $W_r$ is the weight matrix for covariances and means. When the mean structures are not modeled, $u_r$, $\hat{u}_r$, and $W_r$ contain elements pertaining to the covariance elements only. Basically, formulas presented here are the same as the case for a single-group GFI. The only thing added here is the subscript $r$ to denote individual group measures.
Assessment of Fit

- **Bentler-Bonett normed fit index (NFI)**
  For the $r$th group, the Bentler-Bonett NFI is:
  \[
  \Delta_r = \frac{f_{0r} - f_r}{f_{0r}}
  \]
  where $f_{0r}$ is the function value for fitting the independence model to the $r$th group. The larger the value of $\Delta_r$, the better is the fit for the group. Basically, the formula here is the same as the overall Bentler-Bonett NFI. The only difference is that the subscript $r$ is added to denote individual group measures.

**Squared Multiple Correlations and Determination Coefficients**

In the section, squared multiple correlations for endogenous variables are defined. Squared multiple correlation is computed for all of these five estimation methods: ULS, GLS, ML, WLS, and DWLS. These coefficients are also computed as in the LISREL VI program of Jöreskog and Sörbom (1985). The DETAE, DETSE, and DETMV determination coefficients are intended to be multivariate generalizations of the squared multiple correlations for different subsets of variables. These coefficients are displayed only when you specify the `PDETERM` option.

- **$R^2$ values corresponding to endogenous variables**
  \[
  R^2 = 1 - \frac{\hat{\text{Var}}(y)}{\text{Var}(y)}
  \]
  where $y$ denotes an endogenous variable, $\hat{\text{Var}}(y)$ denotes its variance, and $\text{Var}(y)$ denotes its error (or unsystematic) variance. The variance and error variance are estimated under the model.

- **total determination of all equations**
  \[
  \text{DETAE} = 1 - \frac{|\hat{\text{Ecov}}(y, \eta)|}{|\text{Cov}(y, \eta)|}
  \]
  where the $y$ vector denotes all manifest dependent variables, the $\eta$ vector denotes all latent dependent variables, $\hat{\text{Cov}}(y, \eta)$ denotes the covariance matrix of $y$ and $\eta$, and $\text{Ecov}(y, \eta)$ denotes the error covariance matrix of $y$ and $\eta$. The covariance matrices are estimated under the model.

- **total determination of latent equations**
  \[
  \text{DETSE} = 1 - \frac{|\hat{\text{Ecov}}(\eta)|}{|\text{Cov}(\eta)|}
  \]
  where the $\eta$ vector denotes all latent dependent variables, $\hat{\text{Cov}}(\eta)$ denotes the covariance matrix of $\eta$, and $\text{Ecov}(\eta)$ denotes the error covariance matrix of $\eta$. The covariance matrices are estimated under the model.

- **total determination of the manifest equations**
  \[
  \text{DETMV} = 1 - \frac{|\hat{\text{Ecov}}(y)|}{|\text{Cov}(y)|}
  \]
where the $y$ vector denotes all manifest dependent variables, $\text{Cov}(y)$ denotes the covariance matrix of $y$, $\text{Ecov}(y)$ denotes the error covariance matrix of $y$, and $|A|$ denotes the determinant of matrix $A$. All the covariance matrices in the formula are estimated under the model.

You can also use the `DETERM` statement to request the computations of determination coefficients for any subsets of dependent variables.

---

**Case-Level Residuals, Outliers, Leverage Observations, and Residual Diagnostics**

**Residual M-Distances and Outlier Detection**

In structural equation modeling, residual analysis has been developed traditionally for the elements in the moment matrices such as mean vector and covariance matrix. See the section “Residuals in the Moment Matrices” on page 1482 for a detailed description of this type of residual. In fact, almost all fit indices in the field were developed more or less based on measuring the overall magnitude of the residuals in the moment matrices. See the section “Assessment of Fit” on page 1481 for examples of fit indices such as standardized root mean square residual (SRMR), adjusted goodness-of-fit index (AGFI), and so on.

However, recent research advances make it possible to study case-level or observational-level residuals. Although case-level residuals have not yet been used to assess overall model fit, they are quite useful as model diagnostic tools. This section describes how case-level residuals are computed and applied to the detection of outliers, the identification of leverage observations, and various residual diagnostics.

The main difficulty of case-level residual analysis in general structural equation modeling is the presence of latent variables in the model. Another difficulty is the involvement of multivariate responses. Both are nonissues in multiple regression analysis because there is no latent variable and there is only a single response variable. Hence, a good starting point for this section is to explain how PROC CALIS handles these two issues in general structural equation modeling. The residual diagnostic techniques described in the following draw heavily from Yuan and Hayashi (2010).

Latent variables in models are not observed, and their values must be predicted from the data. Although the indeterminacy of factor scores is a well-known issue, it does not mean that factor scores cannot be reasonably derived or predicted from the observed data. One of the factor score estimation methods is Bartlett’s method. Suppose for a factor model that $\Lambda$ represents a $p \times m$ factor matrix and $\Psi$ represents a $p \times p$ error covariance matrix, where $p$ is the number of manifest variables and $m$ is the number of factors. Bartlett’s factor scores are defined by the formula

$$ f = (\Lambda'\Psi^{-1}\Lambda)^{-1}\Lambda'\Psi^{-1}y $$

where $f$ represents an $m \times 1$ random vector of factors and $y$ represents a $p \times 1$ random vector of manifest variables. Yuan and Hayashi (2010) generalize Bartlett’s formulas to a certain class of structural equation models. Essentially, they provide formulas that the quantities $\Lambda$ and $\Psi$ are computable functions from the parameter estimates of the structural equation models. PROC CALIS adopts their formulas with an extension due to Yung and Yuan (2013). With the estimation of factor scores, residuals in structural equation models can be computed as if all values of factors are present. Hence, case-level residual analysis becomes possible.

With the possibility of a multivariate residual vector in structural equation models, a summary measure for the overall magnitude of the residuals in observations is needed. Yuan and Hayashi (2010) consider
the Mahalanobis distance (M-distance) of case-level residuals. For observation \(i\) with residual vector \(\hat{e}_i\) of dimension \(h \times 1\), the residual M-distance of observation \(i\) is defined as

\[
d_{ri} = \sqrt{(L\hat{e}_i)'(L\Omega_{\hat{e}}L')^{-1}(L\hat{e}_i)}
\]

where \(L ((h - q) \times h)\) is a matrix that reduces \(\hat{e}\) to \((h - q)\) independent components, \(q\) is the number of independent factors to be estimated in the structural equation model, and \(\Omega_{\hat{e}}\) is the covariance matrix of \(\hat{e}\). The reduction of the residual vector into independent components is necessary when the number of factors \(q\) is not zero in the model. For \(q > 0\), \(\Omega_{\hat{e}}\) is not invertible and cannot be used in computing the M-distances for residuals. Hence, the covariance matrix \(L\Omega_{\hat{e}}L'\) of independent components \(L\hat{e}\) is used instead. In practice, \(\hat{e}_i\) and \(\Omega_{\hat{e}}\) are evaluated at the estimated parameter values. \(L\) can be computed after \(\hat{e}_i\) is estimated.

Because the residual M-distance \(d_{ri}\) is nonnegative and is a distance measure, the interpretation of \(d_{ri}\) is straightforward—the larger it is, the farther away observation \(i\) is from the origin of the residuals. Therefore, the residual M-distance \(d_{ri}\) can be used to detect outliers. The criterion for defining outliers is based on a selection of an \(\alpha\)-level for the residual M-distance distribution. Under the multivariate normality distribution of the residuals, Yuan and Hayashi (2010) find that \(d_{ri}\) is distributed as a \(\chi_r\) variate (that is, the square root of the corresponding \(\chi^2_r\) variate), where \(r = h - q\) is the degrees of freedom of the distribution. Therefore, a probability-based criterion for outlier detection can be used. An observation is an outlier according to the \(\alpha\)-level criterion if its residual M-distance \(d_{ri}\) is located within the upper \(\alpha \times 100\%\) region of the \(\chi_r\) distribution. Mathematically, an observation with \(d_{ri}\) satisfying the following criterion is an outlier if

\[
\text{prob}(\chi_r \geq d_{ri}) < \alpha
\]

The larger the \(\alpha\), the more liberal the criterion for detecting outliers. By default, PROC CALIS sets \(\alpha\) to 0.01. You can override this value by using the ALPHAOOUT= option.

**Leverage Observations**

Leverage observations are those with large M-distances in the exogenous variables of the model. For example, in the context of a confirmatory factor model, Yuan and Hayashi (2010) define the M-distance of exogenous latent factors \(d_{fi}\) by the formula

\[
d_{fi} = \sqrt{f_i'(\Omega_f)^{-1}f_i}
\]

where \(f_i\) is the Bartlett’s factor scores for observation \(i\) and \(\Omega_f\) is the covariance matrix of the factor scores. See Yuan and Hayashi (2010) for the formulas for \(f_i\) and \(\Omega_f\). PROC CALIS uses a more general formula to compute the M-distance of exogenous observed variables \(x_i\) and latent variables \(f_i\) in structural equation models. That is, with exogenous variables \(v_i = (x_i f_i)'\), the leverage M-distance \(d_{v_i}\) is computed as

\[
d_{v_i} = \sqrt{v_i'(\Omega_v)^{-1}v_i}
\]

where \(\Omega_v\) is the covariance matrix of the exogenous variables. Leverage observations are those judged to have large leverage M-distances.

As with outlier detection, you need to set a criterion for declaring leverage observations. Specifically, an observation with \(d_{v_i}\) satisfying the following criterion is a leverage observation (or leverage point):

\[
\text{prob}(\chi_c \geq d_{v_i}) < \alpha
\]
where $\chi^2_c$ is the square root of the $\chi^2$ variate, with $c$ being the dimension of $v_i$. The larger the $\alpha$, the more liberal the criterion for declaring leverage observations. The default $\alpha$ value is 0.01. You can override this value by using the ALPHALEV= option.

Unlike the outliers that are usually judged as undesirable in models, leverage observations might or might not be bad for model estimation. Simply stated, a leverage observation is bad if it is also an outlier according to its residual M-distance value $d_r_i$. Otherwise, it is a “good” leverage observation and should not be removed in model estimation.

### Residual Diagnostics

Case-level residuals analysis can be very useful in detecting outliers, anomalies in residual distributions, and nonlinear functional relationships. All these techniques are graphical in nature and can be found in most textbooks in regression analysis (see, for example, Belsley, Kuh, and Welsch 1980). PROC CALIS provides these graphical analyses in the context of structural equation modeling. However, because PROC CALIS deals with multivariate responses and predictors in general, in some residual analysis the residuals are measured in terms of M-distances, which are always positive. PROC CALIS can produce the following graphical plots (with their ODS graph names in parentheses) for residual diagnostics:

- **Residual histogram (CaseResidualHistogram):** the distribution of residual M-distances is shown as a histogram. The residual M-distances are theoretically distributed as a $\chi$-variante. Use the PLOTS=CRESHIST option to request this plot.

- **Residual by leverage plot (ResidualByLeverage):** the observations are plotted in a two-dimensional space according to their M-distances for residuals and leverages. Criteria for classifying outliers and leverage observations are shown in the plot so that outliers and leverage observations are located in well-defined regions of the two-dimensional space. Use the PLOTS=RESBYLEV option to request this plot.

- **Residual by predicted values (ResidualByPredicted):** the residuals (not the M-distances) of a dependent observed variable are plotted against the predicted values. This is also called the residual on fit plot. If the relationship of the dependent variable to the predictors is truly linear in the model, the residuals should be randomly dispersed in the plot. If the residuals show some systematic pattern with the predicted values, unexplained nonlinear relationships of the dependent observed variable with other variables or heteroscedasticity of error variance might exist. Use the PLOTS=RESBYPRED option to request this plot.

- **Residual by quantile plot or Q-Q plot (ResidualByQuantile):** the residual M-distances are plotted against their theoretical quantiles. Observations that depart significantly from the theoretical line (with slope=1) are problematic observations. Use the PLOTS=QQPLOT option to request this plot.

- **Residual percentile by theoretical percentile plot or P-P plot (ResPercentileByExpPercentile):** the observed residual M-distance percentiles are plotted against the theoretical percentiles. Observations that depart significantly from the theoretical line (with slope=1) are problematic observations. Use the PLOTS=PPPLOT option to request this plot.
Residual Diagnostics with Robust Estimation

With the presence of multiple outliers in a data set, estimation might suffer from the so-called masking effect. This problem refers to the fact that some prominent outliers in the data might have biased the estimation so that usual residual diagnostic tools might fail to detect some less prominent outliers. As a result, subsequent analyses that are based on the data set with outliers removed might actually not be free from the effects of the outliers.

Robust estimation deals with the outliers differently. Instead of relying on diagnostic tools for detection and removal of outliers, robust methods downweight the outliers during the estimation so that the undesirable outlier effect on estimation is minimized effectively. No outliers need to be removed during or after the robust estimation. Hence, the masking effect is not an issue. In addition, with the use of robust estimation, residual diagnostics for the purpose of outlier removal become a redundant concept. In such a scenario, outlier diagnostics are more important for the purpose of identification than for removal. You can use the ROBUST=RES option (or the default ROBUST option) to downweight the outliers during model estimation.

Similarly, detecting leverage observations might also suffer from the masking effect. That is, some leverage observations might not be detected if the diagnostic results are based on model estimation that includes the leverage observations themselves. Unfortunately, the ROBUST=RES option downweights only the residual outliers and does not downweight the leverage observations during estimation. However, the ROBUST=SAT option does downweight outlying observations in all variable dimensions, and hence it could be used if the masking effect of leverage observations is a concern.

Following the recommendation by Yuan and Hayashi (2010), PROC CALIS uses an eclectic approach to compute the residual and leverage M-distances. That is, when a robust estimation is requested (using either the ROBUST=SAT or ROBUST=RES option) with case-level residual analysis (using the RESIDUAL option), the residual M-distances are computed from the robust estimation results that downweight the model outliers only (that is, the procedure invoked by the ROBUST=RES(E) or RES(F) option); but the leverage M-distances are computed from the robust estimation results that downweight the outlying observations in all variable dimensions (that is, the procedure invoked by the ROBUST=SAT option). Consequently, both the residual M-distances and the leverage M-distances calculated from this eclectic approach are free from the respective masking effects—and this is true for any robust estimation methods implemented in PROC CALIS.

It is important to note that this eclectic approach applies only to the computations of the residual and leverage M-distances. The parameter estimates, fit statistics, and the case-level residuals that are displayed in the residual or fit plots are still obtained from the specific robust method requested. Table 29.10 summarizes the robust methods that PROC CALIS uses to compute the parameter estimates, fit statistics, case-level residuals, and residual and leverage M-distances with different robust options.

<table>
<thead>
<tr>
<th>Robust Option Specified</th>
<th>ROBUST=SAT</th>
<th>ROBUST=RES(E)</th>
<th>ROBUST=RES(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimates</td>
<td>ROBUST=SAT</td>
<td>ROBUST=RES(E)</td>
<td>ROBUST=RES(F)</td>
</tr>
<tr>
<td>Model fit</td>
<td>ROBUST=SAT</td>
<td>ROBUST=RES(E)</td>
<td>ROBUST=RES(F)</td>
</tr>
<tr>
<td>Case-level residuals</td>
<td>ROBUST=SAT</td>
<td>ROBUST=RES(E)</td>
<td>ROBUST=RES(F)</td>
</tr>
<tr>
<td>Residual M-distances</td>
<td>ROBUST=RES(E)</td>
<td>ROBUST=RES(E)</td>
<td>ROBUST=RES(F)</td>
</tr>
<tr>
<td>Leverage M-distances</td>
<td>ROBUST=SAT</td>
<td>ROBUST=SAT</td>
<td>ROBUST=SAT</td>
</tr>
</tbody>
</table>
One important implication from this table is that ROBUST=SAT and ROBUST=RES(E) lead to the same residual and leverage diagnostic results, which are all free from masking effects.

In conclusion, even though outlier removal is unnecessary with robust estimation, case-level residual diagnostics are still useful for identifying outliers and leverage observations. Needless to say, other residual diagnostics such as residual M-distance distribution, Q-Q plots, P-P plots, and residual on fit plots are useful whether you use robust estimation or not.

---

**Total, Direct, and Indirect Effects**

Most structural equation models involve the specification of the effects of variables on each other. Whenever you specify equations in the LINEQS model, paths in the PATH model, path coefficient parameters in the RAM model, variable-factor relations in the FACTOR model, or regression coefficients in model matrices of the LISMOD model, you are specifying direct effects of predictor variables on outcome variables. All direct effects are represented by the associated regression coefficients, either fixed or free, in the specifications. You can examine the direct effect estimates easily in the output for model estimation.

However, direct effects are not the only effects that are important. In some cases, the indirect effects or total effects are of interest too. For example, suppose Self-Esteem is an important factor of Job Performance in your theory. Although it does not have a direct effect on Job Performance, it affects Job Performance through its influences on Motivation and Endurance. Also, Motivation has a direct effect on Endurance in your theory. The following path diagram summarizes such a theory:

**Figure 29.23** Direct and Indirect Effects of Self-Esteem on Job Performance

Clearly, each path in the diagram represents a direct effect of a predictor variable on an outcome variable. Less apparent are the total and indirect effects implied by the same path diagram. Despite this, interesting theoretical questions regarding the total and indirect effects can be raised in such a model. For example, even though there is no direct effect of Self-Esteem on Job Performance, what is its indirect effect on Job Performance? In addition to its direct effect on Job Performance, Motivation also has an indirect effect on Job Performance via its effect on Endurance. So, what is the total effect of Motivation on Job Performance and what portion of this total effect is indirect? The TOTEFF option of the CALIS statement and the EFFPART statement are designed to address these questions. By using the TOTEFF option or the EFFPART statement, PROC CALIS can compute the total, direct, and indirect effects of any sets of predictor variables on any sets of outcome variables. In this section, formulas for computing these effects are presented.
Formulas for Computing Total, Direct and Indirect Effects

No matter which modeling language is used, variables in a model can be classified into three groups. The first group is the so-called dependent variables, which serve as outcome variables at least once in the model. The other two groups consist of the remaining independent variables, which never serve as outcome variables in the model. The second group consists of independent variables that are unsystematic sources such as error and disturbance variables. The third group consists of independent variables that are systematic sources only.

Any variable, no matter which group it falls into, can have effects on the first group of variables. By definition, however, effects of variables in the first group on the other two groups do not exist. Because the effects of unsystematic sources in the second group are treated as residual effects on the first group of dependent variables, these effects are trivial in the sense that they always serve as direct effects only. That is, the effects from the second group of unsystematic sources partition trivially—total effects are always the same as the direct effects for this group. Therefore, for the purpose of effect analysis or partitioning, only the first group (dependent variables) and the third group (systematic independent variables) are considered.

Define $u$ to be the set of $n_u$ dependent variables in the first group and $w$ to be the set of $n_w$ systematic independent variables in the third group. Variables in both groups can be manifest or latent. All variables in the effect analysis is thus represented by the vector $(u', w')'$.

The $(n_u + n_w) \times (n_u + n_w)$ matrix $D$ of direct effects refers to the path coefficients from all column variables to the row variables. This matrix is represented by:

$$
D = \begin{pmatrix}
\beta & \gamma \\
0 & 0
\end{pmatrix}
$$

where $\beta$ is an $(n_u \times n_u)$ matrix for direct effects of dependent variables on dependent variables and $\gamma$ is an $(n_u \times n_w)$ matrix for direct effects of systematic independent variables on dependent variables. By definition, there should not be any direct effects on independent variables, and therefore the lower submatrices of $D$ are null. In addition, by model restrictions the diagonal elements of matrix $\beta$ must be zeros.

Correspondingly, the $(n_u + n_w) \times (n_u + n_w)$ matrix $T$ of total effects of column variables on the row variables is computed by:

$$
T = \begin{pmatrix}
(I - \beta)^{-1} - I & (I - \beta)^{-1} \gamma \\
0 & 0
\end{pmatrix}
$$

Finally, the $(n_u + n_w) \times (n_u + n_w)$ matrix $\mu$ of indirect effects of column variables on the row variables is computed by the difference between $T$ and $D$ as:

$$
\mu = \begin{pmatrix}
(I - \beta)^{-1} - I - \beta & (I - \beta)^{-1} \gamma - \gamma \\
0 & 0
\end{pmatrix}
$$

In PROC CALIS, any subsets of $D$, $T$, and $\mu$ can be requested via the specification in the EFFPART statement. All you need to do is to specify the sets of column variables (variables that have effects on others) and row variables (variables that receive the effects, direct or indirect). Specifications of the column and row variables are done conveniently by specifying variable names—no matrix terminology is needed. This feature is very handy if you have some focused subsets of effects that you want to analyze a priori. See the EFFPART statement on page 1257 for details about specifications.
Stability Coefficient of Reciprocal Causation

For recursive models (that is, models without cyclical paths of effects), using the preceding formulas for computing the total effect and the indirect effect is appropriate without further restrictions. However, for non-recursive models (that is, models with reciprocal effects or cyclical effects) the appropriateness of the preceding formulas for effect computations is restricted to situations with the convergence of the total effects.

A necessary and sufficient condition for the convergence of total effects (with or without cyclical paths) is when all eigenvalues, complex or real, of the $\hat{\mathbf{B}}$ matrix fall into a unit circle (see Bentler and Freeman 1983). Equivalently, define the stability coefficient of reciprocal causation as the largest length (modulus) of the eigenvalues of the $\hat{\mathbf{B}}$ matrix. A stability coefficient less than one would ensure that all eigenvalues, complex or real, of the $\hat{\mathbf{B}}$ matrix fall into a unit circle. Hence, stability coefficient that is less than one is a necessary and sufficient condition for the convergence of the total effects, which in turn substantiates the appropriateness of total and indirect effect computations. Whenever effect analysis or partitioning is requested, PROC CALIS will check the appropriateness of effect computations by evaluating the stability coefficient of reciprocal causation. If the stability coefficient is greater than one, computations of the total and indirect effects will not be done.

Standardized Solutions

Standardized solutions are useful when you want to compare parameter values that are measured on quite different scales. PROC CALIS provides standardized solutions routinely. In standardizing a solution, parameters are classified into five groups:

- **path coefficients, regression coefficients, or direct effects**
  With each parameter $\alpha$ in this group, there is an associated outcome variable and a predictor variable. Denote the predicted variance of the outcome variable by $\sigma_\alpha^2$ and the variance of the predictor variable by $\sigma_p^2$, the standardized parameter $\alpha^*$ is:
  \[
  \alpha^* = \frac{\alpha \sigma_p}{\sigma_\alpha}
  \]

- **fixed ones for the path coefficients attached to error or disturbance terms**
  These fixed values are unchanged in standardization.

- **variances and covariances among exogenous variables, excluding errors and disturbances**
  Let $\sigma_{ij}$ be the covariance between variables $i$ and $j$. In this notation, $\sigma_{ii}$ is the variance of variable $i$. The standardized covariance $\sigma_{ij}^*$ is:
  \[
  \sigma_{ij}^* = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii} \sigma_{jj}}}
  \]
  When $i = j$, $\sigma_{ii}^*$ takes the value of 1 for all $i$. Also, $\sigma_{ij}^*$ is the correlation between the $i$th and $j$th variables.

- **variances and covariances among errors or disturbances**
  Denote the error covariance parameter as $\theta_{ij}$ so that $\theta_{ij}$ represents the variance parameter of error variable $i$. Associated with each error or disturbance variable $i$ is a unique outcome variable. Let the
variance of such an outcome variable be $\sigma_{ij}$. In the standardized solution, the error covariance $\theta_{ij}$ is rescaled as:

$$\theta_{ij}^* = \frac{\theta_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$$

Notice that when $i = j$, $\theta_{ii}^*$ is not standardized to 1 in general. In fact, the error (disturbance) variance is simply rescaled by the reciprocal of the variance of the associated dependent variable. As a result, the rescaled error (disturbance) variance represents the proportion of variation of the dependent variable due to the unsystematic source. By the same token, $\theta_{ij}^*$ does not represent the correlation between errors $i$ and $j$. It is a rescaled covariance of the errors involved.

**intercepts and means of variables**

These parameters are fixed zeros in the standardized solution.

While formulas for the standardized solution are useful in computing the parameter values in the standardized solution, it is conceptually more useful to explain how variables are being transformed in the standardization process. The following provides a summary of the transformation process:

- Observed and latent variables, excluding errors or disturbances, are centered and then divided by their corresponding standard deviations. Therefore, in the standardized solution, all these variables will have variance equal to 1. In other words, these variables are truly standardized.

- Errors or disturbances are divided by the standard deviations of the corresponding outcome variables. In the standardized solution, these variables will not have variance equal to 1 in general. However, the rescaled error variances represent the proportion of unexplained or unsystematic variance of the corresponding outcome variables. Therefore, errors or disturbances in the standardized solution are simply rescaled but not standardized.

Standardized total, direct, and indirect effects are computed using formulas presented in the section “Total, Direct, and Indirect Effects” on page 1498, but with the standardized parameter values substituted into the formulas.

Although parameter values associated with different scales are made more comparable in the standardized solution, a precaution should be mentioned. In the standardized solution, the original constraints on parameters in the unstandardized solution are usually lost. These constraints, however, might underscore some important theoretical position that needs to be maintained in the model. Destroying these constraints in the standardized solution means that interpretations or comparisons of parameter values in the standardized solution are made without maintaining the original theoretical position. You must judge whether such a departure from the original constraints poses conceptual difficulties for interpreting the standardized solution.

---

**Modification Indices**

While fitting structural equation models is mostly a confirmatory analytic procedure, it does not prevent you from exploring what might have been a better model given the data. After fitting your theoretical structural equation model, you might want to modify the original model in order to do one of the following:
• add free parameters to improve the model fit significantly
• reduce the number of parameters without affecting the model fit too much

The first kind of model modification can be achieved by using the Lagrange multiplier (LM) test indices. Parameters that have the largest LM indices would increase the model fit the most. In general, adding more parameters to your model improves the overall model fit, as measured by those absolute or standalone fit indices (see the section “Overall Model Fit Indices” on page 1484 for more details). However, adding parameters liberally makes your model more prone to sampling errors. It also makes your model more complex and less interpretable in most cases. A disciplined use of LM test indices is highly recommended. In addition to the model fit improvement indicated by the LM test indices, you should also consider the theoretical significance when adding particular parameters. See Example 29.28 for an illustration of the use of LM test indices for improving model fit.

The second kind of model modification can be achieved by using the Wald statistics. Parameters that are not significant in your model may be removed from the model without affecting the model fit too much. In general, removing parameters from your model decreases the model fit, as measured by those absolute or standalone fit indices (see the section “Overall Model Fit Indices” on page 1484 for more details). However, for just a little sacrifice in model fit, removing non-significant parameters increases the simplicity and precision of your model, which is the virtue that any modeler should look for.

Whether adding parameters by using the LM test indices or removing unnecessary parameters by the Wald statistics, you should not treat your modified model as if it were your original hypothesized model. That is, you should not publish your modified model as if it were hypothesized a priori. It is perfectly fine to use modification indices to gain additional insights for future research. But if you want to publish your modified model together with your original model, you should report the modification process that leads to your modified model. Theoretical justifications of the modified model should be supplemented if you want to make strong statements to support your modified model. Whenever possible, the best practice is to show reasonable model fit of the modified model with new data.

To modify your model either by LM test indices or Wald statistics, you can use the MODIFICATION or MOD option in the PROC CALIS statement. To customize the LM tests by setting specific regions of parameters, you can use the LMTESTS statements. PROC CALIS computes and displays the following default set of modification indices:

• **univariate Lagrange multiplier (LM) test indices for parameters in the model**
  These are second-order approximations of the decrease in the $\chi^2$ value that would result from allowing the fixed parameter values in the model to be freed to estimate. LM test indices are ranked within their own parameter regions in the model. The ones that suggest greatest model improvements (that is, greatest $\chi^2$ drop) are ranked first. Depending on the type of your model, the set of possible parameter regions varies. For example, in a RAM model, modification indices are ranked in three different parameter regions for the covariance structures: path coefficients, variances of and covariances among exogenous variables, and the error variances and covariances. In addition to the value of the Lagrange multiplier, the corresponding $p$-value ($df = 1$) and the approximate change of the parameter value are displayed.

  If you use the LMMAT option in the LMTESTS statement, LM test indices are shown as elements in model matrices. Not all elements in a particular model matrix will have LM test indices. Elements that are already free parameters in the model do not have LM test indices. Instead, the parameter names are shown. Elements that are model restricted values (for example, direct path from a variable to itself
must be zero) are labeled Excluded in the matrix output. When you customize your own regions of LM tests, some elements might also be excluded from a custom set of LM tests. These elements are also labeled as Excluded in the matrix output. If an LM test for freeing a parameter would result in a singular information matrix, the corresponding element in the matrix is labeled as Singular.

- **univariate Lagrange multiplier test indices for releasing equality constraints**
  These are second-order approximations of the decrease in the $\chi^2$ value that would result from the release of equality constraints. Multiple equality constraints containing $n > 2$ parameters are tested successively in $n$ steps, each assuming the release of one of the equality-constrained parameters. The expected change of the parameter values of the separated parameter and the remaining parameter cluster are displayed, too.

- **univariate Lagrange multiplier test indices for releasing active boundary constraints**
  These are second-order approximations of the decrease in the $\chi^2$ value that would result from the release of the active boundary constraints specified in the BOUNDS statement.

- **stepwise multivariate Wald statistics for constraining free parameters to 0**
  These are second-order approximations of the increases in $\chi^2$ value that would result from constraining free parameters to zero in a stepwise fashion. In each step, the parameter that would lead to the smallest increase in the multivariate $\chi^2$ value is set to 0. Besides the multivariate $\chi^2$ value and its $p$-value, the univariate increments are also displayed. The process stops when the univariate $p$-value is smaller than the specified value in the SLMW= option, of which the default value is 0.05.

All of the preceding tests are approximations. You can often obtain more accurate tests by actually fitting different models and computing likelihood ratio tests. For more details about the Wald and the Lagrange multiplier test, see MacCallum (1986), Buse (1982), Bentler (1986), or Lee (1985). Note that relying solely on the LM tests to modify your model can lead to unreliable models that capitalize purely on sampling errors. See MacCallum, Roznowski, and Necowitz (1992) for the use of LM tests.

For large model matrices, the computation time for the default modification indices can considerably exceed the time needed for the minimization process.

The modification indices are not computed for unweighted least squares or diagonally weighted least squares estimation.

---

**Missing Values and the Analysis of Missing Patterns**

If the DATA= data set contains raw data (rather than a covariance or correlation matrix), in general observations with missing values for any variables in the analysis are omitted from the computations. The only exception is with METHOD=FIML. Incomplete observations with at least one nonmissing variables in the analysis are also used for the estimation.

If a covariance or correlation matrix is read, missing values are allowed as long as every pair of variables has at least one nonmissing value. Unlike the raw data input, METHOD=FIML does not allow missing values in the covariance or correlation matrix.

When you use METHOD=FIML, PROC CALIS provide several analyses on the missing patterns of the raw input data sets. First, PROC CALIS shows the coverage results for the means and covariances. The coverage results refer to the proportions of data present for computing the means and the covariances. Because
distinct missing patterns in the data sets are possible, the coverage proportions for the individual means and covariances could vary. Average coverage proportions of the means and covariances give you an overall idea about the missingness (or the lack of). In order to help locate the problematic means and covariances that have the low coverage, PROC CALIS shows the rank orders of the smallest coverages of mean and covariance elements. The number of smallest coverages shown for the means is equal to half of the total number of variables. The number of smallest coverages shown for the covariances is equal to half of the total number of the distinct elements in the lower triangular of the covariance matrix. However, in both cases at most 10 smallest coverages would be shown.

Second, PROC CALIS ranks the most frequent missing patterns in the data set (the nonmissing pattern is excluded in the ranking). Because the number of missing patterns could be quite large, PROC CALIS displays only a limited number of most frequent missing patterns in the output. You can use the MAXMISSPAT= and the TMISSPAT= options to control the number of missing patterns to display. See these options for details.

Third, PROC CALIS shows the means of the most frequent missing patterns, along with the means for the nonmissing pattern for comparison.

See Example 29.15 for an illustration of the use of the full information maximum likelihood method and the analysis of missing patterns. For examples and details about the FIML method and its missing data treatment in PROC CALIS, see Yung and Zhang (2011); Zhang and Yung (2011).

---

**Measures of Multivariate Kurtosis**

In many applications, the manifest variables are not even approximately multivariate normal. If this happens to be the case with your data set, the default generalized least squares and maximum likelihood estimation methods are not appropriate, and you should compute the parameter estimates and their standard errors by an asymptotically distribution-free method, such as the WLS estimation method. If your manifest variables are multivariate normal, then they have a zero relative multivariate kurtosis, and all marginal distributions have zero kurtosis (Browne 1982). If your DATA= data set contains raw data, PROC CALIS computes univariate skewness and kurtosis and a set of multivariate kurtosis values. By default, the values of univariate skewness and kurtosis are corrected for bias (as in PROC UNIVARIATE), but using the BIASKUR option enables you to compute the uncorrected values also. The values are displayed when you specify the PROC CALIS statement option KURTOSIS.

In the following formulas, $N$ denotes the sample size and $p$ denotes the number of variables.

- **corrected variance for variable $z_j$**
  \[
  \sigma_j^2 = \frac{1}{N - 1} \sum_i^N (z_{ij} - \bar{z}_j)^2
  \]

- **uncorrected univariate skewness for variable $z_j$**
  \[
  \gamma_1(j) = \frac{N \sum_i^N (z_{ij} - \bar{z}_j)^3}{\sqrt{N \sum_i^N (z_{ij} - \bar{z}_j)^2}}
  \]

- **corrected univariate skewness for variable $z_j$**
  \[
  \gamma_1(j) = \frac{N \sum_i^N (z_{ij} - \bar{z}_j)^3}{(N - 1)(N - 2) \sigma_j^3}
  \]
• uncorrected univariate kurtosis for variable \( z_j \)

\[
\gamma_{2(j)} = \frac{N\sum_i^N (z_{ij} - \bar{z}_j)^4}{\left[\sum_i^N (z_{ij} - \bar{z}_j)^2\right]^2} - 3
\]

• corrected univariate kurtosis for variable \( z_j \)

\[
\gamma_{2(j)} = \frac{N(N + 1)}{(N - 1)(N - 2)(N - 3)} \frac{\sum_i^N (z_{ij} - \bar{z}_j)^4}{\sigma_j^4} - \frac{3(N - 1)^2}{(N - 2)(N - 3)}
\]

• Mardia’s multivariate kurtosis

\[
\gamma_2 = \frac{1}{N} \sum_i^N [(z_i - \bar{z})'S^{-1}(z_i - \bar{z})]^2 - p(p + 2)
\]

where \( S \) is the biased sample covariance matrix with \( N \) as the divisor.

• relative multivariate kurtosis

\[
\eta_2 = \frac{\gamma_2 + p(p + 2)}{p(p + 2)}
\]

• normalized multivariate kurtosis

\[
\kappa_0 = \frac{\gamma_2}{\sqrt{8p(p + 2)/N}}
\]

• Mardia based kappa

\[
\kappa_1 = \frac{\gamma_2}{p(p + 2)}
\]

• mean scaled univariate kurtosis

\[
\kappa_2 = \frac{1}{3p} \sum_j^p \gamma_{2(j)}
\]

• adjusted mean scaled univariate kurtosis

\[
\kappa_3 = \frac{1}{3p} \sum_j^p \gamma_{2(j)}^*
\]

with

\[
\gamma_{2(j)}^* = \begin{cases} 
\gamma_{2(j)}, & \text{if } \gamma_{2(j)} > \frac{-6}{p+2} \\
\frac{-6}{p+2}, & \text{otherwise}
\end{cases}
\]
If variable $Z_j$ is normally distributed, the uncorrected univariate kurtosis $\gamma_2(j)$ is equal to 0. If $Z$ has an $p$-variate normal distribution, Mardia’s multivariate kurtosis $\gamma_2$ is equal to 0. A variable $Z_j$ is called leptokurtic if it has a positive value of $\gamma_2(j)$ and is called platykurtic if it has a negative value of $\gamma_2(j)$. The values of $\kappa_1$, $\kappa_2$, and $\kappa_3$ should not be smaller than the following lower bound (Bentler 1985):

$$\hat{\kappa} \geq \frac{-2}{p + 2}$$

PROC CALIS displays a message if $\kappa_1$, $\kappa_2$, or $\kappa_3$ falls below the lower bound.

If weighted least squares estimates (METHOD=WLS or METHOD=ADF) are specified and the weight matrix is computed from an input raw data set, the CALIS procedure computes two more measures of multivariate kurtosis.

- **multivariate mean kappa**

$$\kappa_4 = \frac{1}{m} \sum_{i} \sum_{j} \sum_{k} \sum_{l} \hat{\kappa}_{ij,kl} - 1$$

where

$$\hat{\kappa}_{ij,kl} = \frac{s_{ij,kl}}{s_{ij}s_{kl} + s_{ik}s_{jl} + s_{il}s_{jk}}$$

and $m = p(p + 1)(p + 2)(p + 3)/24$ is the number of elements in the vector $s_{ij,kl}$ (Bentler 1985).

- **multivariate least squares kappa**

$$\kappa_5 = \frac{s'_4 s_2}{s'_2 s_2} - 1$$

where $s_2$ is the vector of the elements in the denominator of $\hat{\kappa}$ (Bentler 1985) and $s_4$ is the vector of the $s_{ij,kl}$, which is defined as

$$s_{ij,kl} = \frac{1}{N} \sum_{r=1}^{N} (z_{ri} - \bar{z}_i)(z_{rj} - \bar{z}_j)(z_{rk} - \bar{z}_k)(z_{rl} - \bar{z}_l)$$

The occurrence of significant nonzero values of Mardia’s multivariate kurtosis $\gamma_2$ and significant amounts of some of the univariate kurtosis values $\gamma_2(j)$ indicate that your variables are not multivariate normal distributed. Violating the multivariate normality assumption in (default) generalized least squares and maximum likelihood estimation usually leads to the wrong approximate standard errors and incorrect fit statistics based on the $\chi^2$ value. In general, the parameter estimates are more stable against violation of the normal distribution assumption. For more details, see Browne (1974, 1982, 1984).

**Initial Estimates**

Each optimization technique requires a set of initial values for the parameters. To avoid local optima, the initial values should be as close as possible to the globally optimal solution. You can check for local optima
by running the analysis with several different sets of initial values; the RANDOM= option in the PROC CALIS statement is useful in this regard.

Except for the case of exploratory FACTOR model, you can specify initial estimates manually for all different types of models. If you do not specify some of the initial estimates and the RANDOM= option is not used, PROC CALIS will use a combination of good strategic methods to compute initial estimates for your model. These initial estimation methods are used in PROC CALIS:

- two-stage least squares estimation
- instrumental variable method (Hägglund 1982; Jennrich 1987)
- approximate factor analysis method
- ordinary least squares estimation
- estimation method of McDonald (McDonald and Hartmann 1992)
- observed moments of manifest exogenous variables

The choice of initial estimation methods is dependent on the data and on the model. In general, it is difficult to tell in advance which initial estimation methods will be used for a given analysis. However, PROC CALIS displays the methods used to obtain initial estimates in the output. Notice that none of these initial estimation methods can be applied to the COSAN model because of the general formulation of the COSAN model. If you do not provide initial parameter estimates for the COSAN model, the default values or random values are used (see the START= and the RANDOM= options).

Poor initial values can cause convergence problems, especially with maximum likelihood estimation. Sufficiently large positive initial values for variance estimates (as compared with the covariance estimates) might help prevent a nonnegative definite initial predicted covariance model matrix from happening. If maximum likelihood estimation fails to converge, it might help to use METHOD=LSML, which uses the final estimates from an unweighted least squares analysis as initial estimates for maximum likelihood. Or you can fit a slightly different but better-behaved model and produce an OUTMODEL= data set, which can then be modified in accordance with the original model and used as an INMODEL= data set to provide initial values for another analysis.

If you are analyzing a covariance or scalar product matrix, be sure to take into account the scales of the variables. The default initial values might be inappropriate when some variables have extremely large or small variances.

---

**Use of Optimization Techniques**

No algorithm for optimizing general nonlinear functions exists that can always find the global optimum for a general nonlinear minimization problem in a reasonable amount of time. Since no single optimization technique is invariably superior to others, PROC CALIS provides a variety of optimization techniques that work well in various circumstances. However, you can devise problems for which none of the techniques in PROC CALIS can find the correct solution. All optimization techniques in PROC CALIS use $O(n^2)$ memory except the conjugate gradient methods, which use only $O(n)$ of memory and are designed to optimize problems with many parameters.
The PROC CALIS statement NLOPTIONS can be especially helpful for tuning applications with nonlinear equality and inequality constraints on the parameter estimates. Some of the options available in NLOPTIONS can also be invoked as PROC CALIS options. The NLOPTIONS statement can specify almost the same options as the SAS/OR NLP procedure.

Nonlinear optimization requires the repeated computation of the following:

- the function value (optimization criterion)
- the gradient vector (first-order partial derivatives)
- for some techniques, the (approximate) Hessian matrix (second-order partial derivatives)
- values of linear and nonlinear constraints
- the first-order partial derivatives (Jacobian) of nonlinear constraints

For the criteria used by PROC CALIS, computing the gradient takes more computer time than computing the function value, and computing the Hessian takes much more computer time and memory than computing the gradient, especially when there are many parameters to estimate. Unfortunately, optimization techniques that do not use the Hessian usually require many more iterations than techniques that do use the (approximate) Hessian, and so they are often slower. Techniques that do not use the Hessian also tend to be less reliable (for example, they might terminate at local rather than global optima).

The available optimization techniques are displayed in the following table and can be chosen by the OMETHOD=name option.

<table>
<thead>
<tr>
<th>OMETHOD=</th>
<th>Optimization Technique</th>
<th>Derivatives</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVMAR</td>
<td>Levenberg-Marquardt method</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>TRUREG</td>
<td>Trust-region method</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>NEWRAP</td>
<td>Newton-Raphson method with line search</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>NRRIDG</td>
<td>Newton-Raphson method with ridging</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>QUANEW</td>
<td>Quasi-Newton methods (DBFGS, DDFP, BFGS, DFP)</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>DBLDOG</td>
<td>Double-dogleg method (DBFGS, DDFP)</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>CONGRA</td>
<td>Conjugate gradient methods (PB, FR, PR, CD)</td>
<td>x</td>
<td>-</td>
</tr>
</tbody>
</table>

The following table shows, for each optimization technique, which derivatives are needed (first-order or second-order) and what kind of constraints (boundary, linear, or nonlinear) can be imposed on the parameters.

<table>
<thead>
<tr>
<th>OMETHOD=</th>
<th>Derivatives</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVMAR</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>TRUREG</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>NEWRAP</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>NRRIDG</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>QUANEW</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>DBLDOG</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>CONGRA</td>
<td>x</td>
<td>-</td>
</tr>
</tbody>
</table>

The Levenberg-Marquardt, trust-region, and Newton-Raphson techniques are usually the most reliable, work well with boundary and general linear constraints, and generally converge after a few iterations to a
precise solution. However, these techniques need to compute a Hessian matrix in each iteration. Computing the approximate Hessian in each iteration can be very time- and memory-consuming, especially for large problems (more than 200 parameters, depending on the computer used). For large problems, a quasi-Newton technique, especially with the BFGS update, can be far more efficient.

For a poor choice of initial values, the Levenberg-Marquardt method seems to be more reliable.

If memory problems occur, you can use one of the conjugate gradient techniques, but they are generally slower and less reliable than the methods that use second-order information.

There are several options to control the optimization process. You can specify various termination criteria. You can specify the GCONV= option to specify a relative gradient termination criterion. If there are active boundary constraints, only those gradient components that correspond to inactive constraints contribute to the criterion. When you want very precise parameter estimates, the GCONV= option is useful. Other criteria that use relative changes in function values or parameter estimates in consecutive iterations can lead to early termination when active constraints cause small steps to occur. The small default value for the FCONV= option helps prevent early termination. Using the MAXITER= and MAXFUNC= options enables you to specify the maximum number of iterations and function calls in the optimization process. These limits are especially useful in combination with the INMODEL= and OUTMODEL= options; you can run a few iterations at a time, inspect the results, and decide whether to continue iterating.

Nonlinearly Constrained QN Optimization

The algorithm used for nonlinearly constrained quasi-Newton optimization is an efficient modification of Powell’s Variable Metric Constrained WatchDog (VMCWD) algorithm (Powell 1978a, b, 1982a, b) A similar but older algorithm (VF02AD) is part of the Harwell library. Both VMCWD and VF02AD use Fletcher’s VE02AD algorithm (also part of the Harwell library) for positive definite quadratic programming. The PROC CALIS QUANEW implementation uses a quadratic programming subroutine that updates and downdates the approximation of the Cholesky factor when the active set changes. The nonlinear QUANEW algorithm is not a feasible point algorithm, and the value of the objective function might not necessarily decrease (minimization) or increase (maximization) monotonically. Instead, the algorithm tries to reduce a linear combination of the objective function and constraint violations, called the merit function.

The following are similarities and differences between this algorithm and VMCWD:

- A modification of this algorithm can be performed by specifying VERSION=1, which replaces the update of the Lagrange vector \( \mu \) with the original update of Powell (1978a, b), which is used in VF02AD. This can be helpful for some applications with linearly dependent active constraints.

- If the VERSION= option is not specified or VERSION=2 is specified, the evaluation of the Lagrange vector \( \mu \) is performed in the same way as Powell (1982a, b) describes.

- Instead of updating an approximate Hessian matrix, this algorithm uses the dual BFGS (or DFP) update that updates the Cholesky factor of an approximate Hessian. If the condition of the updated matrix gets too bad, a restart is done with a positive diagonal matrix. At the end of the first iteration after each restart, the Cholesky factor is scaled.

- The Cholesky factor is loaded into the quadratic programming subroutine, automatically ensuring positive definiteness of the problem. During the quadratic programming step, the Cholesky factor of the projected Hessian matrix \( Z_k^T GZ_k \) and the QT decomposition are updated simultaneously when the active set changes. See Gill et al. (1984) for more information.
• The line-search strategy is very similar to that of Powell (1982a, b). However, this algorithm does not call for derivatives during the line search; hence, it generally needs fewer derivative calls than function calls. The VMCWD algorithm always requires the same number of derivative and function calls. It was also found in several applications of VMCWD that Powell’s line-search method sometimes uses steps that are too long during the first iterations. In those cases, you can use the INSTEP= option specification to restrict the step length \( \alpha \) of the first iterations.

• The watchdog strategy is similar to that of Powell (1982a, b). However, this algorithm does not return automatically after a fixed number of iterations to a former better point. A return here is further delayed if the observed function reduction is close to the expected function reduction of the quadratic model.

• Although Powell’s termination criterion still is used (as FCONV2), the QUANEW implementation uses two additional termination criteria (GCONV and ABSGCONV).

This algorithm is automatically invoked when you specify the NLINCON statement. The nonlinear QUANEW algorithm needs the Jacobian matrix of the first-order derivatives (constraints normals) of the constraints:

\[
(\nabla c_i) = \left( \frac{\partial c_i}{\partial x_j} \right), \quad i = 1, \ldots, n_c, \quad j = 1, \ldots, n
\]

where \( n_c \) is the number of nonlinear constraints for a given point \( x \).

You can specify two update formulas with the UPDATE= option:

• UPDATE=DBFGS performs the dual BFGS update of the Cholesky factor of the Hessian matrix. This is the default.

• UPDATE=DDFP performs the dual DFP update of the Cholesky factor of the Hessian matrix.

This algorithm uses its own line-search technique. All options and parameters (except the INSTEP= option) controlling the line search in the other algorithms do not apply here. In several applications, large steps in the first iterations are troublesome. You can specify the INSTEP= option to impose an upper bound for the step size \( \alpha \) during the first five iterations. The values of the LCSINGULAR=, LCEPSILON=, and LCDEACT= options (which control the processing of linear and boundary constraints) are valid only for the quadratic programming subroutine used in each iteration of the nonlinear constraints QUANEW algorithm.

Optimization and Iteration History

The optimization and iteration histories are displayed by default because it is important to check for possible convergence problems. The optimization history includes the following summary of information about the initial state of the optimization:

• the number of constraints that are active at the starting point, or more precisely, the number of constraints that are currently members of the working set. If this number is followed by a plus sign, there are more active constraints, of which at least one is temporarily released from the working set due to negative Lagrange multipliers.

• the value of the objective function at the starting point
• if the (projected) gradient is available, the value of the largest absolute (projected) gradient element

• for the TRUREG and LEVMAR subroutines, the initial radius of the trust region around the starting point

The optimization history ends with some information concerning the optimization result:

• the number of constraints that are active at the final point, or more precisely, the number of constraints that are currently members of the working set. If this number is followed by a plus sign, there are more active constraints, of which at least one is temporarily released from the working set due to negative Lagrange multipliers.

• the value of the objective function at the final point

• if the (projected) gradient is available, the value of the largest absolute (projected) gradient element

• other information specific to the optimization technique

The iteration history generally consists of one line of displayed output containing the most important information for each iteration.

The iteration history always includes the following:

• the iteration number

• the number of iteration restarts

• the number of function calls

• the number of active constraints

• the value of the optimization criterion

• the difference between adjacent function values

• the maximum of the absolute gradient components that correspond to inactive boundary constraints

An apostrophe trailing the number of active constraints indicates that at least one of the active constraints is released from the active set due to a significant Lagrange multiplier.

For the Levenberg-Marquardt technique (LEVMAR), the iteration history also includes the following information:

• an asterisk trailing the iteration number when the computed Hessian approximation is singular and consequently ridged with a positive lambda value. If all or the last several iterations show a singular Hessian approximation, the problem is not sufficiently identified. Thus, there are other locally optimal solutions that lead to the same optimum function value for different parameter values. This implies that standard errors for the parameter estimates are not computable without the addition of further constraints.
• the value of the Lagrange multiplier (lambda). This value is 0 when the optimum of the quadratic function approximation is inside the trust region (a trust-region-scaled Newton step can be performed) and is greater than 0 when the optimum of the quadratic function approximation is located at the boundary of the trust region (the scaled Newton step is too long to fit in the trust region and a quadratic constraint optimization is performed). Large values indicate optimization difficulties. For a nonsingular Hessian matrix, the value of lambda should go to 0 during the last iterations, indicating that the objective function can be well approximated by a quadratic function in a small neighborhood of the optimum point. An increasing lambda value often indicates problems in the optimization process.

• the value of the ratio $\rho$ between the actually achieved difference in function values and the predicted difference in the function values on the basis of the quadratic function approximation. Values much less than 1 indicate optimization difficulties. The value of the ratio $\rho$ indicates the goodness of the quadratic function approximation. In other words, $\rho << 1$ means that the radius of the trust region has to be reduced; a fairly large value of $\rho$ means that the radius of the trust region does not need to be changed. And a value close to or greater than 1 means that the radius can be increased, indicating a good quadratic function approximation.

For the Newton-Raphson technique (NRRIDG), the iteration history also includes the following information:

• the value of the ridge parameter. This value is 0 when a Newton step can be performed, and it is greater than 0 when either the Hessian approximation is singular or a Newton step fails to reduce the optimization criterion. Large values indicate optimization difficulties.

• the value of the ratio $\rho$ between the actually achieved difference in function values and the predicted difference in the function values on the basis of the quadratic function approximation. Values much less than 1.0 indicate optimization difficulties.

For the Newton-Raphson with line-search technique (NEWRAP), the iteration history also includes the following information:

• the step size $\alpha$ (alpha) computed with one of the line-search algorithms

• the slope of the search direction at the current parameter iterate. For minimization, this value should be significantly negative. Otherwise, the line-search algorithm has difficulty reducing the function value sufficiently.

For the trust-region technique (TRUREG), the iteration history also includes the following information:

• an asterisk after the iteration number when the computed Hessian approximation is singular and consequently ridged with a positive lambda value.

• the value of the Lagrange multiplier (lambda). This value is zero when the optimum of the quadratic function approximation is inside the trust region (a trust-region-scaled Newton step can be performed) and is greater than zero when the optimum of the quadratic function approximation is located at the boundary of the trust region (the scaled Newton step is too long to fit in the trust region and a quadratically constrained optimization is performed). Large values indicate optimization difficulties. As in Gay (1983), a negative lambda value indicates the special case of an indefinite Hessian matrix (the smallest eigenvalue is negative in minimization).
• the value of the radius $\Delta$ of the trust region. Small trust-region radius values combined with large lambda values in subsequent iterations indicate optimization problems.

For the quasi-Newton (QUANEW) and conjugate gradient (CONGRA) techniques, the iteration history also includes the following information:

• the step size (alpha) computed with one of the line-search algorithms
• the descent of the search direction at the current parameter iterate. This value should be significantly smaller than 0. Otherwise, the line-search algorithm has difficulty reducing the function value sufficiently.

Frequent update restarts (rest) of a quasi-Newton algorithm often indicate numerical problems related to required properties of the approximate Hessian update, and they decrease the speed of convergence. This can happen particularly if the ABSCGCONV= termination criterion is too small—that is, when the requested precision cannot be obtained by quasi-Newton optimization. Generally, the number of automatic restarts used by conjugate gradient methods are much higher.

For the nonlinearly constrained quasi-Newton technique, the iteration history also includes the following information:

• the maximum value of all constraint violations,
$$\text{conmax} = \max\{|c_i(x)| : c_i(x) < 0\}$$
• the value of the predicted function reduction used with the GCONV and FCONV2 termination criteria,
$$\text{pred} = |g(x^{(k)})s(x^{(k)})| + \sum_{i=1}^{m} |\lambda_i c_i(x^{(k)})|$$
• the step size $\alpha$ of the quasi-Newton step. Note that this algorithm works with a special line-search algorithm.
• the maximum element of the gradient of the Lagrange function,
$$\text{lfgmax} = \nabla_x L(x^{(k)}, \lambda^{(k)})$$
$$= \nabla_x f(x^{(k)}) - \sum_{i=1}^{m} \lambda_i^{(k)} \nabla x c_i(x^{(k)})$$

For the double dogleg technique, the iteration history also includes the following information:

• the parameter $\lambda$ of the double-dogleg step. A value $\lambda = 0$ corresponds to the full (quasi) Newton step.
• the slope of the search direction at the current parameter iterate. For minimization, this value should be significantly negative.
Line-Search Methods

In each iteration \( k \), the (dual) quasi-Newton, hybrid quasi-Newton, conjugate gradient, and Newton-Raphson minimization techniques use iterative line-search algorithms that try to optimize a linear, quadratic, or cubic approximation of the nonlinear objective function \( f \) of \( n \) parameters \( x \) along a feasible descent search direction \( s^{(k)} \) as follows:

\[
    f(x^{(k+1)}) = f(x^{(k)} + \alpha^{(k)} s^{(k)})
\]

by computing an approximately optimal scalar \( \alpha^{(k)} > 0 \). Since the outside iteration process is based only on the approximation of the objective function, the inside iteration of the line-search algorithm does not have to be perfect. Usually, it is satisfactory that the choice of \( \alpha \) significantly reduces (in a minimization) the objective function. Criteria often used for termination of line-search algorithms are the Goldstein conditions (Fletcher 1987).

Various line-search algorithms can be selected by using the LIS= option on page 1218. The line-search methods LIS=1, LIS=2, and LIS=3 satisfy the left-hand-side and right-hand-side Goldstein conditions (Fletcher 1987).

The line-search method LIS=2 seems to be superior when function evaluation consumes significantly less computation time than gradient evaluation. Therefore, LIS=2 is the default value for Newton-Raphson, (dual) quasi-Newton, and conjugate gradient optimizations.

Restricting the Step Length

Almost all line-search algorithms use iterative extrapolation techniques that can easily lead to feasible points where the objective function \( f \) is no longer defined (resulting in indefinite matrices for ML estimation) or is difficult to compute (resulting in floating point overflows). Therefore, PROC CALIS provides options that restrict the step length or trust region radius, especially during the first main iterations.

The inner product \( g^t s \) of the gradient \( g \) and the search direction \( s \) is the slope of \( f(\alpha) = f(x + \alpha s) \) along the search direction \( s \) with step length \( \alpha \). The default starting value \( \alpha^{(0)} = \alpha^{(k,0)} \) in each line-search algorithm \( \min_{\alpha > 0} f(x + \alpha s) \) during the main iteration \( k \) is computed in three steps:

1. Use either the difference \( df = |f^{(k)} - f^{(k-1)}| \) of the function values during the last two consecutive iterations or the final stepsize value \( \alpha^- \) of the previous iteration \( k - 1 \) to compute a first value \( \alpha_1^{(0)} \).
   - Using the DAMPSTEP<\( r > \) option:
     \[
     \alpha_1^{(0)} = \min(1, r\alpha^-)
     \]
     The initial value for the new step length can be no greater than \( r \) times the final step length \( \alpha^- \) of the previous iteration. The default is \( r = 2 \).
   - Not using the DAMPSTEP option:
     \[
     \alpha_1^{(0)} = \begin{cases} \text{step} & \text{if } 0.1 \leq \text{step} \leq 10 \\ 10 & \text{if } \text{step} > 10 \\ 0.1 & \text{if } \text{step} < 0.1 \end{cases}
     \]
     with
     \[
     \text{step} = \begin{cases} \frac{df}{|g^t s|} & \text{if } |g^t s| \geq \epsilon \max(100df, 1) \\ 1 & \text{otherwise} \end{cases}
     \]
This value of $\alpha_1^{(0)}$ can be too large and can lead to a difficult or impossible function evaluation, especially for highly nonlinear functions such as the EXP function.

2. During the first five iterations, the second step enables you to reduce $\alpha_1^{(0)}$ to a smaller starting value $\alpha_2^{(0)}$ using the INSTEP=$r$ option:

$$\alpha_2^{(0)} = \min(\alpha_1^{(0)}, r)$$

After more than five iterations, $\alpha_2^{(0)}$ is set to $\alpha_1^{(0)}$.

3. The third step can further reduce the step length by

$$\alpha_3^{(0)} = \min(\alpha_2^{(0)}, \min(10, u))$$

where $u$ is the maximum length of a step inside the feasible region.

The INSTEP=$r$ option lets you specify a smaller or larger radius of the trust region used in the first iteration by the trust-region, double-dogleg, and Levenberg-Marquardt algorithms. The default initial trust region radius is the length of the scaled gradient (Moré 1978). This default length for the initial trust region radius corresponds to the default radius factor of $r = 1$. This choice is successful in most practical applications of the TRUREG, DBLDOG, and LEVMAR algorithms. However, for bad initial values used in the analysis of a covariance matrix with high variances or for highly nonlinear constraints (such as using the EXP function) in your SAS programming statements, the default start radius can result in arithmetic overflows. If this happens, you can try decreasing values of INSTEP=$r$ ($0 < r < 1$), until the iteration starts successfully. A small factor $r$ also affects the trust region radius of the next steps because the radius is changed in each iteration by a factor $0 < c \leq 4$ depending on the $\rho$ ratio. Reducing the radius corresponds to increasing the ridge parameter $\lambda$ that produces smaller steps directed closer toward the gradient direction.

Computational Problems

First Iteration Overflows

Analyzing a covariance matrix that includes high variances in the diagonal and using bad initial estimates for the parameters can easily lead to arithmetic overflows in the first iterations of the minimization algorithm. The line-search algorithms that work with cubic extrapolation are especially sensitive to arithmetic overflows. If this occurs with quasi-Newton or conjugate gradient minimization, you can specify the INSTEP= option to reduce the length of the first step. If an arithmetic overflow occurs in the first iteration of the Levenberg-Marquardt algorithm, you can specify the INSTEP= option to reduce the trust region radius of the first iteration. You also can change the minimization technique or the line-search method. If none of these help, you can consider doing the following:

- scaling the covariance matrix
- providing better initial values
- changing the model
No Convergence of Minimization Process

If convergence does not occur during the minimization process, perform the following tasks:

- If there are negative variance estimates, you can do either of the following:
  - Specify the BOUNDS statement to obtain nonnegative variance estimates.
  - Specify the HEYWOOD option, if the FACTOR statement is specified.

- Change the estimation method to obtain a better set of initial estimates. For example, if you use METHOD=ML, you can do either of the following:
  - Change to METHOD=LSML.
  - Run some iterations with METHOD=DWLS or METHOD=GLS, write the results in an OUTMODEL= data set, and use the results as initial values specified by an INMODEL= data set in a second run with METHOD=ML.

- Change the optimization technique. For example, if you use the default OMETHOD=LEVMAR, you can do either of the following:
  - Change to OMETHOD=QUANEW or to OMETHOD=NEWRAP.
  - Run some iterations with OMETHOD=CONGRA, write the results in an OUTMODEL= data set, and use the results as initial values specified by an INMODEL= data set in a second run with a different OMETHOD= technique.

- Change or modify the update technique or the line-search algorithm or both when using OMETHOD=QUANEW or OMETHOD=CONGRA. For example, if you use the default update formula and the default line-search algorithm, you can do any or all of the following:
  - Change the update formula with the UPDATE= option.
  - Change the line-search algorithm with the LIS= option.
  - Specify a more precise line search with the LSPRECISION= option, if you use LIS=2 or LIS=3.

- Add more iterations and function calls by using the MAXIT= and MAXFU= options.

- Change the initial values. For many categories of model specifications, PROC CALIS computes an appropriate set of initial values automatically. However, for some of the model specifications (for example, structural equations with latent variables on the left-hand side and manifest variables on the right-hand side), PROC CALIS might generate very obscure initial values. In these cases, you have to set the initial values yourself.
  - Increase the initial values of the variance parameters by one of the following ways:
    - Set the variance parameter values in the model specification manually.
    - Use the DEPHAS= option to increase all initial variance parameter values.
  - Use a slightly different, but more stable, model to obtain preliminary estimates.
  - Use additional information to specify initial values, for example, by using other SAS software like the FACTOR, REG, SYSLIN, and MODEL (SYSNLIN) procedures for the modified, unrestricted model case.
Unidentified Model

The parameter vector $\Theta$ in the structural model

$$\Sigma = \Sigma(\Theta)$$

is said to be identified in a parameter space $G$, if

$$\Sigma(\Theta) = \Sigma(\hat{\Theta}), \quad \hat{\Theta} \in G$$

implies $\Theta = \hat{\Theta}$. The parameter estimates that result from an unidentified model can be very far from the parameter estimates of a very similar but identified model. They are usually machine dependent. Do not use parameter estimates of an unidentified model as initial values for another run of PROC CALIS.

Singular Predicted Covariance Model Matrix

Sometimes you might inadvertently specify models with singular predicted covariance model matrices (for example, by fixing diagonal elements to zero). In such cases, you cannot compute maximum likelihood estimates (the ML function value $F$ is not defined). Since singular predicted covariance model matrices can also occur temporarily in the minimization process, PROC CALIS tries in such cases to change the parameter estimates so that the predicted covariance model matrix becomes positive definite. This process does not always work well, especially if there are fixed instead of free diagonal elements in the predicted covariance model matrices. A famous example where you cannot compute ML estimates is a component analysis with fewer components than given manifest variables. See the section “FACTOR Statement” on page 1258 for more details. If you continue to obtain a singular predicted covariance model matrix after changing initial values and optimization techniques, then your model might be specified so that ML estimates cannot be computed.

Saving Computing Time

For large models, the most computing time is needed to compute the modification indices. If you do not really need the Lagrange multipliers or multiple Wald test indices (the univariate Wald test indices are the same as the $t$ values), using the NOMOD option can save a considerable amount of computing time.

Predicted Covariance Matrices with Negative Eigenvalues

A covariance matrix cannot have negative eigenvalues, since a negative eigenvalue means that some linear combination of the variables has negative variance. PROC CALIS displays a warning if the predicted covariance matrix has negative eigenvalues but does not actually compute the eigenvalues. Sometimes this warning can be triggered by 0 or very small positive eigenvalues that appear negative because of numerical error. If you want to be sure that the predicted covariance matrix you are fitting can be considered to be a variance-covariance matrix, you can use the SAS/IML command $\text{VAL}=\text{EIGVAL}(U)$ to compute the vector $\text{VAL}$ of eigenvalues of matrix $U$.

Negative $R^2$ Values

The estimated squared multiple correlations $R^2$ of the endogenous variables are computed using the estimated error variances:

$$R^2_i = 1 - \frac{\text{var}(\xi_i)}{\text{var}(\eta_i)}$$
When $\text{var}(\xi_i) > \text{var}(\eta_i)$, $R^2$ is negative. This might indicate poor model fit or $R$ square is an inappropriate measure for the model. For the latter case, for example, negative $R$ square might be due to cyclical (nonrecursive) paths in the model so that the $R$ square interpretation is not appropriate.

### Displayed Output

The output of PROC CALIS includes the following:

- a list of basic modeling information such as: the data set, the number of records read and used in the raw data set, the number of observations assumed by the statistical analysis, and the model type. When a multiple-group analysis is specified, the groups and their corresponding models are listed. This output assumes at least the PSHORT option.

- a list of all variables in the models. This output is displayed by default or by the PINITIAL option. It will not be displayed when you use the PSHORT or the PSUMMARY option.

Depending on the modeling language, the variable lists vary, as shown in the following:

- COSAN: a list of the observed variables
- FACTOR: a list of the variables and the factors
- LINEQS, PATH, and RAM: a list of the endogenous and exogenous variables specified in the model
- LISMOD: a list of $x$, $y$, $\xi$, and $\eta$- variables specified in the model
- MSTRUCT: a list of the manifest variables specified in the model

- initial model specification. This output is displayed by default or by the PINITIAL option. It will not be displayed when you use the PSHORT or the PSUMMARY option.

Depending on the modeling language, the sets of output vary, as shown in the following:

- COSAN: matrix equations for the covariance and mean structures, dimensions of the model matrices, and types and transformations of the model matrices. The initial values for free parameters, the fixed values, and the parameter names are also displayed in the matrices.

- FACTOR: factor loading matrix, factor covariance matrix, intercepts, factor means, and error variances as specified initially in the model. The initial values for free parameters, the fixed values, and the parameter names are also displayed.

- LINEQS: linear equations, variance and covariance parameters, and mean parameters as specified initially in the model. The initial values for free parameters, the fixed values, and the parameter names are also displayed.

- LISMOD: all model matrices as specified initially in the model. The initial values for free parameters, the fixed values, and the parameter names are also displayed.

- MSTRUCT: initial covariance matrix and mean vectors, with parameter names and initial values displayed.
– PATH: the path list, variance and covariance parameters, intercept and mean parameters as specified initially in the model. The initial values for free parameters, the fixed values, and the parameter names are also displayed.
– RAM: a list of parameters, their types, names, and initial values.

• mean and standard deviation of each manifest variable if you specify the SIMPLE option, as well as skewness and kurtosis if the DATA= data set is a raw data set and you specify the KURTOSIS option.

• various coefficients of multivariate kurtosis and the numbers of observations that contribute most to the normalized multivariate kurtosis if the DATA= data set is a raw data set and the KURTOSIS option is used or you specify at least the PRINT option. See the section “Measures of Multivariate Kurtosis” on page 1504 for more information.

• covariance coverage, variable coverage, average coverage of covariances and means, rank orders of the variable (mean) and covariance coverage, most frequent missing patterns in the input data set, and the means of the missing patterns when there are incomplete observations (with some missing values in the analysis variables) in the input raw data set and when you use METHOD=FIML or METHOD=LSFIML for estimation.

• covariance or correlation matrix to be analyzed and the value of its determinant if you specify the output option PCORR or PALL. A zero determinant indicates a singular data matrix. In this case, the generalized least squares estimates with default weight matrix S and maximum likelihood estimates cannot be computed.

• robust covariance matrix and its determinant and robust mean vector if you specify the output option PCORR or PALL for robust estimation with the ROBUST option. Raw data input is required.

• the weight matrix W or its inverse is displayed if GLS, WLS, or DWLS estimation is used and you specify the PWEIGHT or PALL option.

• initial estimation methods for generating initial estimates. This output is displayed by default. It will not be displayed when you use the PSHORT or the PSUMMARY option.

• vector of parameter names and initial values and gradients. This output is displayed by default, unless you specify the PSUMMARY or NOPRINT option.

• special features of the optimization technique chosen if you specify at least the PSHORT option.

• optimization history if at least the PSHORT option is specified. For more details, see the section “Use of Optimization Techniques” on page 1507.

• specific output requested by options in the NLOPTIONS statement; for example, parameter estimates, gradient, constraints, projected gradient, Hessian, projected Hessian, Jacobian of nonlinear constraints, estimated covariance matrix of parameter estimates, and information matrix. Note that the estimated covariance of parameter estimates and the information matrix are not printed for the ULS and DWLS estimation methods.

• fit summary table with various model fit test statistics or fit indices, and some basic modeling information. For the listing of fit indices and their definitions, see the section “Overall Model Fit Indices” on page 1484. Note that for ULS and DWLS estimation methods, many of those fit indices that are based on model fit χ² are not displayed. See the section “Overall Model Fit Indices” on page 1484 for details. This output can be suppressed by the NOPRINT option.
• fit comparison for multiple-group analysis. See the section “Individual Fit Indices for Multiple Groups” on page 1491 for the fit indices for group comparison. This output can be suppressed by the NOPRINT option.

• the predicted covariance matrix and its determinant and mean vector, if you specify the output option PCORR or PALL.

• outlier, leverage, and case-level residual analysis if you specify the RESIDUAL option (or at least the PRINT option) with raw data input.

• leverage and outlier plot, quantile and percentile plots of residual M-distances, distribution of residual M-distances, and residual on fit plots if you request ODS Graphics by using the relevant PLOTS= option.

• residual and normalized residual matrix if you specify the RESIDUAL option or at least the PRINT option. The variance standardized or asymptotically standardized residual matrix can be displayed also. The average residual and the average off-diagonal residual are also displayed. Note that normalized or asymptotically standardized residuals are not applicable for the ULS and DWLS estimation methods. See the section “Residuals in the Moment Matrices” on page 1482 for more details.

• rank order of the largest normalized residuals if you specify the RESIDUAL option or at least the PRINT option.

• bar chart of the normalized residuals if you specify the RESIDUAL option or at least the PRINT option.

• plotting of smoothed density functions of residuals if you request ODS Graphics by the relevant PLOTS= option.

• equations of linear dependencies among the parameters used in the model specification if the information matrix is recognized as singular at the final solution.

• the (unstandardized) estimation results for all types of models and the standardized results for all but the COSAN models. Except for ULS or DWLS estimates, the approximate standard errors, $t$ values, and $p$ values are also displayed. This output is displayed by default or if you specify the PESTIM option or at least the PSHORT option. If you specify the CI option, confidence intervals are also displayed.

Depending on the modeling language, the sets of output vary, as shown in the following:

- COSAN: all model matrices in the model.
- FACTOR: factor loading matrix, rotation matrix, rotated factor loading matrix (if rotation requested), factor covariance matrix, intercepts, factor means, and error variances in the model. Factor rotation matrix is printed for the unstandardized solution.
- LINEQS: linear equations, effects in equations, variance and covariance parameters, and mean parameters in the model.
- LISMOD: all model matrices in the model.
- MSTRUCT: covariance matrix and mean vectors.
- PATH: the path list, variance and covariance parameters, intercept and mean parameters.
- RAM: a list of parameters, their types, names, and initial values.
squared multiple correlations table which displays the error variance, total variance, and the squared multiple correlation of each endogenous variable in the model. The total variances are the diagonal elements of the predicted covariance matrix. This output is displayed if you specify the PESTIM option or at least the PSHORT option.

the total determination of all equations, the total determination of the latent equations, and the total determination of the manifest equations if you specify the PDETERM or the PALL option. See the section “Assessment of Fit” on page 1481 for more details. If you specify subsets of variables in the DETERM statements, the corresponding determination coefficients will also be shown. If one of the determinants in the formula for computing the determination coefficient is zero, the corresponding coefficient is displayed as the missing value ‘.’.

the matrix of estimated covariances among the latent variables in the model if you specify the PLATCOV option or at least the PRINT option.

the matrix of estimated covariances between latent and manifest variables in the model if you specify the PLATCOV option or at least the PRINT option.

the vector of estimated means for the latent and manifest variables in the model if you specify the PLATCOV option or at least the PRINT option.

the matrix $FSR$ of latent variable scores regression coefficients if you specify the PLATCOV option or at least the PRINT option. The $FSR$ matrix is a generalization of Lawley and Maxwell (1971, p. 109) factor scores regression matrix,

$$FSR = \hat{\Sigma}_{yy} \hat{\Sigma}_{xx}^{-1}$$

where $\hat{\Sigma}_{xx}$ is the $p \times p$ predicted covariance matrix among manifest variables and $\hat{\Sigma}_{yy}$ is the $m \times p$ matrix of the predicted covariances between latent and manifest variables, with $p$ being the number of manifest variables, and $m$ being the number of latent variables. You can multiply the observed values by this matrix to estimate the scores of the latent variables used in your model.

stability coefficient of reciprocal causation if you request the effect analysis by using the EFFPART or TOTEFF option, and you must not use the NOPRINT option.

the matrices for the total, direct, and indirect effects if you specify the EFFPART or TOTEFF option or at least the PRINT option, and you must not use the NOPRINT option. Unstandardized and standardized effects are printed in separate tables. Standard errors for the all estimated effects are also included in the output. Additional tables for effects are available if you request specialized effect analysis in the EFFPART statements.

the matrix of rotated factor loadings and the orthogonal transformation matrix if you specify the ROTATE= and PESTIM options or at least the PSHORT options. This output is available for the FACTOR models.

factor scores regression matrix, if you specify the PESTIM option or at least the PSHORT option. The determination of manifest variables is displayed only if you specify the PDETERM option.

univariate Lagrange multiplier indices if you specify the MODIFICATION (or MOD) or the PALL option. The value of a Lagrange multiplier (LM) index indicates the approximate drop in $\chi^2$ when the corresponding fixed parameter in the original model is freely estimated. The corresponding probability
(with \( df = 1 \)) and the estimated change of the parameter value are printed. Ranking of the LM indices is automatically done for prescribed parameter subsets of the original model. The LM indices with greatest improvement of \( \chi^2 \) model fit appear in the beginning of the ranking list. Note that LM indices are not applicable to the ULS and the DWLS estimation methods. See the section “Modification Indices” on page 1501 for more detail.

- matrices of univariate Lagrange multiplier (LM) indices if you specify the MODIFICATION (or MOD) or the PALL option, and the LMMAT option in the LMQEST statement. These matrices are predefined in PROC CALIS, or you can specify them in the LMQTESTS statements. If releasing a fixed parameter in the matrix would result in a singular information matrix, the string ‘Singular’ is displayed instead of the Lagrange multiplier index. If a fixed entry in the matrix is restricted by the model (for example, fixed ones for coefficients associated with error terms) or being excluded in the specified subsets in the LMQTESTS statement, the string ‘Excluded’ is displayed. Note that matrices for LM indices are not printed for the ULS and the DWLS estimation methods. See the section “Modification Indices” on page 1501 for more detail.

- univariate Lagrange multiplier test indices for releasing equality constraints if you specify the MODIFICATION (or MOD) or the PALL option. Note that this output is not applicable to the ULS and the DWLS estimation methods. See the section “Modification Indices” on page 1501 for more detail.

- univariate Lagrange multiplier test indices for releasing active boundary constraints specified by the BOUNDS statement if you specify the MODIFICATION (or MOD) or the PALL option. Note that this output is not applicable to the ULS and the DWLS estimation methods. See the section “Modification Indices” on page 1501 for more detail.

- the stepwise multivariate Wald test for constraining estimated parameters to zero constants if the MODIFICATION (or MOD) or the PALL option is specified and the univariate probability is greater than the value specified in the PMW= option (default PMW=0.05). Note that this output is not applicable to the ULS and the DWLS estimation methods. See the section “Modification Indices” on page 1501 for more detail.

- path diagrams if you specify the PLOTS=PATHDIAGRAM option or the PATHDIAGRAM statement.
**ODS Table Names**

PROC CALIS assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. For more information about ODS, see Chapter 20, “Using the Output Delivery System.”

There are numerous ODS tables in the CALIS procedure. The conditions for these ODS tables to display vary a lot. For convenience in presentation, the ODS tables for the PROC CALIS procedure are organized in the following categories:

- ODS tables for descriptive statistics, missing patterns, and residual analysis
- ODS tables for model specification and results
- ODS tables for supplementary model analysis
- ODS tables for modification indices
- ODS tables for optimization control and results

Many ODS tables are displayed when you set either a specialized option in a certain statement or a global display option in the PROC CALIS statement. Rather than requesting displays by setting specialized options separately, you can request a group of displays by using a global display option.

There are five global display levels, represented by five options: **PALL** (highest), **PRINT**, **default**, **PSHORT**, and **PSUMMARY**. The higher the level, the more output requested. The **default** printing level is in effect when you do not specify any other global printing options in the PROC CALIS statement. See the section “Global Display Options” on page 1203 for details.

In the following description of ODS tables whenever applicable, the lowest level of global printing options for an ODS table to print is listed. It is understood that global printing options at higher levels can also be used. For example, if **PSHORT** is the global display option to print an ODS table, you can also use **PALL**, **PRINT**, or **default**.

**ODS Tables for Descriptive Statistics, Missing Patterns, and Residual Analysis**

These ODS tables are group-oriented, meaning that each group has its own set of tables in the output. To display these tables in your output, you can set a specialized option in either the PROC CALIS or GROUP statement. If the specialized option is set in the PROC CALIS statement, it will apply to all groups. If the option is set in the GROUP statement, it will apply to the associated group only. Alternatively, you can set a global printing option in the PROC CALIS statement to print these tables. Either a specialized or a global printing option is sufficient to print the tables.
Table Names for Descriptive Statistics

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Specialized Option</th>
<th>Global Display Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContKurtosis</td>
<td>Contributions to kurtosis from observations</td>
<td>KURTOSIS</td>
<td>PRINT</td>
</tr>
<tr>
<td>CovRobust</td>
<td>Robust covariance matrix, with raw data input</td>
<td>PCORR&lt;sup&gt;1&lt;/sup&gt;</td>
<td>PRINT</td>
</tr>
<tr>
<td>CovDetRobust</td>
<td>Determinant of the robust covariance matrix, with raw data input</td>
<td>PCORR&lt;sup&gt;1&lt;/sup&gt;</td>
<td>PRINT</td>
</tr>
<tr>
<td>InCorr</td>
<td>Input correlation matrix</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>InCorrDet</td>
<td>Determinant of the input correlation matrix</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>InCov</td>
<td>Input covariance matrix</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>InCovDet</td>
<td>Determinant of the input covariance matrix</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>InMean</td>
<td>Input mean vector</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>Kurtosis, with raw data input</td>
<td>KURTOSIS</td>
<td>PRINT</td>
</tr>
<tr>
<td>MeanRobust</td>
<td>Robust mean vector, with raw data input</td>
<td>PCORR&lt;sup&gt;1&lt;/sup&gt;</td>
<td>PRINT</td>
</tr>
<tr>
<td>PredCorr</td>
<td>Predicted correlation matrix</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>PredCorrDet</td>
<td>Determinant of the predicted correlation matrix</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>PredCov</td>
<td>Predicted covariance matrix</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>PredCovDet</td>
<td>Determinant of the predicted covariance matrix</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>PredMean</td>
<td>Predicted mean vector</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>RidgedCovRobust</td>
<td>Ridged robust covariance matrix, with raw data input</td>
<td>PCORR&lt;sup&gt;1&lt;/sup&gt;</td>
<td>PRINT</td>
</tr>
<tr>
<td>RidgedCovDetRobust</td>
<td>Determinant of the ridged robust covariance matrix, with raw data input</td>
<td>PCORR&lt;sup&gt;1&lt;/sup&gt;</td>
<td>PRINT</td>
</tr>
<tr>
<td>RidgedInCorr</td>
<td>Ridged input correlation matrix</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>RidgedInCorrDet</td>
<td>Determinant of the ridged input correlation matrix</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>RidgedInCov</td>
<td>Ridged input covariance matrix</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>RidgedInCovDet</td>
<td>Determinant of the ridged input covariance matrix</td>
<td>PCORR</td>
<td>PALL</td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Simple statistics, with raw data input</td>
<td>SIMPLE</td>
<td>Default</td>
</tr>
<tr>
<td>Weights</td>
<td>Weight matrix</td>
<td>PWEIGHT</td>
<td>PALL</td>
</tr>
<tr>
<td>WeightsDet</td>
<td>Determinant of the weight matrix</td>
<td>PWEIGHT</td>
<td>PALL</td>
</tr>
</tbody>
</table>

1. Robust estimation with the ROBUST option is required.
### Table Names for Missing Pattern Analysis

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Specialized Option</th>
<th>Global Display Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AveCoverage</td>
<td>Average proportion coverages of means (variances) and covariances</td>
<td>SIMPLE or PCORR</td>
<td>Default²</td>
</tr>
<tr>
<td>MeanCovCoverage</td>
<td>Proportions of data present for means (variances) and covariances</td>
<td>SIMPLE or PCORR</td>
<td>Default²</td>
</tr>
<tr>
<td>MissPatternsMeans</td>
<td>Means of the nonmissing and the most frequent missing patterns</td>
<td>SIMPLE or PCORR</td>
<td>Default²</td>
</tr>
<tr>
<td>RankCovCoverage</td>
<td>Rank order of the covariance coverages</td>
<td>SIMPLE or PCORR</td>
<td>Default²</td>
</tr>
<tr>
<td>RankMissPatterns</td>
<td>Rank order of the most frequent missing patterns</td>
<td>SIMPLE or PCORR</td>
<td>Default²</td>
</tr>
<tr>
<td>RankVariableCoverage</td>
<td>Rank order of the proportion coverages of the variables</td>
<td>SIMPLE or PCORR</td>
<td>Default²</td>
</tr>
</tbody>
</table>

1. You can use the NOMISSPAT option in the PROC CALIS statement to suppress the analytic output of the missing patterns. If you use the NOMISSPAT option in the GROUP statements, only the output of the missing pattern analysis for the corresponding groups are suppressed.

2. PROC CALIS outputs these tables by default only when there are incomplete observations in the data sets and you use METHOD=FIML or METHOD=LSFIML for estimation.

### Table Names for Residual Displays

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Specialized Option</th>
<th>Global Display Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residuals in the Moment Matrices</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AsymStdRes</td>
<td>Asymptotically standardized residual matrix</td>
<td>RESIDUAL=ASYSTAND¹</td>
<td>PALL</td>
</tr>
<tr>
<td>AveAsymStdRes</td>
<td>Average of absolute asymptotically standardized residual values</td>
<td>RESIDUAL=ASYSTAND¹</td>
<td>PALL</td>
</tr>
<tr>
<td>AveNormRes</td>
<td>Average of absolute normalized residual values</td>
<td>RESIDUAL=NORM¹</td>
<td>PALL</td>
</tr>
<tr>
<td>AveRawRes</td>
<td>Average of absolute raw residual values</td>
<td>RESIDUAL</td>
<td>PALL</td>
</tr>
<tr>
<td>AveVarStdRes</td>
<td>Average of absolute variance standardized residual values</td>
<td>RESIDUAL=VARSTAND¹</td>
<td>PALL</td>
</tr>
<tr>
<td>DistAsymStdRes</td>
<td>Distribution of asymptotically standardized residuals</td>
<td>RESIDUAL=ASYSTAND¹</td>
<td>PALL</td>
</tr>
<tr>
<td>DistNormRes</td>
<td>Distribution of normalized residuals</td>
<td>RESIDUAL=NORM¹</td>
<td>PALL</td>
</tr>
<tr>
<td>DistRawRes</td>
<td>Distribution of raw residuals</td>
<td>RESIDUAL</td>
<td>PALL</td>
</tr>
<tr>
<td>DistVarStdRes</td>
<td>Distribution of variance standardized residuals</td>
<td>RESIDUAL=VARSTAND¹</td>
<td>PALL</td>
</tr>
</tbody>
</table>
### ODS Table Name | Description | Specialized Option | Global Display Option
--- | --- | --- | ---
NormRes | Normalized residual matrix | RESIDUAL=NORM<sup>1</sup> | PALL
RawRes | Raw residual matrix | RESIDUAL<sup>2</sup> | PALL
RankAsymStdRes | Rank order of asymptotically standardized residuals | RESIDUAL=ASYSTAND<sup>1</sup> | PALL
RankNormRes | Rank order of normalized residuals | RESIDUAL=NORM<sup>1</sup> | PALL
RankRawRes | Rank order of raw residuals | RESIDUAL | PALL
RankVarStdRes | Rank order of variance standardized residuals | RESIDUAL=VARSTAND<sup>1</sup> | PALL
VarStdRes | Variance standardized residual matrix | RESIDUAL=VARSTAND<sup>1</sup> | PALL
Case-Level Residuals<sup>3</sup> | DepartResidualDist | Departures from the theoretical distribution for the residual M-distances | RESIDUAL | PALL
Leverage | Observations with largest leverage M-distances | RESIDUAL | PALL
Outlier | Observations with largest residual M-distances | RESIDUAL | PALL

1. The RESIDUAL= option specifies the RESIDUAL option and the type of residuals. For example, RESIDUAL=ASYSTAND requests asymptotically standardized residuals, in addition to the tables enabled by the RESIDUAL option. In some cases, the requested types of residuals are not available due to the specific estimation method or the data type used. When this occurs, PROC CALIS will determine the appropriate types of normalized or standardized residuals to display.
2. Raw residuals are also printed for correlation analysis even if RESIDUAL or PALL is not specified.
3. Raw data input is required.

#### ODS Tables for Model Specification and Results

Some ODS tables of this group are model-oriented. Others are not. Model-oriented ODS tables are printed for each model, while others are printed no more than once no matter how many models you have.

#### Non-Model-Oriented ODS Tables

The ODS tables that are not model-oriented are listed in the following table:
<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Global Display Option</th>
<th>Additional Specification Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddParms</td>
<td>Estimates for additional parameters</td>
<td>PSHORT</td>
<td>PARAMETERS statement</td>
</tr>
<tr>
<td>AddParmsInit</td>
<td>Initial values for additional parameters</td>
<td>PSHORT</td>
<td>PARAMETERS statement</td>
</tr>
<tr>
<td>Fit</td>
<td>Fit summary</td>
<td>PSUMMARY</td>
<td></td>
</tr>
<tr>
<td>GroupFit</td>
<td>Fit comparison among groups</td>
<td>PSUMMARY</td>
<td>Multiple groups</td>
</tr>
<tr>
<td>ModelingInfo</td>
<td>General modeling information</td>
<td>PSHORT</td>
<td></td>
</tr>
<tr>
<td>ModelSummary</td>
<td>Summary of models and their labels and types</td>
<td>PSHORT</td>
<td>Multiple models¹</td>
</tr>
<tr>
<td>ParmFunc</td>
<td>Parametric function testing</td>
<td>PSHORT</td>
<td>TESTFUNC statement</td>
</tr>
<tr>
<td>Simtests</td>
<td>Simultaneous tests of parametric functions</td>
<td>PSHORT</td>
<td>SIMTESTS statement</td>
</tr>
</tbody>
</table>

1. This table is displayed when you have multiple models that have labels specified by the LABEL= option, or when you define a model with more than a single level of reference by using the REFMODEL option. Otherwise, the ModelingInfo table contains all pertinent information regarding the models in the analysis.

**Model-Oriented ODS Tables**

These ODS tables are model-oriented, meaning that each model has its own set of ODS tables in the output. There are three types of model specification and results printing in PROC CALIS: initial specification, (unstandardized) estimated model results, and standardized model results. To distinguish these three types of ODS tables, different suffixes for the ODS table names are used. An “Init” suffix indicates initial specification, while a “Std” suffix indicates standardized solutions. All other tables are for unstandardized solutions.

These ODS tables require some specialized options to print. If you set the specialized option in the PROC CALIS statement, it applies to all models. If you set the specialized option in the MODEL statement, it applies to the associated model only. Alternatively, to print all these ODS tables, you can use the PSHORT or any higher level global printing option in the PROC CALIS statement. Either a specialized or a global printing option is sufficient to print these ODS tables. The following is a summary of the specialized and global printing options for these three types of ODS tables:

<table>
<thead>
<tr>
<th>Type of ODS Tables</th>
<th>Table Name Suffix</th>
<th>Specialized Option</th>
<th>Global Display Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial specification</td>
<td>Init</td>
<td>PINITIAL</td>
<td>PSHORT</td>
</tr>
<tr>
<td>Unstandardized solutions</td>
<td>(none)</td>
<td>PESTIM</td>
<td>PSHORT</td>
</tr>
<tr>
<td>Standardized solutions</td>
<td>Std</td>
<td>PESTIM, and NOSTAND not used</td>
<td>PSHORT</td>
</tr>
</tbody>
</table>

In the following list of ODS tables, the prefixes of the ODS table names indicate the modeling language required for the ODS tables to print. The last column of the list indicates whether the PRIMAT option is needed to print the corresponding ODS tables in matrix formats. You can use the PRIMAT option either in the PROC CALIS or MODEL statement. If you want matrix output for all models, set this option in the
PROC CALIS statement. If you want matrix output for a specific model, set this option in the associated MODEL statement only.

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Additional Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>COSANVariables</td>
<td>Variables in the COSAN model</td>
<td></td>
</tr>
<tr>
<td>COSANModel</td>
<td>Mean and covariance structure formulas</td>
<td></td>
</tr>
<tr>
<td>COSANMatrixSummary</td>
<td>Summary of COSAN model matrices</td>
<td></td>
</tr>
<tr>
<td>COSANMatrix</td>
<td>Estimated model matrix</td>
<td></td>
</tr>
<tr>
<td>COSANMatrixInit</td>
<td>Initial model matrix</td>
<td></td>
</tr>
<tr>
<td>FACTORCov</td>
<td>Estimated factor covariances</td>
<td></td>
</tr>
<tr>
<td>FACTORCovInit</td>
<td>Initial factor covariances</td>
<td></td>
</tr>
<tr>
<td>FACTORCovStd</td>
<td>Factor correlations</td>
<td></td>
</tr>
<tr>
<td>FACTORErrVar</td>
<td>Estimated error variances</td>
<td></td>
</tr>
<tr>
<td>FACTORErrVarInit</td>
<td>Initial error variances</td>
<td></td>
</tr>
<tr>
<td>FACTORErrVarStd</td>
<td>Standardized results for error variances</td>
<td></td>
</tr>
<tr>
<td>FACTORIntercepts</td>
<td>Estimated intercepts</td>
<td></td>
</tr>
<tr>
<td>FACTORInterceptsInit</td>
<td>Initial intercepts</td>
<td></td>
</tr>
<tr>
<td>FACTORLoadings</td>
<td>Estimated factor loadings</td>
<td></td>
</tr>
<tr>
<td>FACTORLoadingsInit</td>
<td>Initial factor loadings</td>
<td></td>
</tr>
<tr>
<td>FACTORLoadingsStd</td>
<td>Standardized factor loadings</td>
<td></td>
</tr>
<tr>
<td>FACTORMeans</td>
<td>Estimated factor means</td>
<td></td>
</tr>
<tr>
<td>FACTORMeansInit</td>
<td>Initial factor means</td>
<td></td>
</tr>
<tr>
<td>FACTORRotCov</td>
<td>Estimated rotated factor covariances</td>
<td></td>
</tr>
<tr>
<td>FACTORRotCovStd</td>
<td>Rotated factor correlations</td>
<td></td>
</tr>
<tr>
<td>FACTORRotErrVar</td>
<td>Error variances in rotated solution</td>
<td></td>
</tr>
<tr>
<td>FACTORRotErrVarStd</td>
<td>Standardized results for error variances in rotated solution</td>
<td></td>
</tr>
<tr>
<td>FACTORRotLoadings</td>
<td>Rotated factor loadings</td>
<td></td>
</tr>
<tr>
<td>FACTORRotLoadingsStd</td>
<td>Standardized rotated factor loadings</td>
<td></td>
</tr>
<tr>
<td>FACTORRotMat</td>
<td>Rotation matrix</td>
<td></td>
</tr>
<tr>
<td>FACTORScoresRegCoef</td>
<td>Factor scores regression coefficients</td>
<td></td>
</tr>
<tr>
<td>FACTORVariables</td>
<td>Variables in the analysis</td>
<td></td>
</tr>
<tr>
<td>LINEQSAlpha</td>
<td>Estimated intercept vector</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSAlphaInit</td>
<td>Initial intercept vector</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSBeta</td>
<td>Estimated <em>EQSBETA</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSBetaInit</td>
<td>Initial <em>EQSBETA</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSBetaStd</td>
<td>Standardized results for <em>EQSBETA</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSCovExog</td>
<td>Estimated covariances among exogenous variables</td>
<td></td>
</tr>
<tr>
<td>LINEQSCovExogInit</td>
<td>Initial covariances among exogenous variables</td>
<td></td>
</tr>
<tr>
<td>LINEQSCovExogStd</td>
<td>Standardized results for covariances among exogenous variables</td>
<td></td>
</tr>
<tr>
<td>LINEQSEffects</td>
<td>Estimated effects in equations</td>
<td></td>
</tr>
<tr>
<td>LINEQSEffectsStd</td>
<td>Estimated standardized effects in equations</td>
<td></td>
</tr>
<tr>
<td>LINEQSeq</td>
<td>Estimated equations</td>
<td></td>
</tr>
<tr>
<td>LINEQSeqInit</td>
<td>Initial equations</td>
<td></td>
</tr>
<tr>
<td>LINEQSeqStd</td>
<td>Standardized equations</td>
<td></td>
</tr>
<tr>
<td>ODS Table Name</td>
<td>Description</td>
<td>Additional Option</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>LINEQSGamma</td>
<td>Estimated <em>EQSGAMMA</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSGammaInit</td>
<td>Initial <em>EQSGAMMA</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSGammaStd</td>
<td>Standardized results for <em>EQSGAMMA</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSMMeans</td>
<td>Estimated means for exogenous variables</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSMMeansInit</td>
<td>Initial means for exogenous variables</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSNu</td>
<td>Estimated mean vector</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSNuInit</td>
<td>Initial mean vector</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSPhi</td>
<td>Estimated <em>EQSPHI</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSPhiInit</td>
<td>Initial <em>EQSPHI</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSPhiStd</td>
<td>Standardized results for <em>EQSPHI</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>LINEQSVarExog</td>
<td>Estimated variances of exogenous variables</td>
<td></td>
</tr>
<tr>
<td>LINEQSVarExogInit</td>
<td>Initial variances of exogenous variables</td>
<td></td>
</tr>
<tr>
<td>LINEQSVarExogStd</td>
<td>Standardized results for variances of exogenous</td>
<td></td>
</tr>
<tr>
<td>LINEQSVarsExog</td>
<td>Exogenous and endogenous variables</td>
<td></td>
</tr>
<tr>
<td>LISMODAlpha</td>
<td>Estimated <em>ALPHA</em> vector</td>
<td></td>
</tr>
<tr>
<td>LISMODAlphaInit</td>
<td>Initial <em>ALPHA</em> vector</td>
<td></td>
</tr>
<tr>
<td>LISMODBeta</td>
<td>Estimated <em>BETA</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODBetaInit</td>
<td>Initial <em>BETA</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODBetaStd</td>
<td>Standardized <em>BETA</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODGamma</td>
<td>Estimated <em>GAMMA</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODGammaInit</td>
<td>Initial <em>GAMMA</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODGammaStd</td>
<td>Standardized <em>GAMMA</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODKappa</td>
<td>Estimated <em>KAPPA</em> vector</td>
<td></td>
</tr>
<tr>
<td>LISMODKappaInit</td>
<td>Initial <em>KAPPA</em> vector</td>
<td></td>
</tr>
<tr>
<td>LISMODLambdaX</td>
<td>Estimated <em>LAMBDAX</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODLambdaXInit</td>
<td>Initial <em>LAMBDAX</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODLambdaXStd</td>
<td>Standardized <em>LAMBDAX</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODlambdaY</td>
<td>Estimated <em>LAMBDA</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODlambdaYInit</td>
<td>Initial <em>LAMBDA</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODlambdaYStd</td>
<td>Standardized <em>LAMBDA</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODNuX</td>
<td>Estimated <em>NUX</em> vector</td>
<td></td>
</tr>
<tr>
<td>LISMODNuXInit</td>
<td>Initial <em>NUX</em> vector</td>
<td></td>
</tr>
<tr>
<td>LISMODNuY</td>
<td>Estimated <em>NUY</em> vector</td>
<td></td>
</tr>
<tr>
<td>LISMODNuYInit</td>
<td>Initial <em>NUY</em> vector</td>
<td></td>
</tr>
<tr>
<td>LISMODPhi</td>
<td>Estimated <em>PHI</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODPhiInit</td>
<td>Initial <em>PHI</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODPhiStd</td>
<td>Standardized <em>PHI</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODPsi</td>
<td>Estimated <em>PSI</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODPsiInit</td>
<td>Initial <em>PSI</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODPsiStd</td>
<td>Standardized <em>PSI</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODThetaX</td>
<td>Estimated <em>THETAX</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODThetaXInit</td>
<td>Initial <em>THETAX</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODThetaXStd</td>
<td>Standardized <em>THETAX</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODThetaY</td>
<td>Estimated <em>THETAY</em> matrix</td>
<td></td>
</tr>
<tr>
<td>ODS Table Name</td>
<td>Description</td>
<td>Additional Option</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>LISMODThetaYInit</td>
<td>Initial <em>THETAY</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMODThetaYStd</td>
<td>Standardized <em>THETAY</em> matrix</td>
<td></td>
</tr>
<tr>
<td>LISMOVariables</td>
<td>Variables in the model</td>
<td></td>
</tr>
<tr>
<td>MSTRUCTCov</td>
<td>Estimated <em>COV</em> matrix</td>
<td></td>
</tr>
<tr>
<td>MSTRUCTCovInit</td>
<td>Initial <em>COV</em> matrix</td>
<td></td>
</tr>
<tr>
<td>MSTRUCTCovStd</td>
<td>Standardized <em>COV</em> matrix</td>
<td></td>
</tr>
<tr>
<td>MSTRUCTMean</td>
<td>Estimated <em>MEAN</em> vector</td>
<td></td>
</tr>
<tr>
<td>MSTRUCTMeanInit</td>
<td>Initial <em>MEAN</em> vector</td>
<td></td>
</tr>
<tr>
<td>MSTRUCTVariables</td>
<td>Variables in the model</td>
<td></td>
</tr>
<tr>
<td>PATHCovErrors</td>
<td>Estimated error covariances</td>
<td></td>
</tr>
<tr>
<td>PATHCovErrorsInit</td>
<td>Initial error covariances</td>
<td></td>
</tr>
<tr>
<td>PATHCovErrorsStd</td>
<td>Standardized error covariances</td>
<td></td>
</tr>
<tr>
<td>PATHCovVarErr</td>
<td>Estimated covariances between exogenous variables and errors</td>
<td></td>
</tr>
<tr>
<td>PATHCovVarErrInit</td>
<td>Initial covariances between exogenous variables and errors</td>
<td></td>
</tr>
<tr>
<td>PATHCovVarErrStd</td>
<td>Standardized results for covariances between exogenous variables and errors</td>
<td></td>
</tr>
<tr>
<td>PATHCovVars</td>
<td>Estimated covariances among exogenous variables</td>
<td></td>
</tr>
<tr>
<td>PATHCovVarsInit</td>
<td>Initial covariances among exogenous variables</td>
<td></td>
</tr>
<tr>
<td>PATHCovVarsStd</td>
<td>Standardized results for covariances among exogenous variables</td>
<td></td>
</tr>
<tr>
<td>PATHList</td>
<td>Estimated path list</td>
<td></td>
</tr>
<tr>
<td>PATHListInit</td>
<td>Initial path list</td>
<td></td>
</tr>
<tr>
<td>PATHListStd</td>
<td>Standardized path list</td>
<td></td>
</tr>
<tr>
<td>PATHMeansIntercepts</td>
<td>Estimated intercepts</td>
<td></td>
</tr>
<tr>
<td>PATHMeansInterceptsIni</td>
<td>Initial intercepts</td>
<td></td>
</tr>
<tr>
<td>PATHVariables</td>
<td>Exogenous and endogenous variables</td>
<td></td>
</tr>
<tr>
<td>PATHVarParms</td>
<td>Estimated variances or error variances</td>
<td></td>
</tr>
<tr>
<td>PATHVarParmsInit</td>
<td>Initial variances or error variances</td>
<td></td>
</tr>
<tr>
<td>PATHVarParmsStd</td>
<td>Standardized results for variances or error variances</td>
<td></td>
</tr>
<tr>
<td>RAMAMat</td>
<td>Estimated <em>A</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>RAMAMatInit</td>
<td>Initial <em>A</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>RAMAMatStd</td>
<td>Standardized results of <em>A</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>RAMList</td>
<td>List of RAM estimates</td>
<td></td>
</tr>
<tr>
<td>RAMListInit</td>
<td>List of initial RAM estimates</td>
<td></td>
</tr>
<tr>
<td>RAMListStd</td>
<td>Standardized results for RAM estimates</td>
<td></td>
</tr>
<tr>
<td>RAMPMat</td>
<td>Estimated <em>P</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>RAMPMatInit</td>
<td>Initial <em>P</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>RAMPMatStd</td>
<td>Standardized results of <em>P</em> matrix</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>RAMVariables</td>
<td>Exogenous and endogenous variables</td>
<td></td>
</tr>
<tr>
<td>RAMWVec</td>
<td>Estimated mean and intercept vector</td>
<td>PRIMAT</td>
</tr>
<tr>
<td>RAMWVecInit</td>
<td>Initial mean and intercept vector</td>
<td>PRIMAT</td>
</tr>
</tbody>
</table>
ODS Tables for Supplementary Model Analysis

These ODS tables are model-oriented. They are printed for each model in your analysis. To display these ODS tables, you can set some specialized options in either the PROC CALIS or MODEL statement. If the specialized options are used in the PROC CALIS statement, they apply to all models. If the specialized options are used in the MODEL statement, they apply to the associated model only. For some of these ODS tables, certain specialized statements for the model might also enable the printing. Alternatively, you can use the global printing options in the PROC CALIS statement to print these ODS tables. Either a specialized option (or statement) or a global printing option is sufficient to print a particular ODS table.

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Specialized Option or Statement</th>
<th>Global Display Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determination</td>
<td>Coefficients of determination</td>
<td>PDETERM</td>
<td>Default</td>
</tr>
<tr>
<td>DirectEffects</td>
<td>Direct effects</td>
<td>TOTEFF</td>
<td>PRINT</td>
</tr>
<tr>
<td>DirectEffectsStd</td>
<td>Standardized direct effects</td>
<td>TOTEFF;3</td>
<td>PRINT</td>
</tr>
<tr>
<td>EffectsOf</td>
<td>Effects of the listed variables</td>
<td>EFFPART</td>
<td>PRINT</td>
</tr>
<tr>
<td>EffectsOn</td>
<td>Effects on the listed variables</td>
<td>EFFPART</td>
<td>PRINT</td>
</tr>
<tr>
<td>IndirectEffects</td>
<td>Indirect effects</td>
<td>TOTEFF</td>
<td>PRINT</td>
</tr>
<tr>
<td>IndirectEffectsStd</td>
<td>Standardized indirect effects</td>
<td>TOTEFF;3</td>
<td>PRINT</td>
</tr>
<tr>
<td>LatentScoresRegCoef</td>
<td>Latent variable scores regression coefficients</td>
<td>PLATCOV</td>
<td>PRINT</td>
</tr>
<tr>
<td>PredCovLatent</td>
<td>Predicted covariances among latent variables</td>
<td>PLATCOV</td>
<td>PRINT</td>
</tr>
<tr>
<td>PredCovLatMan</td>
<td>Predicted covariances between latent and manifest variables</td>
<td>PLATCOV</td>
<td>PRINT</td>
</tr>
<tr>
<td>PredMeanLatent</td>
<td>Predicted means of latent variables</td>
<td>PLATCOV</td>
<td>PRINT</td>
</tr>
<tr>
<td>SqMultCorr</td>
<td>Squared multiple correlations</td>
<td>PESTIM</td>
<td>PSHORT</td>
</tr>
<tr>
<td>Stability</td>
<td>Stability coefficient of reciprocal causation</td>
<td>PDETERM, DETERM</td>
<td>Default</td>
</tr>
<tr>
<td>StdEffectsOf</td>
<td>Standardized effects of the listed variables</td>
<td>EFFPART;2,3</td>
<td>PSHORT</td>
</tr>
<tr>
<td>StdEffectsOn</td>
<td>Standardized effects on the listed variables</td>
<td>EFFPART;2,3</td>
<td>PSHORT</td>
</tr>
<tr>
<td>TotalEffects</td>
<td>Total effects</td>
<td>TOTEFF</td>
<td>PRINT</td>
</tr>
<tr>
<td>TotalEffectsStd</td>
<td>Standardized total effects</td>
<td>TOTEFF;3</td>
<td>PRINT</td>
</tr>
</tbody>
</table>

1. This refers to the TOTEFF or EFFPART option in the PROC CALIS or MODEL statement.
2. This refers to the EFFPART statement specifications.
3. NOSTAND option must not be specified in the MODEL or PROC CALIS statement.
4. PDETERM is an option specified in the PROC CALIS or MODEL statement, while DETERM is a statement name.

ODS Tables for Model Modification Indices

To print the ODS tables for model modification indices, you can use the MODIFICATION option in either the PROC CALIS or MODEL statement. When this option is set in the PROC CALIS statement, it applies
to all models. When this option is set in the MODEL statement, it applies to the associated model only. Alternatively, you can also use the PALL option in the PROC CALIS statement to print these ODS tables.

If the NOMOD option is set in the PROC CALIS statement, these ODS tables are not printed for all models, unless the MODIFICATION is respecified in the individual MODEL statements. If the NOMOD option is set in the MODEL statement, then the ODS tables for modification do not print for the associated model.

For convenience in presentation, three different classes of ODS tables for model modifications are described in the following. First, ODS tables for ranking of LM indices are the default printing when the MODIFICATION option is specified. Second, ODS tables for LM indices in matrix forms require an additional option to print. Last, ODS tables for other modification indices, including the Wald test indices, require specific data-analytic conditions to print. While the first two classes of ODS tables are model-oriented (that is, each model has its own sets of output), the third one is not.

**ODS Table Names for Ranking of LM indices**

Rankings of the LM statistics in different regions of parameter space are the default printing format when you specify the MODIFICATION option in the PROC CALIS or MODEL statement. You can also turn off these default printing by the NODEFAULT option in the LMTESTS statement for models. If you want to print matrices of LM test statistics rather than the rankings of LM test statistics, you can use the NORANK or MAXRANK=0 option in the LMTESTS statement.

These ODS tables for ranking LM statistics are specific to the types of modeling languages used. This is noted in the last column of the following table.

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMRankCosanMatrix</td>
<td>Any COSAN model matrix</td>
<td>COSAN</td>
</tr>
<tr>
<td>LMRankCov</td>
<td>Covariances among variables</td>
<td>MSTRUCT</td>
</tr>
<tr>
<td>LMRankCovErr</td>
<td>Covariances among errors</td>
<td>LINEQS</td>
</tr>
<tr>
<td>LMRankCovErrofVar</td>
<td>Covariances among errors of variables</td>
<td>PATH</td>
</tr>
<tr>
<td>LMRankCovExog</td>
<td>Covariances among existing exogenous variables</td>
<td>LINEQS or PATH</td>
</tr>
<tr>
<td>LMRankCovFactors</td>
<td>Covariance among factors</td>
<td>FACTOR</td>
</tr>
<tr>
<td>LMRankCustomSet</td>
<td>Customized sets of parameters defined in LMTESTS statements</td>
<td>any model</td>
</tr>
<tr>
<td>LMRankErrorVar</td>
<td>Error variances</td>
<td>FACTOR</td>
</tr>
<tr>
<td>LMRankFactMeans</td>
<td>Factor means</td>
<td>FACTOR</td>
</tr>
<tr>
<td>LMRankIntercepts</td>
<td>Intercepts</td>
<td>FACTOR, LINEQS, or PATH</td>
</tr>
<tr>
<td>LMRankLisAlpha</td>
<td>LISMOD <em>ALPHA</em></td>
<td>LISMOD</td>
</tr>
<tr>
<td>LMRankLisBeta</td>
<td>LISMOD <em>BETA</em></td>
<td>LISMOD</td>
</tr>
<tr>
<td>LMRankLisGamma</td>
<td>LISMOD <em>GAMMA</em></td>
<td>LISMOD</td>
</tr>
<tr>
<td>LMRankLisKappa</td>
<td>LISMOD <em>KAPPA</em></td>
<td>LISMOD</td>
</tr>
<tr>
<td>LMRankLisLambdaX</td>
<td>LISMOD <em>LAMBDAX</em></td>
<td>LISMOD</td>
</tr>
<tr>
<td>LMRankLisLambdaY</td>
<td>LISMOD <em>LAMBDAY</em></td>
<td>LISMOD</td>
</tr>
<tr>
<td>LMRankLisNuX</td>
<td>LISMOD <em>NUX</em></td>
<td>LISMOD</td>
</tr>
<tr>
<td>LMRankLisNuY</td>
<td>LISMOD <em>NUY</em></td>
<td>LISMOD</td>
</tr>
<tr>
<td>LMRankLisPhi</td>
<td>LISMOD <em>PHI</em></td>
<td>LISMOD</td>
</tr>
<tr>
<td>LMRankLisPsi</td>
<td>LISMOD <em>PSI</em></td>
<td>LISMOD</td>
</tr>
<tr>
<td>LMRankLisThetaX</td>
<td>LISMOD <em>THETAX</em></td>
<td>LISMOD</td>
</tr>
<tr>
<td>ODS Table Name</td>
<td>Description</td>
<td>Model</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>LMRankLisThetaY</td>
<td>LISMOD <em>THETAY</em></td>
<td>LISMOD</td>
</tr>
<tr>
<td>LMRankLoadings</td>
<td>Factor loadings</td>
<td>FACTOR</td>
</tr>
<tr>
<td>LMRankMeans</td>
<td>Means of existing variables</td>
<td>LINEQS, MSTRUCT, or PATH</td>
</tr>
<tr>
<td>LMRankPaths</td>
<td>All possible paths in the model</td>
<td>PATH</td>
</tr>
<tr>
<td>LMRankPathsFromEndo</td>
<td>Paths from existing endogenous variables</td>
<td>LINEQS</td>
</tr>
<tr>
<td>LMRankPathsFromExog</td>
<td>Paths from existing exogenous variables</td>
<td>LINEQS</td>
</tr>
<tr>
<td>LMRankPathsNewEndo</td>
<td>Paths to existing exogenous variables</td>
<td>LINEQS</td>
</tr>
<tr>
<td>LMRankRamA</td>
<td><em>RAMA</em> matrix</td>
<td>RAM</td>
</tr>
<tr>
<td>LMRankRamAlpha</td>
<td><em>RAMALPHA</em> matrix</td>
<td>RAM</td>
</tr>
<tr>
<td>LMRankRamNu</td>
<td><em>RAMNU</em> matrix</td>
<td>RAM</td>
</tr>
<tr>
<td>LMRankRamP11</td>
<td><em>RAMP11</em> matrix</td>
<td>RAM</td>
</tr>
<tr>
<td>LMRankRamP22</td>
<td><em>RAMP22</em> matrix</td>
<td>RAM</td>
</tr>
</tbody>
</table>

**ODS Table Names for Lagrange Multiplier Tests in Matrix Form**

To print matrices of LM test indices for a model, you must also use the LMMAT option in the LMTESTS statement for the model. Some of these matrices are printed by default, while others are printed only when certain regions of parameter are specified in the LM test sets. In the following tables, the ODS table names for LM test statistics in matrix form are listed for each model type.

**The COSAN Model**

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Selected Region in Test Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMCosanMatrix</td>
<td>Any COSAN model matrix</td>
<td>(default)</td>
</tr>
</tbody>
</table>

**The FACTOR Model**

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Selected Region in Test Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMFactErrv</td>
<td>Vector of error variances</td>
<td>FACTERRV (default)</td>
</tr>
<tr>
<td>LMFactFcov</td>
<td>Factor covariance matrix</td>
<td>FACTFCOV (default)</td>
</tr>
<tr>
<td>LMFactInte</td>
<td>Intercept vector</td>
<td>FACTINTE (default)</td>
</tr>
<tr>
<td>LMFactLoad</td>
<td>Factor loading matrix</td>
<td>FACTLOAD (default)</td>
</tr>
<tr>
<td>LMFactMean</td>
<td>Factor mean vector</td>
<td>FACTMEAN (default)</td>
</tr>
</tbody>
</table>

**The LINEQS Model**

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Selected Regions in Test Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMEqsAlpha</td>
<td><em>EQSALPHA</em> vector</td>
<td><em>EQSALPHA</em> (default)</td>
</tr>
<tr>
<td>LMEqsBeta</td>
<td><em>EQSVALUE</em> matrix</td>
<td><em>EQSVALUE</em> (default)</td>
</tr>
<tr>
<td>LMEqsGammaSub</td>
<td><em>EQSGAMMA</em> matrix, excluding entries with error variables in columns</td>
<td><em>EQSGAMMA</em> (default)</td>
</tr>
</tbody>
</table>
### The CALIS Procedure

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Selected Regions in Test Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMEqsNewDep</td>
<td>New rows for expanding <em>EQSBETA</em> and <em>EQSGAMMA</em> matrices</td>
<td>NEWDEP</td>
</tr>
<tr>
<td>LMEqsNuSub</td>
<td><em>EQSNU</em> vector, excluding fixed zero means for error variables</td>
<td><em>EQSNU</em> (default)</td>
</tr>
<tr>
<td>LMEqsPhi</td>
<td><em>EQSPHI</em> matrix</td>
<td><em>EQSPHI</em> alone or <em>EQSPHI11</em>, <em>EQSPHI21</em> and <em>EQSPHI22</em> together</td>
</tr>
<tr>
<td>LMEqsPhi11</td>
<td>Upper left portion (exogenous variances and covariances) of the <em>EQSPHI</em> matrix</td>
<td><em>EQSPHI11</em> (default)</td>
</tr>
<tr>
<td>LMEqsPhi21</td>
<td>Lower left portion (error variances and covariances) of the <em>EQSPHI</em> matrix</td>
<td><em>EQSPHI21</em></td>
</tr>
<tr>
<td>LMEqsPhi22</td>
<td>Lower right portion (error variances and covariances) of the <em>EQSPHI</em> matrix</td>
<td><em>EQSPHI22</em> (default)</td>
</tr>
</tbody>
</table>

### The LISMOD Model

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Selected Regions in Test Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMLisAlpha</td>
<td>LISMOD <em>ALPHA</em> vector</td>
<td><em>ALPHA</em> (default)</td>
</tr>
<tr>
<td>LMLisBeta</td>
<td>LISMOD <em>BETA</em> matrix</td>
<td><em>BETA</em> (default)</td>
</tr>
<tr>
<td>LMLisGamma</td>
<td>LISMOD <em>GAMMA</em> matrix</td>
<td><em>GAMMA</em> (default)</td>
</tr>
<tr>
<td>LMLisKappa</td>
<td>LISMOD <em>KAPPA</em> vector</td>
<td><em>KAPPA</em> (default)</td>
</tr>
<tr>
<td>LMLisLambdaX</td>
<td>LISMOD <em>LAMBDAX</em> matrix</td>
<td><em>LAMBDAX</em> (default) or <em>LAMBDA</em></td>
</tr>
<tr>
<td>LMLisLambdaY</td>
<td>LISMOD <em>LAMBDAY</em> matrix</td>
<td><em>LAMBDAY</em> (default) or <em>LAMBDA</em></td>
</tr>
<tr>
<td>LMLisNuX</td>
<td>LISMOD <em>NUX</em> vector</td>
<td><em>NUX</em> (default) or <em>NU</em></td>
</tr>
<tr>
<td>LMLisNuY</td>
<td>LISMOD <em>NUY</em> vector</td>
<td><em>NUY</em> (default) or <em>NU</em></td>
</tr>
<tr>
<td>LMLisPhi</td>
<td>LISMOD <em>PHI</em> matrix</td>
<td><em>PHI</em> (default)</td>
</tr>
<tr>
<td>LMLisPsi</td>
<td>LISMOD <em>PSI</em> matrix</td>
<td><em>PSI</em> (default)</td>
</tr>
<tr>
<td>LMLisThetaX</td>
<td>LISMOD <em>THETAX</em> matrix</td>
<td><em>THETAX</em> (default) or <em>THETA</em></td>
</tr>
<tr>
<td>LMLisThetaY</td>
<td>LISMOD <em>THETAY</em> matrix</td>
<td><em>THETAY</em> (default) or <em>THETA</em></td>
</tr>
</tbody>
</table>

### The MSTRUCT Model

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Selected Regions in Test Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMMstructCov</td>
<td>Covariance matrix</td>
<td>MSTRUCTCOV (default) or <em>COV</em></td>
</tr>
<tr>
<td>LMMstructMean</td>
<td>Mean vector</td>
<td>MSTRUCTMEAN (default) or <em>MEAN</em></td>
</tr>
</tbody>
</table>
### The PATH Model

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Selected Regions in Test Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMRamA</td>
<td><em>RAMA</em> matrix</td>
<td>ARROWS or <em>RAMA</em> (default)</td>
</tr>
<tr>
<td>LMRamALeft</td>
<td>Left portion of the <em>RAMA</em> matrix</td>
<td><em>RAMA_LEFT</em> alone or <em>RAMBETA</em> and <em>RAMA_LL</em> together</td>
</tr>
<tr>
<td>LMRamALL</td>
<td>Lower left portion of the <em>RAMA</em> matrix</td>
<td><em>RAMA_LL</em></td>
</tr>
<tr>
<td>LMRamALower</td>
<td>Lower portion of the <em>RAMA</em> matrix</td>
<td>NEWENDO or <em>RAMA_LOWER</em> or <em>RAMA_LL</em> and <em>RAMA_LR</em> together</td>
</tr>
<tr>
<td>LMRamALR</td>
<td>Lower right portion of the <em>RAMA</em> matrix</td>
<td><em>RAMA_LR</em></td>
</tr>
<tr>
<td>LMRamARight</td>
<td>Right portion of the <em>RAMA</em> matrix</td>
<td><em>RAMA_RIGHT</em></td>
</tr>
<tr>
<td>LMRamAUpper</td>
<td>Upper portion of the <em>RAMA</em> matrix</td>
<td><em>RAMA_UPPER</em> alone or <em>RAMBETA</em> and <em>RAMGAMMA</em> together</td>
</tr>
<tr>
<td>LMRamAlpha</td>
<td><em>RAMALPHA</em> matrix</td>
<td>INTERCEPTS (default)</td>
</tr>
<tr>
<td>LMRamBeta</td>
<td>Upper left portion of the <em>RAMA</em> matrix</td>
<td><em>RAMBETA</em></td>
</tr>
<tr>
<td>LMRamGamma</td>
<td>Upper right portion of the <em>RAMA</em> matrix</td>
<td><em>RAMGAMMA</em></td>
</tr>
<tr>
<td>LMRamNu</td>
<td><em>RAMNU</em> matrix</td>
<td>MEANS (default)</td>
</tr>
<tr>
<td>LMRamP</td>
<td><em>RAMP</em> matrix</td>
<td><em>RAMP</em> alone or <em>RAMP11</em>, <em>RAMP21</em>, and <em>RAMP22</em> together</td>
</tr>
<tr>
<td>LMRamP11</td>
<td>Upper left portion of the <em>RAMP</em> matrix</td>
<td>COVERR (default)</td>
</tr>
<tr>
<td>LMRamP21</td>
<td>Lower left portion of the <em>RAMP</em> matrix</td>
<td>COVEXOGERR</td>
</tr>
<tr>
<td>LMRamP22</td>
<td>Lower right portion of the <em>RAMP</em> matrix</td>
<td>COVEXOG (default)</td>
</tr>
<tr>
<td>LMRamW</td>
<td><em>RAMW</em> matrix</td>
<td>FIRSTMOMENTS alone or MEANS and INTERCEPTS together</td>
</tr>
</tbody>
</table>

### The RAM Model

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Selected Regions in Test Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMRamA</td>
<td><em>RAMA</em> matrix</td>
<td><em>RAMA</em> (default)</td>
</tr>
<tr>
<td>LMRamALeft</td>
<td>Left portion of the <em>RAMA</em> matrix</td>
<td><em>RAMA_LEFT</em> alone or <em>RAMBETA</em> and <em>RAMA_LL</em> together</td>
</tr>
<tr>
<td>LMRamALL</td>
<td>Lower left portion of the <em>RAMA</em> matrix</td>
<td><em>RAMA_LL</em></td>
</tr>
<tr>
<td>LMRamALower</td>
<td>Lower portion of the <em>RAMA</em> matrix</td>
<td>NEWENDO or <em>RAMA_LOWER</em> or <em>RAMA_LL</em> and <em>RAMA_LR</em> together</td>
</tr>
<tr>
<td>LMRamALR</td>
<td>Lower right portion of the <em>RAMA</em> matrix</td>
<td><em>RAMA_LR</em></td>
</tr>
<tr>
<td>LMRamARight</td>
<td>Right portion of the <em>RAMA</em> matrix</td>
<td><em>RAMA_RIGHT</em></td>
</tr>
<tr>
<td>LMRamAUpper</td>
<td>Upper portion of the <em>RAMA</em> matrix</td>
<td><em>RAMBETA</em> and <em>RAMGAMMA</em></td>
</tr>
</tbody>
</table>
Chapter 29: The CALIS Procedure

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Selected Regions in Test Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMRamAlpha</td>
<td><em>RAMALPHA</em> matrix</td>
<td><em>RAMALPHA</em> (default)</td>
</tr>
<tr>
<td>LMRamBeta</td>
<td>Upper left portion of the <em>RAMA</em> matrix</td>
<td><em>RAMBETA</em></td>
</tr>
<tr>
<td>LMRamGamma</td>
<td>Upper right portion of the <em>RAMA</em> matrix</td>
<td><em>RAMGAMMA</em></td>
</tr>
<tr>
<td>LMRamNu</td>
<td><em>RAMNU</em> matrix</td>
<td><em>RAMNU</em> (default)</td>
</tr>
<tr>
<td>LMRamP</td>
<td><em>RAMP</em> matrix</td>
<td><em>RAMP</em> alone or <em>RAMP11</em>, <em>RAMP21</em>, and <em>RAMP22</em> together</td>
</tr>
<tr>
<td>LMRamP11</td>
<td>Upper left portion of the <em>RAMP</em> matrix</td>
<td><em>RAMP11</em> (default)</td>
</tr>
<tr>
<td>LMRamP21</td>
<td>Lower left portion of the <em>RAMP</em> matrix</td>
<td><em>RAMP21</em></td>
</tr>
<tr>
<td>LMRamP22</td>
<td>Lower right portion of the <em>RAMP</em> matrix</td>
<td><em>RAMP22</em> (default)</td>
</tr>
<tr>
<td>LMRamW</td>
<td><em>RAMW</em> matrix</td>
<td><em>RAMW</em></td>
</tr>
</tbody>
</table>

**ODS Table Names for Other Modification Indices**

The following table shows the ODS tables for the remaining modification indices.

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Additional Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>LagrangeBoundary</td>
<td>LM tests for active boundary constraints</td>
<td>Presence of active boundary constraints</td>
</tr>
<tr>
<td>LagrangeDepParmEquality</td>
<td>LM tests for equality constraints in dependent parameters</td>
<td>Presence of equality constraints in dependent parameters</td>
</tr>
<tr>
<td>LagrangeEquality</td>
<td>LM tests for equality constraints</td>
<td>Presence of equality constraints in independent parameters</td>
</tr>
<tr>
<td>WaldTest</td>
<td>Wald tests for testing existing parameters equaling zeros</td>
<td>At least one insignificant parameter value</td>
</tr>
</tbody>
</table>

**ODS Table for Optimization Control and Results**

To display the ODS tables for optimization control and results, you must specify any of the following global display options in the PROC CALIS statement: PRINT, PALL, or default (that is, NOPRINT is not specified). Also, you must not use the NOPRINT option in the NLOPTIONS statement. For some of these tables, you must also specify additional options, either in the PROC CALIS or the NLOPTIONS statement. Some restrictions might apply. Additional options and restrictions are noted in the last column.
<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Additional Option Required or Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>CovParm</td>
<td>Covariances of parameters</td>
<td>PCOVES¹ or PALL², restriction³</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td></td>
</tr>
<tr>
<td>DependParmsResults</td>
<td>Final dependent parameter estimates</td>
<td>Restriction⁴</td>
</tr>
<tr>
<td>DependParmsStart</td>
<td>Initial dependent parameter estimates</td>
<td>Restriction⁴</td>
</tr>
<tr>
<td>Information</td>
<td>Information matrix</td>
<td>PCOVES¹ or PALL², restriction³</td>
</tr>
<tr>
<td>InitEstMethods</td>
<td>Initial estimation methods</td>
<td>PALL²</td>
</tr>
<tr>
<td>InputOptions</td>
<td>Optimization options</td>
<td></td>
</tr>
<tr>
<td>IterHist</td>
<td>Iteration history</td>
<td></td>
</tr>
<tr>
<td>IterStart</td>
<td>Iteration start</td>
<td></td>
</tr>
<tr>
<td>IterStop</td>
<td>Iteration stop</td>
<td></td>
</tr>
<tr>
<td>Lagrange</td>
<td>First and second order Lagrange multipliers</td>
<td>PALL²</td>
</tr>
<tr>
<td>LinCon</td>
<td>Linear constraints</td>
<td>PALL², restriction⁵</td>
</tr>
<tr>
<td>LinConDel</td>
<td>Deleted constraints</td>
<td>PALL², restriction⁵</td>
</tr>
<tr>
<td>LinConSol</td>
<td>Linear constraints evaluated at solution</td>
<td>PALL², restriction⁵</td>
</tr>
<tr>
<td>LinDep</td>
<td>Linear dependencies of parameter estimates</td>
<td>Restriction⁶</td>
</tr>
<tr>
<td>ParameterEstimatesResults</td>
<td>Final estimates</td>
<td></td>
</tr>
<tr>
<td>ParameterEstimatesStart</td>
<td>Initial estimates</td>
<td></td>
</tr>
<tr>
<td>ProblemDescription</td>
<td>Problem description</td>
<td></td>
</tr>
<tr>
<td>ProjGrad</td>
<td>Projected gradient</td>
<td>PALL²</td>
</tr>
</tbody>
</table>

1. PCOVES option is specified in the PROC CALIS statement.
2. PALL option is specified in the NLOPTIONS statement.
3. Estimation method must not be ULS or DWLS.
4. Existence of dependent parameters.
5. Linear equality or boundary constraints are imposed.
ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, “Statistical Graphics Using ODS.”

Before you create graphs, ODS Graphics must be enabled (for example, by specifying the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section “Enabling and Disabling ODS Graphics” on page 606 in Chapter 21, “Statistical Graphics Using ODS.”

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section “A Primer on ODS Statistical Graphics” on page 605 in Chapter 21, “Statistical Graphics Using ODS.”

In the following table, ODS graph names and the options to display the graphs are listed.

<table>
<thead>
<tr>
<th>ODS Graph Name</th>
<th>Plot Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution of Residuals in the Moment Matrices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AsymStdResidualHistogram</td>
<td>Asymptotically standardized residuals</td>
<td>PLOTS=RESIDUALS and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESIDUAL=ASYMSTD, METHOD= is not ULS or DWLS</td>
</tr>
<tr>
<td>NormResidualHistogram</td>
<td>Normalized residuals</td>
<td>PLOTS=RESIDUALS and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESIDUAL=NORM</td>
</tr>
<tr>
<td>RawResidualHistogram</td>
<td>Raw residuals</td>
<td>PLOTS=RESIDUALS</td>
</tr>
<tr>
<td>VarStdResidualHistogram</td>
<td>Variance standardized residuals</td>
<td>PLOTS=RESIDUALS and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESIDUAL=VARSTD</td>
</tr>
<tr>
<td>Case-Level Residual Diagnostics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CaseResidualHistogram</td>
<td>Distribution of the residual M-distances</td>
<td>PLOTS=CRESHIST or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CASERESIDUAL</td>
</tr>
<tr>
<td>ResidualByLeverage</td>
<td>Residual M-distances against the leverage M-distances</td>
<td>PLOTS=RESBYLEV or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CASERESIDUAL</td>
</tr>
<tr>
<td>ResidualByPredicted</td>
<td>Residuals against the predicted values of dependent variables</td>
<td>PLOTS=RESBYPRED or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CASERESIDUAL</td>
</tr>
<tr>
<td>ResidualByQuantile</td>
<td>Residual M-distances against the theoretical quantiles</td>
<td>PLOTS=QQ or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CASERESIDUAL</td>
</tr>
<tr>
<td>ResPercentileByExpPercentile</td>
<td>Percentiles of residual M-distances against the theoretical percentiles</td>
<td>PLOTS=PP or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CASERESIDUAL</td>
</tr>
<tr>
<td>Path Diagrams</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PathDiagramInit</td>
<td>Path diagram for initial model specifications</td>
<td>DIAGRAM=INITIAL in the PATHDIAGRAM statement</td>
</tr>
<tr>
<td>PathDiagram</td>
<td>Path diagram for the unstandardized solution</td>
<td>PLOTS=PATHDIAGRAM or PATHDIAGRAM statement</td>
</tr>
<tr>
<td>PathDiagramStand</td>
<td>Path diagram for the standardized solution</td>
<td>DIAGRAM=STANDARD in the PATHDIAGRAM statement</td>
</tr>
<tr>
<td>PathDiagramStructInit</td>
<td>Path diagram for initial structural model specifications</td>
<td>DIAGRAM=INITIAL and STRUCTURAL in the PATHDIAGRAM statement</td>
</tr>
</tbody>
</table>
Table 29.21  continued

<table>
<thead>
<tr>
<th>ODS Graph Name</th>
<th>Plot Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>PathDiagramStruct</td>
<td>Path diagram for the unstandardized solution of the</td>
<td>STRUCTURAL in the PATHDIAGRAM</td>
</tr>
<tr>
<td></td>
<td>structural model</td>
<td>statement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DIAGRAM=STANDARD and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STRUCTURAL in the PATHDIAGRAM</td>
</tr>
<tr>
<td>PathDiagramStructStand</td>
<td>Path diagram for the standardized solution of the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>structural model</td>
<td></td>
</tr>
</tbody>
</table>

Examples: CALIS Procedure

Example 29.1: Estimating Covariances and Correlations

This example shows how you can use PROC CALIS to estimate the covariances and correlations of the variables in your data set. Estimating the covariances introduces you to the most basic form of covariance structures—a saturated model with all variances and covariances as parameters in the model. To fit such a saturated model when there is no need to specify the functional relationships among the variables, you can use the MSTRUCT modeling language of PROC CALIS.

The following data set contains four variables q1–q4 for the quarterly sales (in millions) of a company. The 14 observations represent 14 retail locations in the country. The input data set is shown in the following DATA step:

```latex
data sales;
  input q1 q2 q3 q4;
datalines;
  1.03  1.54  1.11  2.22
  1.23  1.43  1.65  2.12
  3.24  2.21  2.31  5.15
  1.23  2.35  2.21  7.17
  .98  2.13  1.76  2.38
  1.02  2.05  3.15  4.28
  1.54  1.99  1.77  2.00
  1.76  1.79  2.28  3.18
  1.11  3.41  2.20  3.21
  1.32  2.32  4.32  4.78
  1.22  1.81  1.51  3.15
  1.11  2.15  2.45  6.17
  1.01  2.12  1.96  2.08
  1.34  1.74  2.16  3.28
;
```

Use the following PROC CALIS specification to estimate a saturated covariance structure model with all variances and covariances as parameters:
In the PROC CALIS statement, specify the data set with the DATA= option. Use the PCORR option to display the observed and predicted covariance matrix. Next, use the MSTRUCT statement to fit a covariance matrix of the variables that are provided in the VAR= option. Without further specifications such as the MATRIX statement, PROC CALIS assumes all elements in the covariance matrix are model parameters. Hence, this is a saturated model.

Output 29.1.1 shows the modeling information. Information about the model is displayed: the name and location of the data set, the number of data records read and used, and the number of observations in the analysis. The number of data records read is the actual number of records (or observations) that PROC CALIS processes from the data set. The number of data records used might or might not be the same as the actual number of records read from the data set. For example, records with missing values are read but not used in the analysis for the default maximum likelihood (ML) method. The number of observations refers to the \( N \) used for testing statistical significance and model fit. This number might or might not be the same as the number of records used for at least two reasons. First, if you use a frequency variable in the FREQ statement, the number of observations used is a weighted sum of the number of records, with the frequency variable being the weight. Second, if you use the NOBS= option in the PROC CALIS statement, you can override the number of observations that are used in the analysis. Because the current data set does not have any missing data and there are no frequency variables or an NOBS= option specified, these three numbers are all 14.

The model type is MSTRUCT because you use the MSTRUCT statement to define your model. The analysis type is covariances, which is the default. Output 29.1.1 then shows the four variables in the covariance structure model.

**Output 29.1.1** \[Modeling Information of the Saturated Covariance Structure Model for the Sales Data\]

---

**Estimating the Covariance Matrix by the MSTRUCT Modeling Language**

**The CALIS Procedure**

**Covariance Structure Analysis: Model and Initial Values**

<table>
<thead>
<tr>
<th>Modeling Information</th>
<th>Maximum Likelihood Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set</td>
<td>WORK.SALES</td>
</tr>
<tr>
<td>N Records Read</td>
<td>14</td>
</tr>
<tr>
<td>N Records Used</td>
<td>14</td>
</tr>
<tr>
<td>N Obs</td>
<td>14</td>
</tr>
<tr>
<td>Model Type</td>
<td>MSTRUCT</td>
</tr>
<tr>
<td>Analysis</td>
<td>Covariances</td>
</tr>
</tbody>
</table>

**Variables in the Model**

\[q_1 \quad q_2 \quad q_3 \quad q_4\]

**Number of Variables = 4**
Output 29.1.2 shows the initial covariance structure model for these four variables. All lower triangular elements (including the diagonal elements) of the covariance matrix are parameters in the model. PROC CALIS generates the names for these parameters: _Add01–_Add10. Because the covariance matrix is symmetric, all upper triangular elements of the matrix are redundant. The initial estimates for covariance are denoted by missing values no initial values were specified.

**Output 29.1.2  Initial Saturated Covariance Structure Model for the Sales Data**

<table>
<thead>
<tr>
<th>Initial MSTRUCT <em>COV</em> Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
</tr>
<tr>
<td>q1</td>
</tr>
<tr>
<td>q2</td>
</tr>
<tr>
<td>q3</td>
</tr>
<tr>
<td>q4</td>
</tr>
</tbody>
</table>

The PCORR option in the PROC CALIS statement displays the sample covariance matrix in Output 29.1.3. By default, PROC CALIS computes the unbiased sample covariance matrix (with variance divisor equal to N – 1) and uses it for the covariance structure analysis.

**Output 29.1.3  Sample Covariance Matrix for the Sales Data**

<table>
<thead>
<tr>
<th>Covariance Matrix (DF = 13)</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
</tr>
<tr>
<td>q1</td>
</tr>
<tr>
<td>q2</td>
</tr>
<tr>
<td>q3</td>
</tr>
<tr>
<td>q4</td>
</tr>
</tbody>
</table>

The fit summary and the fitted covariance matrix are shown in Output 29.1.4 and Output 29.1.5, respectively.

**Output 29.1.4  Fit Summary of the Saturated Covariance Structure Model for the Sales Data**

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
</tbody>
</table>
In Output 29.1.4, the model fit chi-square is 0 (df = 0). The p-value cannot be computed because the degrees of freedom is zero. This fit is perfect because the model is saturated.

Output 29.1.5 shows the fitted covariance matrix, along with standard error estimates and t values in each cell. The variance and covariance estimates match exactly those of the sample covariance matrix shown in Output 29.1.3.

A common practice for determining statistical significance for estimates in structural equation modeling is to require the absolute value of t to be greater than 1.96, which is the critical value of a standard normal variate at $\alpha=0.05$. While all diagonal elements in Output 29.1.5 show statistical significance, all off-diagonal elements are not significantly different from zero. The t values for these elements range from 0.002 to 1.601.

Output 29.1.6 shows the standardized estimates of the variance and covariance elements. This is also the correlation matrix under the MSTRUCT model. Standard error estimates and t values are computed with the correlation estimates. Note that because the diagonal element values are fixed at 1, no standard errors or t values are shown.
Sometimes researchers do not need to estimate the standard errors that are in their models. You can suppress the standard error and $t$ value computations by using the NOSE option in the PROC CALIS statement:

```
proc calis data=sales nose;
   mstruct var=q1-q4;
run;
```

Output 29.1.7 shows the fitted covariance matrix with the NOSE option. These values are exactly the same as in the sample covariance matrix shown in Output 29.1.3.

**Output 29.1.7** Fitted Covariance Matrix without Standard Error Estimates for the Sales Data

<table>
<thead>
<tr>
<th></th>
<th>q1</th>
<th>q2</th>
<th>q3</th>
<th>q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>0.00000</td>
<td>0.000717</td>
<td>0.0797</td>
<td>0.2331</td>
</tr>
<tr>
<td></td>
<td>0.2773</td>
<td>0.2756</td>
<td>0.2623</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.002587</td>
<td>0.2892</td>
<td>0.8888</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.9979</td>
<td>0.7724</td>
<td>0.3741</td>
<td></td>
</tr>
<tr>
<td>q2</td>
<td>0.000717</td>
<td>1.0000</td>
<td>0.3428</td>
<td>0.3156</td>
</tr>
<tr>
<td></td>
<td>0.2773</td>
<td>0.2448</td>
<td>0.2497</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.002587</td>
<td>1.4008</td>
<td>1.2640</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.9979</td>
<td>0.1613</td>
<td>0.2062</td>
<td></td>
</tr>
<tr>
<td>q3</td>
<td>0.0797</td>
<td>0.3428</td>
<td>1.0000</td>
<td>0.4957</td>
</tr>
<tr>
<td></td>
<td>0.2756</td>
<td>0.2448</td>
<td>0.2092</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.2892</td>
<td>1.4008</td>
<td>2.3692</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.7724</td>
<td>0.1613</td>
<td>0.0178</td>
<td></td>
</tr>
<tr>
<td>q4</td>
<td>0.2331</td>
<td>0.3156</td>
<td>0.4957</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>0.2623</td>
<td>0.2497</td>
<td>0.2092</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.8888</td>
<td>1.2640</td>
<td>2.3692</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.3741</td>
<td>0.2062</td>
<td>0.0178</td>
<td></td>
</tr>
</tbody>
</table>

This example shows a very simple application of PROC CALIS: estimating the covariance matrix with standard error estimates. The covariance structure model is saturated. Several extensions of this very simple model are possible. To estimate the means and covariances simultaneously, see Example 29.2. To fit nonsaturated covariance structure models with certain hypothesized patterns, see Example 29.3 and Example 29.4. To fit structural models with implied covariance structures that are based on specified functional relationships among variables, see Example 29.6.
Example 29.2: Estimating Covariances and Means Simultaneously

This example uses the same data set that is used in Example 29.1 and estimates the means and covariances. Use the MSTRUCT model specification as shown in the following statements:

```
proc calis data=sales meanstr nostand;
   mstruct var=q1-q4;
run;
```

In the PROC CALIS statement, specify the MEANSTR option to request the mean structure analysis in addition to the default covariance structure analysis. If you are not interested in the standardized solution, specify the NOSTAND option in the PROC CALIS statement to suppress computation of the standardized estimates. Without further model specification (such as the MATRIX statement), PROC CALIS assumes a saturated structural model with all means, variances, and covariances as model parameters.

Output 29.2.1 shows the modeling information. With the MEANSTR option specified in the PROC CALIS statement, the current analysis type is Means and Covariances, instead of the default Covariances in Example 29.1.

Output 29.2.1  Modeling Information of the Saturated Mean and Covariance Structure Model for the Sales Data

Saturated Means and Covariance Structures Using MSTRUCT

<table>
<thead>
<tr>
<th>The CALIS Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean and Covariance Structures: Model and Initial Values</td>
</tr>
</tbody>
</table>

- **Modeling Information**
  - Maximum Likelihood Estimation
    - Data Set: WORK.SALES
    - N Records Read: 14
    - N Records Used: 14
    - N Obs: 14
    - Model Type: MSTRUCT
    - Analysis: Means and Covariances

- **Variables in the Model**
  - q1  q2  q3  q4
  - Number of Variables = 4

Output 29.2.2 shows the fit summary of the current model. Again, this is a perfect model fit with 0 chi-square value and 0 degrees of freedom.

Output 29.2.2  Fit Summary of the Saturated Mean and Covariance Structure Model for the Sales Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square: 0.0000</td>
</tr>
<tr>
<td>Chi-Square DF: 0</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square: .</td>
</tr>
</tbody>
</table>
Output 29.2.3 shows the estimates of the means, together with the standard error estimates and the \( t \) values. These estimated means are exactly the same as the sample means, which are not shown here.

**Output 29.2.3** Mean Estimates for the Sales Data

| Variable | Estimate | Standard Error | \( t \) Value | Pr > |t| |
|----------|----------|----------------|---------------|------|---|
| q1       | 1.36714  | 0.16132        | 8.4749        | <.0001 |
| q2       | 2.07429  | 0.13146        | 15.7790       | <.0001 |
| q3       | 2.20286  | 0.21596        | 10.2001       | <.0001 |
| q4       | 3.65500  | 0.45281        | 8.0718        | <.0001 |

Output 29.2.4 shows the variance and covariance estimates. These estimates are exactly the same as the elements in the sample covariance matrix. In addition, these estimates match the estimates in Output 29.1.5 of Example 29.1, where only the covariance structures are analyzed.

**Output 29.2.4** Variance and Covariance Estimates for the Sales Data

<table>
<thead>
<tr>
<th></th>
<th>MSTRUCT <em>COV</em> Matrix: Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>q1</td>
</tr>
<tr>
<td>q1</td>
<td>0.3383</td>
</tr>
<tr>
<td></td>
<td>0.1327</td>
</tr>
<tr>
<td></td>
<td>2.5495</td>
</tr>
<tr>
<td></td>
<td>0.0108</td>
</tr>
<tr>
<td>q2</td>
<td>0.000198</td>
</tr>
<tr>
<td></td>
<td>0.0765</td>
</tr>
<tr>
<td></td>
<td>0.002587</td>
</tr>
<tr>
<td></td>
<td>0.9979</td>
</tr>
<tr>
<td>q3</td>
<td>0.0361</td>
</tr>
<tr>
<td></td>
<td>0.1260</td>
</tr>
<tr>
<td></td>
<td>0.2865</td>
</tr>
<tr>
<td></td>
<td>0.7745</td>
</tr>
<tr>
<td>q4</td>
<td>0.2214</td>
</tr>
<tr>
<td></td>
<td>0.2704</td>
</tr>
<tr>
<td></td>
<td>0.8186</td>
</tr>
<tr>
<td></td>
<td>0.4130</td>
</tr>
</tbody>
</table>

These estimates are essentially the same as the sample means, variances, and covariances. This kind of analysis is much easier using PROC CORR with the NOMISS option. However, the main purpose of Example 29.1 and Example 29.2 is to introduce the MSTRUCT modeling language and some basic but important options in PROC CALIS. You can apply the MSTRUCT modeling language to more sophisticated situations that are beyond the saturated mean and covariance structure models. Example 29.3 and Example 29.4 fit some patterned covariance models that are nonsaturated. Also, options such as NOSE, NOSTAND, and MEANSTR are useful for all modeling languages in PROC CALIS.

**Example 29.3: Testing Uncorrelatedness of Variables**

This example uses the sales data in Example 29.1 and tests the uncorrelatedness of the variables in the model by using the MSTRUCT model specification. With the multivariate normality assumption, this is also the test of independence of the variables. The MATRIX statement defines the parameters in the model.
The uncorrelatedness model assumes that the correlations or covariances among the four variables are zero. Therefore, only the four diagonal elements of the covariance matrix, which represent the variances of the variables, are free parameters in the covariance structure model. To specify these parameters, use the MATRIX statement with the MSTRUCT model specification:

```plaintext
proc calis data=sales;
  mstruct var=q1-q4;
  matrix _cov_ [1,1], [2,2], [3,3], [4,4];
run;
```

Example 29.1 specifies exactly the same MSTRUCT statement for the four variables. The difference here is the addition of the MATRIX statement. Without a MATRIX statement, the MSTRUCT model assumes that all nonredundant elements in the covariance matrix are model parameters. This assumption is not the case in the current specification. The MATRIX statement specification for the covariance matrix (denoted by the _cov_ keyword) specifies four free parameters on the diagonal of the covariance matrix: [1,1], [2,2], [3,3], and [4,4]. All other unspecified elements in the covariance matrix are fixed zeros by default.

The uncorrelatedness model is displayed in the output for the initial model specification. Output 29.3.1 shows that all off-diagonal elements of the covariance matrix are fixed zeros while the diagonal elements are missing and labeled with _Parm1–_Parm4. PROC CALIS generates these parameter names automatically and estimates these four parameters in the analysis.

**Output 29.3.1** Initial Uncorrelatedness Model for the Sales Data

<table>
<thead>
<tr>
<th></th>
<th>q1</th>
<th>q2</th>
<th>q3</th>
<th>q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>.</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>[Parm1]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q2</td>
<td>0</td>
<td>.</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>[Parm2]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q3</td>
<td>0</td>
<td>0</td>
<td>.</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>[Parm3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>[Parm4]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Output 29.3.2 shows the model fit chi-square test of the uncorrelatedness model. The chi-square is 6.528 ($df = 6$, $p = 0.3667$), which is not significant. This means that you fail to reject the uncorrelatedness model. In other words, the data is consistent with the uncorrelatedness model (zero covariances or correlations among the quarterly sales).

**Output 29.3.2** Fit Summary of the Uncorrelatedness Model for the Sales Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>6.528</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>6</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.3667</td>
</tr>
</tbody>
</table>

Output 29.3.3 shows the estimates of the covariance matrix under the uncorrelatedness model, together with standard error estimates and $t$ values. All off-diagonal elements are fixed zeros in the estimation results.
Output 29.3.3  Estimates of Variance under the Uncorrelatedness Model for the Sales Data

<table>
<thead>
<tr>
<th>Estimate/StdErr/t-value/p-value</th>
<th>q1</th>
<th>q2</th>
<th>q3</th>
<th>q4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.3383</td>
<td>0.02247</td>
<td>0.6063</td>
<td>2.6655</td>
</tr>
<tr>
<td></td>
<td>0.1327</td>
<td>0.0881</td>
<td>0.2378</td>
<td>1.0455</td>
</tr>
<tr>
<td></td>
<td>2.5495</td>
<td>2.5495</td>
<td>2.5495</td>
<td>2.5495</td>
</tr>
<tr>
<td></td>
<td>0.0108</td>
<td>0.0108</td>
<td>0.0108</td>
<td>0.0108</td>
</tr>
</tbody>
</table>

This example shows how to specify free parameters in the MSTRUCT model by using the MATRIX statement. To specify the covariance matrix, use the _COV_ keyword in the MATRIX statement. To specify the parameters in the mean structures, you need use an additional MATRIX statement with the _MEAN_ keyword.

Two important notes regarding the MSTRUCT model specification are now in order:

- When you use the MSTRUCT statement without any MATRIX statements, all elements in the covariance matrix are free parameters in the model (for example, see Example 29.1). However, if the MATRIX statement includes at least one free or fixed parameter in the covariance matrix, PROC CALIS assumes that all other unspecified elements in the covariance matrix are fixed zeros (such as the current example).

- Using parameter names in the MATRIX statement specification is optional. In the context of the current example, naming the parameters is optional because there is no need to refer to them anywhere in the specification. PROC CALIS automatically generates unique names for these parameters. Alternatively, you can specify your own parameter names in the MATRIX statement. Naming parameters is not only useful for references, but is also indispensable when you need to constrain model parameters by referring to their names. See Example 29.4 to use parameter names to define a covariance pattern.

Example 29.4: Testing Covariance Patterns

In the test for sphericity, a covariance matrix is hypothesized to be a constant multiple of an identity matrix. That is, the null hypothesis for the population covariance matrix is

$$\Sigma = \sigma^2 I$$
Chapter 29: The CALIS Procedure

where \( \sigma^2 \) is an unknown positive constant and \( I \) is an identity matrix. When this covariance pattern is applied to the sales data in Example 29.1, this hypothesis states that all four variables have the same variance \( \sigma^2 \) and are uncorrelated with each other. This model is more restricted than the uncorrelatedness model in Example 29.3, which requires uncorrelatedness but does not require equal variances. Use the following specification to conduct a sphericity test for the sales data:

```
proc calis data=sales;
  mstruct var=q1-q4;
  matrix _cov_ [1,1] = 4*sigma_sq;
run;
```

This specification is similar to that of Example 29.3. The major difference is the MATRIX statement specification. The current example uses a parameter name sigma_sq to represent the unknown variance parameter \( \sigma^2 \), whereas Example 29.3 specifies only the locations of the four free variance parameters.

The current MATRIX statement specification uses a shorthand notation. On the left-hand side of the equal sign, \([1,1]\) indicates the starting location of the covariance matrix. The matrix entries automatically proceed to \([2,2]\), \([3,3]\) and so on, depending on the length of the parameter list specified on the right-hand sign of the equal sign. For example, if there is just one parameter on the right-hand side, the matrix specification contains only \([1,1]\). In the current example, the specification \(4*\text{sigma}\_sq\) means that \(\text{sigma}\_sq\) appears four times in the specification. As a result, the preceding MATRIX statement specification is equivalent to the following statement:

```
matrix _cov_ [1,1] = sigma_sq,
               [2,2] = sigma_sq,
               [3,3] = sigma_sq,
               [4,4] = sigma_sq;
```

This matrix is what is required by the sphericity test. Use either the expanded notation or the shorthand notation for specifying the covariance pattern. For details about various types of shorthand notation for parameter specifications, see the MATRIX statement.

Output 29.4.1 shows the initial model specification under the test of sphericity. All the diagonal elements are labeled with the same name \(\text{sigma}\_sq\), indicating that they are the same parameter.

```
Output 29.4.1  Covariance Model under Sphericity for the Sales Data

<table>
<thead>
<tr>
<th>q1</th>
<th>q2</th>
<th>q3</th>
<th>q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[sigma_sq]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q2</td>
<td>.</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[sigma_sq]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q3</td>
<td>0</td>
<td>.</td>
<td>0</td>
</tr>
<tr>
<td>[sigma_sq]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q4</td>
<td>0</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td>[sigma_sq]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Output 29.4.2 shows that the model fit chi-square is 31.5951 (\(df = 9, p = 0.0002\)). This means that the covariance pattern under the sphericity hypothesis is not supported.
Output 29.4.2  Fit Summary of the Sphericity Test for the Sales Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>31.5951</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>9</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.0002</td>
</tr>
</tbody>
</table>

Output 29.4.3 shows the estimated covariance matrix under the sphericity hypothesis. The variance estimate for all four diagonal elements is 0.9587 (standard error=0.1880).

Output 29.4.3  Fitted Covariance Matrix under the Sphericity Hypothesis for the Sales Data

\[
\begin{array}{cccc}
q1 & q2 & q3 & q4 \\
q1 & 0.9587 & 0.1880 & 0.0990 & <.0001 \\
    &        & [sigma_sq] &       &       \\
q2 & 0 & 0.9587 & 0.1880 & 0.0990 & <.0001 \\
    &        &          &       & [sigma_sq] & \\
q3 & 0 & 0 & 0.9587 & 0.1880 & 0.0990 & <.0001 \\
    &        &          &       &          & [sigma_sq] & \\
q4 & 0 & 0 & 0 & 0.9587 \\
    &        &          &       &       & 0.1880 & 0.0990 & <.0001 \\
    &        &          &       &       & [sigma_sq] & \\
\end{array}
\]

This example shows how you can specify a simple covariance pattern by using the MATRIX statement. Use the same parameter names to constrain variance parameters that are supposed to be the same under the model. Constraining parameters by using the same parameter names is applicable not only to the MSTRUCT models, but also to more complicated covariance structure models, such as multiple-group modeling (see Example 29.19 and Example 29.21).

The MSTRUCT modeling language is handy when you can directly specify the covariance pattern or structures in your model. However, in most applications of structural equation modeling, it is difficult to specify such direct covariance structures. Instead, the covariance structures are usually implied from the functional relationships among the variables in the model. Using the MSTRUCT modeling language in such a situation is not easy. Fortunately, PROC CALIS supports other modeling languages that enable you to specify the functional relationships among variables. The functional relationships can be in the form of a set of path-like descriptions, a system of linear equations, or parameter specifications in matrices. See Example 29.6 for an introduction to using the PATH modeling language for specifying path models.
Example 29.5: Testing Some Standard Covariance Pattern Hypotheses

In Example 29.3, you test the uncorrelatedness of variables by using the MSTRUCT model specification. In Example 29.4, you test the sphericity of the covariance matrix by using the same model specification technique. In both examples, you need to specify the parameters in the covariance structure model explicitly by using the MATRIX statements.

Some covariance patterns are well-known in multivariate statistics, including the two tests in Example 29.3 and Example 29.4. To facilitate the tests of these “standard” covariance patterns, PROC CALIS provides the COVPATTERN= option to specify those standard covariance patterns more efficiently. With the COVPATTERN= option, you do not need to use the MSTRUCT and MATRIX statements to specify the covariance patterns explicitly. See the COVPATTERN= option for the supported covariance patterns. This example illustrates the use of the COVPATTERN= option.

In Example 29.3, you conduct a test of uncorrelatedness for the four variables in the sales data (see Example 29.1 for the data set). That is, the variables are hypothesized to be uncorrelated and only the four variances on the diagonal of the covariance matrix are population parameters of interest. The null hypothesis for the population covariance matrix is

$$\Sigma = \begin{pmatrix} x & 0 & 0 & 0 \\ 0 & x & 0 & 0 \\ 0 & 0 & x & 0 \\ 0 & 0 & 0 & x \end{pmatrix}$$

where each x represents a distinct parameter (that is, the diagonal elements are not constrained with each other). You can test the diagonal covariance pattern easily by the following specification:

```plaintext
proc calis data=sales covpattern=diag;
run;
```

The COVPATTERN=DIAG option specifies the required diagonal covariance pattern for the test. PROC CALIS then sets up the covariance structures automatically. Output 29.5.1 shows the initial specification of the covariance pattern. As required, only the diagonal elements are parameters in the test and the other elements are fixed to zero. PROC CALIS names the variance parameters automatically—that is, _varparm_1–_varparm_4 are the four parameters for the variances. This is the same pattern as shown in Output 29.3.1 of Example 29.3, although the parameter names are different.

Output 29.5.1 Initial Diagonal Pattern for the Covariance Matrix of the Sales Data

<table>
<thead>
<tr>
<th>Initial MSTRUCT <em>COV</em> Matrix</th>
<th>q1</th>
<th>q2</th>
<th>q3</th>
<th>q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>.</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>q2</td>
<td>0</td>
<td>.</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>q3</td>
<td>0</td>
<td>0</td>
<td>.</td>
<td>0</td>
</tr>
<tr>
<td>q4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.</td>
</tr>
</tbody>
</table>
Example 29.5: Testing Some Standard Covariance Pattern Hypotheses

Output 29.5.2 shows the results of the chi-square test of the diagonal covariance pattern. The chi-square is 5.44 ($df = 6, p = 0.4887$), which is not significant. You fail to reject the null hypothesis of the diagonal covariance pattern in the population.

Output 29.5.2  Fit Summary of the Diagonal Covariance Pattern Test for the Sales Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
</tbody>
</table>

The numerical results shown in Output 29.5.2 are different from those of the same test by using the MSTRUCT model specification, which is shown in Output 29.3.2 of Example 29.3, although you do not reject the null hypothesis in both cases. The reason is that with the use of COVPATTERN= option, PROC CALIS applies the appropriate chi-square correction to the test statistic automatically. In the current example, the chi-square correction due to Bartlett (1950) has been applied. Test results with chi-square corrections are theoretically more accurate.

To obtain the same numerical results as those in Output 29.3.2, you can turn off the chi-square correction by using the CHICORRECT=0 option, as shown in the following specification:

```
proc calis data=sales covpattern=diag chicorrect=0;
run;
```

Output 29.5.3 shows the fit summary results without any chi-square correction. The numerical results match exactly to those shown in Output 29.3.2 of Example 29.3.

Output 29.5.3  Fit Summary of the Diagonal Covariance Pattern Test for the Sales Data: No Chi-Square Correction

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
</tbody>
</table>

Example 29.4 tests the sphericity of the covariance matrix of the same data set. The null hypothesis for the population covariance matrix is

$$\Sigma = \sigma^2 I$$

where $\sigma^2$ is an unknown positive constant and $I$ is an identity matrix. You can use the following specification to test this hypothesis easily:

```
proc calis data=sales covpattern=sigsqi;
run;
```

Output 29.5.4 shows the initial specification of the covariance pattern. As required, the diagonal elements are all the same parameter named _varparm, and all the off-diagonal elements are fixed to zero. This is the same pattern as shown in Output 29.4.1 of Example 29.4.
**Output 29.5.4** Initial Covariance Pattern for the Sphericity Test on the Sales Data

<table>
<thead>
<tr>
<th></th>
<th>q1</th>
<th>q2</th>
<th>q3</th>
<th>q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>.</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>[_varpam]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q2</td>
<td>0</td>
<td>.</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[_varpam]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>q3</td>
<td>0</td>
<td>0</td>
<td>.</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[_varpam]</td>
<td></td>
</tr>
<tr>
<td>q4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[_varpam]</td>
</tr>
</tbody>
</table>

Output 29.5.5 shows the fit summary of the sphericity test. The chi-square is 27.747 ($df = 9$, $p = 0.0011$), which is statistically significant. You reject the sphericity hypothesis for the population covariance matrix.

**Output 29.5.5** Fit Summary of the Sphericity Test on the Sales Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>27.747</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>9</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.0011</td>
</tr>
</tbody>
</table>

Again, the numerical results in Output 29.5.5 are different from those shown in Output 29.4.2 of Example 29.4. This is because with the COVPATTERN=SIGSQI option, the chi-square correction due to Box (1949) has been applied in the current example. To turn off the automatic chi-square correction, you can use the following specification:

```
proc calis data=sales covpattern=sigsqi chicorrect=0;
run;
```

As expected, the numerical results in Output 29.5.6 match exactly to those of in Output 29.4.2 of Example 29.4.

**Output 29.5.6** Fit Summary of the Sphericity Test on the Sales Data: No Chi-Square Correction

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>31.5951</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>9</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.0002</td>
</tr>
</tbody>
</table>

This example shows that for the tests of some standard covariance patterns, you can use the COVPATTERN= option directly. As compared with the use of the explicit MSTRUCT model specifications, which are shown in Example 29.3 and Example 29.4, the use of COVPATTERN= option is more efficient and less error-prone in coding. In addition, it can apply chi-square corrections in appropriate situations.

PROC CALIS also provides the test of some “standard” mean patterns by the MEANPATTERN= option. You can use the COVPATTERN= and MEANPATTERN= options together to define the desired combinations of covariance and mean patterns. See these two options for details. See Example 29.22 for a multiple-group analysis with the simultaneous use of the COVPATTERN= and MEANPATTERN= options. Certainly, the COVPATTERN= and MEANPATTERN= options are limited to the standard covariance and mean patterns.
provided by PROC CALIS. When you need to fit some specific (nonstandard) covariance or mean patterns, the MSTRUCT model specification would be indispensable. See Example 29.19 and Example 29.22 for applications.

---

**Example 29.6: Linear Regression Model**

This example shows how you can use PROC CALIS to fit the basic regression models. Unlike the preceding examples (Example 29.1, Example 29.2, Example 29.3, and Example 29.4) where you specify the covariance structures directly, in this example the covariance structures being analyzed are implied by the functional relationships specified in the model. The PATH modeling language introduced in the current example requires you to specify only the functional or path relationships among variables. PROC CALIS analyzes the implied covariance structures that are derived from the specified functional or path relationships.

Consider the same sales data as in Example 29.1. This example demonstrates a simple linear regression that uses \( q_1 \) (the sales in the first quarter) to predict \( q_4 \) (the sales in the fourth quarter).

In covariance structural analysis, or in general structural equation modeling, relationships among variables are usually represented by the so-called path diagram. For example, you can represent the linear regression of \( q_4 \) on \( q_1 \) by the following simple path diagram:

```
   q1 ----> q4
```

In the path diagram, \( q_1 \) is an exogenous (or independent) variable and \( q_4 \) is an endogenous (or dependent) variable. Formally, a variable in a path diagram is endogenous if there is at least one single-headed arrow pointing to it. Otherwise, the variable is exogenous. In some situations, researchers apply “causal” interpretations among variables in the path diagram, with the single-headed arrows indicating the causal directions. However, causal interpretations are not a requirement for using covariance structure analysis or structural equation modeling.

It is easy to transcribe the preceding path diagram into the PATH model specification in PROC CALIS, as shown in the following statements:

```plaintext
proc calis data=sales;
   path q1 ===> q4;
run;
```

Output 29.6.1 shows the modeling information of the linear regression model. It shows that all 14 observations are used and the model type is PATH. PROC CALIS analyzes the (implied) covariance structure model for the data. In the next table of Output 29.6.1, PROC CALIS shows the nature of the variables in the model: \( q_4 \) is an endogenous manifest variable and \( q_1 \) is an exogenous manifest variable. There is no latent variable in this simple path model.
**Output 29.6.1** Modeling Information of the Linear Regression Model for the Sales Data

**Simple Linear Regression Model by the PATH Modeling Language**

### The CALIS Procedure

**Covariance Structure Analysis: Model and Initial Values**

<table>
<thead>
<tr>
<th>Modeling Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Likelihood</td>
</tr>
<tr>
<td>Estimation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Set</th>
<th>WORK.SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>N Records Read</td>
<td>14</td>
</tr>
<tr>
<td>N Records Used</td>
<td>14</td>
</tr>
<tr>
<td>N Obs</td>
<td>14</td>
</tr>
<tr>
<td>Model Type</td>
<td>PATH</td>
</tr>
<tr>
<td>Analysis</td>
<td>Covariances</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variables in the Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endogenous Manifest</td>
</tr>
<tr>
<td>q4</td>
</tr>
<tr>
<td>Latent</td>
</tr>
<tr>
<td>Exogenous Manifest</td>
</tr>
<tr>
<td>q1</td>
</tr>
<tr>
<td>Latent</td>
</tr>
<tr>
<td>Number of Endogenous Variables = 1</td>
</tr>
<tr>
<td>Number of Exogenous Variables = 1</td>
</tr>
</tbody>
</table>

Output 29.6.2 shows the initial model specification. The path is in the first table. A parameter name is attached to the path. The name _Parm1, which is generated automatically by PROC CALIS, denotes the effect parameter of q1 on q4. In the context of linear regression, _Parm1 also denotes the regression coefficient.

**Output 29.6.2** Initial Specification of the Linear Regression Model for the Sales Data

<table>
<thead>
<tr>
<th>Initial Estimates for PATH List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Parameter Estimate</td>
</tr>
<tr>
<td>q1    q4 _Parm1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial Estimates for Variance Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Exogenous</td>
</tr>
<tr>
<td>Error</td>
</tr>
</tbody>
</table>

NOTE: Parameters with prefix ‘_Add’ are added by PROC CALIS.

Next, Output 29.6.2 shows the variance parameters in the model. You do not need to specify any of these parameters in the preceding PATH model specification—because PROC CALIS adds these parameters by default. _Add1 denotes the variance parameter for the exogenous variable q1. _Add2 denotes the error variance parameter for the endogenous variable q4.

In the PATH model of PROC CALIS, all variances of exogenous variables and all error variances of endogenous variables are free parameters by default. In most practical applications, these parameters are usually free parameters in models and it would be laborious to specify them each time when you fit a covariance structure model. Therefore, to make the PATH model specification more efficient and easier,
PROC CALIS sets these free parameters by default. In fact, with these default parameters in the PATH model, PROC CALIS produces essentially the same regression analysis results as those produced by common linear regression procedures such as PROC REG. This consistency is shown in the subsequent estimation results for the current example.

You can also explicitly specify those otherwise default parameters of the PATH model in PROC CALIS. Depending on the modeling situation, you can set any parameter in the PATH model as a free, fixed, or constrained parameter. You can also provide names for the parameters. Naming parameters is very useful for parameter referencing and for setting up parameter constraints. See Example 29.4. For details, see the PATH statement and the section “The PATH Model” on page 1419.

Output 29.6.3 shows some fit statistics from the linear regression model. The model fit chi-square is 0 with 0 degrees of freedom. This is a perfect model fit. The fit is perfect because the covariance model contains three distinct elements (variance of q1, variance of q4, and covariance between q1 and q4) that are fitted perfectly by three parameters: _Parm1 for the effect of q1 on q4, _Add1 for the variance of variable q1, and _Add2 for the error variance of variable q4. Thus, the unconstrained linear regression model estimates are simply a transformation of the covariance elements. Hence, the model is saturated with a perfect fit and zero degrees of freedom.

Output 29.6.3  Model Fit of the Linear Regression Model for the Sales Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
</tr>
</tbody>
</table>

Output 29.6.4 shows the estimates of the model. The effect of q1 on q4 is 0.6544 (standard error=0.7571). The associated t value is 0.86433, which is not significantly different from zero. The estimated variance of q1 is 0.3383 and the estimated error variance for q4 is 2.5207. Both estimates are significant.

Output 29.6.4  Parameter Estimates of the Linear Regression Model for the Sales Data

<table>
<thead>
<tr>
<th>PATH List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Parameter Estimate</td>
</tr>
<tr>
<td>Standard Error t Value Pr &gt;</td>
</tr>
<tr>
<td>q1  ===&gt;  q4 _Parm1 0.65436 0.75707 0.8643 0.3874</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variance Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Variable Parameter</td>
</tr>
<tr>
<td>Estimate Standard Error t Value Pr &gt;</td>
</tr>
<tr>
<td>Exogenous q1 _Add1 0.33830 0.13269 2.5495 0.0108</td>
</tr>
<tr>
<td>Error q4 _Add2 2.52066 0.98869 2.5495 0.0108</td>
</tr>
</tbody>
</table>

For a simple linear regression such as this one, you could have used PROC REG. You get essentially the same estimates by specifying the following statements:
proc reg data=sales;
   model q4 = q1;
run;

Output 29.6.5 shows the parameter estimates from PROC REG. The intercept estimate is 2.7604 (standard error=1.1643) and the regression coefficient is 0.6544 (standard error=0.7880). The regression coefficient estimate matches PROC CALIS. However, the corresponding standard error estimate in PROC CALIS is 0.7571, which is slightly different from PROC REG. This difference is due to the different variance divisors that are used in calculating the standard error estimates. PROC CALIS uses \((N - 1)\) as the divisor (by default) while PROC REG uses \((N - q - 1)\), where \(N\) is the number of observations and \(q\) is the number of regression coefficients. In the current example, \(q\) is 1 so that the variance divisor in PROC REG is 1 less than the divisor in PROC CALIS. If you have at least a moderate sample size and the number of regression parameters is relatively small compared to the sample size, the discrepancy due to using different variance divisors is of little consequence.

Output 29.6.5 Parameter Estimates from PROC REG for the Sales Data

| Variable | DF | Parameter Estimate | Standard Error | t Value | Pr > |t| |
|----------|----|--------------------|----------------|---------|------|
| Intercept| 1  | 2.76040            | 1.16430        | 2.37    | 0.0353 |
| q1       | 1  | 0.65436            | 0.78798        | 0.83    | 0.4225 |

By default, PROC CALIS analyzes only the covariance structures, which are properties of the second-order moments of the data. PROC CALIS does not automatically produce intercept estimates, which are properties of the first-order moments of the data.

In order to produce the intercept estimate in the linear regression context, you can add the MEANSTR (mean structures) option in the PROC CALIS statement, as shown in the following statements:

```plaintext
proc calis data=sales meanstr;
   path q1 ===> q4;
run;
```

Output 29.6.6 shows the parameter estimates of the model with the MEANSTR option added. Compared with Output 29.6.4, Output 29.6.6 produces one more table: estimates of the mean and intercept. The intercept estimate for q4 is 2.7604, which matches the intercept estimate from PROC REG. The estimated mean of q1 is 1.3671. All other estimates are the same for the analyses with and without the MEANSTR option.

Output 29.6.6 Parameter Estimates of the Linear Regression Model with the MEANSTR option for the Sales Data

| Path | Parameter Estimate | Standard Error | t Value | Pr > |t| |
|------|--------------------|----------------|---------|------|
| q1   | _Parm1             | 0.65436        | 0.75707 | 0.8643 | 0.3874 |

| Variance Parameters | | | |
|---------------------|-----------------|-----------------|---------|------|
| Type                | Variable | Parameter Estimate | Standard Error | t Value | Pr > |t| |
| Exogenous           | q1       | _Add1             | 0.33830        | 0.13269 | 2.5495 | 0.0108 |
| Error               | q4       | _Add2             | 2.52066        | 0.98869 | 2.5495 | 0.0108 |
Linear regression estimates from PROC CALIS are comparable to those obtained from PROC REG, although the two procedures have different default treatments of the variance divisor in calculating the standard error estimates. With the MEANSTR option in the PROC CALIS statement, you can analyze the mean and covariance structures simultaneously. PROC CALIS prints the estimates of the intercepts and means when you model the mean structures.

This example shows how you can fit the linear regression model as a PATH model in PROC CALIS. You need to specify only path relationships among the variables in the PATH statement, because the implied covariance structures are generated and analyzed by PROC CALIS. To make model specification more efficient, PROC CALIS sets default variance parameters for exogenous variables and default error variance parameters for endogenous variables. You can also overwrite these default parameters by explicit specifications. See Example 29.7 for some sophisticated regression models that you can specify with PROC CALIS. See Example 29.17 for a more elaborate path model specification.

**Example 29.7: Multivariate Regression Models**

This example shows how to analyze different types of multivariate regression models with PROC CALIS. **Example 29.6** fits a simple linear regression model to the sales data that are described in **Example 29.1**. The simple linear regression model predicts the fourth quarter sales (q4) from the first quarter sales (q1). There is only one dependent (outcome) variable (q4) and one independent (predictor) variable (q1) in the analysis. Also, there are no constraints on the parameters. This example fits more sophisticated regression models. The models include more than one predictor. Some variables can serve as outcome variables and predictor variables at the same time. This example also illustrates the use of parameter constraints in model specifications and the use of the model fit statistics to search for a “best” model for the sales data.

**Multiple Regression Model for the Sales Data**

Consider a multiple regression model for q4. Instead of using just q1 as the predictor in the model as in **Example 29.6**, use all previous sales q1–q3 to predict the fourth-quarter sale (q4). The path model representation is shown in the following path diagram:
You can transcribe this path diagram into the following PATH model specification:

```
proc calis data=sales;
  path q1 q2 q3 ===> q4;
run;
```

In the path statement, the shorthand path specification

```
path q1 q2 q3 ===> q4;
```

is equivalent to the following specification:

```
path q1 ===> q4, q2 ===> q4, q3 ===> q4;
```

The shorthand notation provides a more convenient way to specify the path model. Some of the model fit statistics are shown in Output 29.7.1. This is a saturated model with perfect fit and zero degrees of freedom. Because the chi-square statistic is always smallest in a saturated model (with a zero chi-square value), it does not make much sense to judge the model quality solely by looking at the chi-square value. However, a saturated model is useful for serving as a baseline model with which other nonsaturated competing models are compared.

**Output 29.7.1** Model Fit of the Multiple Regression Model for the Sales Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
</tr>
</tbody>
</table>

In addition to the model fit chi-square statistic, **Output 29.7.1** also shows Akaike’s information criterion (AIC), Bozdogan’s CAIC, and Schwarz’s Bayesian criterion (SBC) of the saturated model. The AIC, CAIC, and SBC are derived from information theory and henceforth they are referred to as the information-theoretic fit indices. These information-theoretic fit indices measure the model quality by taking the model parsimony into account. The root mean square error of approximation (RMSEA) also takes the model parsimony into account, but it is not an information-theoretic fit index. The values of these information-theoretic fit indices themselves do not indicate the quality of the model. However, when you fit several different models to the same data, you can order the models by these fit indices. The better the model, the smaller the fit index values. Unlike the chi-square statistic, these fit indices do not always favor a saturated model because a saturated model lacks model parsimony (the saturated model uses the most parameters to explain the data). The subsequent discussion uses these fit indices to select the “best” model for the sales data.

**Output 29.7.2** shows the parameter estimates of the multiple regression model. In the first table, all path effect estimates are not statistically significant—that is, all \( t \) values are less than 1.96. The next table in **Output 29.7.2** shows the variance estimates of \( q_1 \)–\( q_3 \) and the error variance estimate for \( q_4 \). All of these estimates are significant. The last table in **Output 29.7.2** shows the covariances among the exogenous variables \( q_1 \)–\( q_3 \). These covariance estimates are small and are not statistically significant.
Example 29.7: Multivariate Regression Models

Output 29.7.2 Parameter Estimates of the Multiple Regression Model for the Sales Data

| Path   | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|--------|-----------|----------|----------------|---------|------|---|
| q1 ==> q4 _Parm1 | 0.55980 | 0.64938 | 0.8621 | 0.3887 |
| q2 ==> q4 _Parm2 | 0.58946 | 0.84558 | 0.6971 | 0.4857 |
| q3 ==> q4 _Parm3 | 0.88290 | 0.51635 | 1.7099 | 0.0873 |

Variance Parameters

| Variance Type | Variable | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|---------------|----------|-----------|----------|----------------|---------|------|---|
| Exogenous     | q1       | _Add1     | 0.33830  | 0.13269        | 2.5495  | 0.0108 |
| q2            | _Add2    | 0.22466   | 0.08812  | 2.5495         | 0.0108  |
| q3            | _Add3    | 0.60633   | 0.23782  | 2.5495         | 0.0108  |
| Error         | q4       | _Add4     | 1.84128  | 0.72221        | 2.5495  | 0.0108 |

Covariances Among Exogenous Variables

| Var1 | Var2 | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|------|------|-----------|----------|----------------|---------|------|---|
| q2   | q1   | _Add5     | 0.0001978| 0.07646        | 0.00259 | 0.9979 |
| q3   | q1   | _Add6     | 0.03610  | 0.12601        | 0.2865  | 0.7745 |
| q3   | q2   | _Add7     | 0.12653  | 0.10821        | 1.1693  | 0.2423 |

In Output 29.7.2, the total number of parameter estimates is 10 (_Parm1–_Parm3 and _Add1–_Add7). Under the covariance structure model, these 10 parameters explain the 10 nonredundant elements in the covariance matrix for the sales data. That is why the model has a perfect fit with zero degrees of freedom.

In Output 29.7.2, notice that some parameters have the prefix ‘_Parm’, while others have the prefix ‘_Add’. Both types of parameter names are generated by PROC CALIS. The parameters named with the ‘_Parm’ prefix are those that were specified in the model, but were not named. In the current example, the parameters specified but not named are the path coefficients (effects) for the three paths in the PATH statement. The parameters named with the ‘_Add’ prefix are default parameters added by PROC CALIS. In the current multiple regression example, the variances and covariances among the predictors (q1–q3) and the error variance for the outcome variable (q4) are default parameters in the model. In general, variances and covariances among exogenous variables and error variances of endogenous variables are default parameters in the PATH model. Avoid using parameter names with the ‘_Parm’ and ‘_Add’ prefixes to avoid confusion with parameters that are generated by PROC CALIS.

Direct and Indirect Effects Model for the Sales Data

In the multiple regression model, q1–q3 are all predictors that have direct effects on q4. This example considers the possibility of adding indirect effects into the multiple regression model. Because of the time ordering, it is reasonable to assume that there is a causal sequence q1 ===> q2 ===> q3. To implement this idea into the model, put two more paths into the preceding path diagram to form the following new path diagram:

...
With the \( q_1 \longrightarrow q_2 \) and \( q_2 \longrightarrow q_3 \) paths, \( q_2 \) and \( q_3 \) are no longer exogenous in the model. They become endogenous. The only exogenous variable in the model is \( q_1 \), which has a direct effect in addition to indirect effects on \( q_4 \). The direct effect is indicated by the \( q_1 \longrightarrow q_4 \) path. The indirect effects are indicated by the following two causal chains: \( q_1 \longrightarrow q_2 \longrightarrow q_4 \) and \( q_1 \longrightarrow q_2 \longrightarrow q_3 \longrightarrow q_4 \). Similarly, \( q_2 \) has a direct and an indirect effect on \( q_4 \). However, \( q_3 \) has only a direct effect on \( q_4 \). You can use the following statements to specify this direct and indirect effects model:

```plaintext
proc calis data=sales;
  path   q1 ===> q2,
         q2 ===> q3,
         q1 q2 q3 ===> q4;
run;
```

Although the direct and indirect effects model has two more paths in the PATH statement than does the preceding multiple regression model, the current model is more precise because it has one fewer parameter. By introducing the causal paths \( q_1 \longrightarrow q_2 \) and \( q_2 \longrightarrow q_3 \), the six variances and covariances among \( q_1 \)–\( q_3 \) are explained by: the two causal effects, the exogenous variance of \( q_1 \), and the error variances for \( q_2 \) and \( q_3 \) (that is, five parameters in the model). Hence, the current direct and indirect effects model has one fewer parameter than the preceding multiple regression model.

Output 29.7.3 shows some model fit indices of the direct and indirect effects model. The model fit chi-square is 0.0934 with one degree of freedom. It is not significant. Therefore, you cannot reject the model on statistical grounds. The standardized root mean squares of residuals (SRMR) is 0.028 and the root mean square error of approximation (RMSEA) is close to zero. Both indices point to a very good model fit. The AIC, CAIC, and SBC are all smaller than those of the saturated model, as shown in Output 29.7.1. This suggests that the direct and indirect effects model is better than the saturated model.

**Output 29.7.3** Model Fit of the Direct and Indirect Effects Model for the Sales Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>0.0934</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>1</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.7600</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0280</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0000</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>18.0934</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>32.8449</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>23.8449</td>
</tr>
</tbody>
</table>
Output 29.7.4 shows the parameter estimates of the *direct and indirect effects* model. All the path effects are not significant, while all the variance or error variance estimates are significant. Unlike the saturated model where you have covariance estimates among several exogenous variables (as shown in Output 29.7.2), in the *direct and indirect effects* model there is only one exogenous variable (q1) and hence there is no covariance estimate in the results.

**Output 29.7.4** Parameter Estimates of the Direct and Indirect Effects Model for the Sales Data

<table>
<thead>
<tr>
<th>PATH List</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>Parameter</td>
</tr>
<tr>
<td>q1 ===&gt; q2</td>
<td>_Parm1</td>
</tr>
<tr>
<td>q2 ===&gt; q3</td>
<td>_Parm2</td>
</tr>
<tr>
<td>q1 ===&gt; q4</td>
<td>_Parm3</td>
</tr>
<tr>
<td>q2 ===&gt; q4</td>
<td>_Parm4</td>
</tr>
<tr>
<td>q3 ===&gt; q4</td>
<td>_Parm5</td>
</tr>
</tbody>
</table>

**Variance Parameters**

| Variance Type | Variable | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|---------------|----------|-----------|----------|----------------|---------|-------|
| Exogenous     | q1       | _Add1     | 0.33830  | 0.13269        | 2.5495  | 0.0108 |
| Error         | q2       | _Add2     | 0.22466  | 0.08812        | 2.5495  | 0.0108 |
|               | q3       | _Add3     | 0.53506  | 0.20987        | 2.5495  | 0.0108 |
|               | q4       | _Add4     | 1.84128  | 0.72221        | 2.5495  | 0.0108 |

Although the current *direct and indirect effects* model is better than the saturated model and both the SRMR and RMSEA indicate a good model fit, the nonsignificant path effect estimates are unsettling. You continue to explore alternative models for the data.

**Indirect Effects Model for the Sales Data**

The saturated model includes only the direct effects of q1–q3 on q4, while the *direct and indirect effects* model includes both the direct and indirect effects of q1 and q2 on q4. An alternative model with only the indirect effects of q1 and q2 on q4, but without their direct effects, is possible. Such an *indirect effects* model is represented by the following path diagram:
You can easily transcribe this path diagram into the following PATH model specification:

```plaintext
calis data=sales;
path q1 ===> q2,
     q2 ===> q3,
     q3 ===> q4;
run;
```

Output 29.7.5 shows some model fit indices for the indirect effects model. The chi-square model fit statistic is not statistically significant, so the model is not rejected. The standardized RMR is 0.0905, which is a bit higher than the conventional value of 0.05 for an acceptable good model fit. However, the RMSEA is close to zero, which shows a very good model fit. The AIC, CAIC and SBC are all smaller than the direct and indirect effects model. These information-theoretic fit indices suggest that the indirect effects model is better.

**Output 29.7.5** Model Fit of the Indirect Effects Model for the Sales Data

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fit Summary</td>
<td></td>
</tr>
<tr>
<td>Chi-Square</td>
<td>1.2374</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>3</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.7440</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0905</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0000</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>15.2374</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>26.7108</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>19.7108</td>
</tr>
</tbody>
</table>

Output 29.7.6 shows the parameter estimates of the indirect effects model. All the variance and error variance estimates are statistically significant. However, only the path effect of q3 on q4 is statistically significant, and all other path effects are not. Having significant variances with nonsignificant paths raises some concerns about accepting the current model even though the AIC, CAIC, and SBC values suggest that it is the best model so far.

**Output 29.7.6** Parameter Estimates of the Indirect Effects Model for the Sales Data

<table>
<thead>
<tr>
<th>Path</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1 ===&gt; q2</td>
<td>_Parm1</td>
<td>0.0005847</td>
<td>0.22602</td>
<td>0.00259</td>
<td>0.9979</td>
<td></td>
</tr>
<tr>
<td>q2 ===&gt; q3</td>
<td>_Parm2</td>
<td>0.56323</td>
<td>0.42803</td>
<td>1.3159</td>
<td>0.1882</td>
<td></td>
</tr>
<tr>
<td>q3 ===&gt; q4</td>
<td>_Parm3</td>
<td>1.03924</td>
<td>0.50506</td>
<td>2.0577</td>
<td>0.0396</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variance Parameters</th>
<th>Variance Type</th>
<th>Variable</th>
<th>Parameter</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exogenous</td>
<td>q1</td>
<td>_Add1</td>
<td>0.33830</td>
<td>0.13269</td>
<td>2.5495</td>
<td>0.0108</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>q2</td>
<td>_Add2</td>
<td>0.22466</td>
<td>0.08812</td>
<td>2.5495</td>
<td>0.0108</td>
<td></td>
</tr>
<tr>
<td></td>
<td>q3</td>
<td>_Add3</td>
<td>0.53506</td>
<td>0.20987</td>
<td>2.5495</td>
<td>0.0108</td>
<td></td>
</tr>
<tr>
<td></td>
<td>q4</td>
<td>_Add4</td>
<td>2.01067</td>
<td>0.78865</td>
<td>2.5495</td>
<td>0.0108</td>
<td></td>
</tr>
</tbody>
</table>
Constrained Indirect Effects Model for the Sales Data

In the preceding indirect effects model, some path effects are not significant. In the current model, all the path effects are constrained to be equal. The following path diagram represents the constrained indirect effects model:

```
  q1  
  |  gamma
  v
q2  
  |  gamma
  v
q3  
  |  gamma
  v
  q4
```

Except for one notable difference, this path diagram is the same as the path diagram for the preceding indirect effects model. The current path diagram labels all the paths with the same name (gamma) to signify that they are the same parameter. You can specify this constrained indirect effects model with this chosen constraint on the path effects by the using following statements:

```
proc calis data=sales;
  path q1 ===> q2 = gamma,
       q2 ===> q3 = gamma,
       q3 ===> q4 = gamma;
run;
```

In the PATH statement, append an equal sign and a parameter name gamma in each of the path entries. This specification means that all the associated path effects are the same parameter named gamma.

Output 29.7.7 shows some fit indices for the constrained indirect effects model. Again, the model fit chi-square statistic is not significant. However, the SRMR is 0.2115, which is too large to accept as a good model. The RMSEA is 0.0499, which still indicates a good model fit. The AIC, CAIC, and SBC values are a bit smaller than those of the preceding unconstrained indirect effects model. Therefore, it seems that constraining the path effects leads to a slightly better model.

**Output 29.7.7** Model Fit of the Constrained Indirect Effects Model for the Sales Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>5.1619</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>5</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.3964</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.2115</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0499</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>15.1619</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>23.3572</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>18.3572</td>
</tr>
</tbody>
</table>

Output 29.7.8 shows the parameter estimates of the constrained indirect effects model. Again, all variance and error variance estimates are significant, and all path effects are not significant. The effect estimate is 0.24 (standard error=0.19, $t = 1.25$).
Constrained Indirect Effects and Error Variances Model for the Sales Data

In addition to constraining all the path effects in the preceding model, the current model constrains all the error variances. Before using a path diagram to represent the current constrained indirect effects and constrained error variances, it is important to realize that you have not manually defined variances and covariances in the path diagrams for all of the preceding models. The default parameterization in PROC CALIS defined those parameters.

Represent the variances and covariances in a path diagram with double-headed arrows. When a double-headed arrow points to a single variable, it represents the variance parameter. When a double-headed arrow points to two distinct variables, it represents the covariance between the two variables. Consider the unconstrained indirect effects model for the sales data as an example. A more complete path diagram representation is as follows:

In this path diagram, a double-headed arrow on each variable represents variance or error variance. For q1, the double-headed arrow represents the variance parameter of q1. For other variables, the double-headed arrows represent error variances because those variables are endogenous (that is, they are predicted from other variables) in the model.

In order to represent the equality-constrained parameters in the model, you can put parameter names in the respective parameter locations in the path diagram. For the current constrained indirect effects and error variances model, you can represent the model by the following path diagram:
In the path diagram, label all the path effects by the parameter $\gamma$ and all error variances by the parameter $\text{evar}$. The double-headed arrow attached to $q_1$ is not labeled by any name. This means that it is an unnamed free parameter in the model.

You can transcribe the path diagram into the following statements:

```plaintext
proc calis data=sales;
   path q1 ===> q2 = $\gamma$,
        q2 ===> q3 = $\gamma$,
        q3 ===> q4 = $\gamma$;
   pvar q2 q3 q4 = 3 * $\text{evar}$;
run;
```

The specification in the PATH statement is the same as the preceding PATH model specification for the constrained indirect effects model. The new specification here is the PVAR statement. You use the PVAR statement to specify partial variances, which include the (total) variances of exogenous variables and the error variances of the endogenous variables. In the PVAR statement, you specify the variables for which you intend to define variances. If you do not specify anything after the list of variables, the variances of these variables are unnamed free parameters. If you put an equal sign after the variable lists, you can specify parameter names, initial values, or fixed parameters for the variances of the variables. See the PVAR statement for details. In the current model, $3 \times \text{evar}$ means that you want to specify $\text{evar}$ three times (for the error variance parameters of $q_2$, $q_3$, and $q_4$).

Note that you did not specify the variance of $q_1$ in the PVAR statement. This variance is a default parameter in the model, and therefore you do not need to specify it in the PVAR statement. Alternatively, you can specify it explicitly in the PVAR statement by giving it a parameter name. For example, you can specify the following:

```plaintext
pvar q2 q3 q4 = 3 * $\text{evar}$,
     q1 = MyOwnName;
```

Or, you can specify it explicitly without giving it a parameter name, as shown in following statement:

```plaintext
pvar q2 q3 q4 = 3 * $\text{evar}$,
       q1 ;
```

All these specifications lead to the same estimation results. The difference between the two specifications is the explicit parameter name for the variance of $q_1$. Without putting $q_1$ in the PVAR statement, the variance parameter is named with the prefix _Add, which is generated as a default parameter by PROC CALIS. With the explicit specification of $q_1$, the variance parameter is named MyOwnName. With the explicit specification
of q1, but without giving it a parameter name in the PVAR statement, the variance parameter is named with the prefix _Parm, which PROC CALIS generates for unnamed free parameters.

Output 29.7.9 shows some fit indices for the constrained indirect effects and error variances model. The model fit chi-square is 19.7843, which is significant at the 0.05 \( \alpha \)-level. In practice, the model fit chi-square statistic is not the only criterion for judging model fit. In fact, it might not even be the most commonly used criterion for measuring model fit. Other criteria such as the SRMR and RMSEA are more popular or important. Unfortunately, the values of these two fit indices do not support the current constrained model either. The SRMR is 1.5037 and the RMSEA is 0.3748. Both are much greater than the commonly accepted 0.05 criterion.

**Output 29.7.9** Model Fit of the Constrained Indirect Effects and Error Variances Model for the Sales Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
</tr>
</tbody>
</table>

The AIC, CAIC, and SBC values are all much greater than those of the preceding constrained indirect effects model. Therefore, constraining the error variances in addition to the constrained indirect effects does not lead to a better model.

Output 29.7.10 shows the parameter estimates of the constrained indirect effects and error variances model. All estimates are significant in the model, which is often desirable. However, because of the bad model fit, this model is not acceptable.

**Output 29.7.10** Parameter Estimates of the Constrained Indirect Effects and Error Variances Model for the Sales Data

<table>
<thead>
<tr>
<th>PATH List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
</tr>
<tr>
<td>q1 ===&gt; q2</td>
</tr>
<tr>
<td>q2 ===&gt; q3</td>
</tr>
<tr>
<td>q3 ===&gt; q4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variance Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance Type</td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Exogenous q1</td>
</tr>
</tbody>
</table>
Partially Constrained Model for the Sales Data

In the preceding model, constraining all error variances to be same shows that the model fit is unacceptable, even though all parameter estimates are significant. Relaxing those constraints a little might improve the model. The following path diagram represents such a partially constrained model:

![Path Diagram](image)

The only difference between the current partially constrained model and the preceding constrained indirect effects and error variances model is that the error variance for q4 is no longer constrained to be equal to the error variances of q2 and q3. In the path diagram, evar is no longer attached to the double-headed arrow that is associated with the error variance of q4. You can transcribe this path diagram representation into the following PATH model specification:

```plaintext
proc calis data=sales;
    path q1 ===> q2 = gamma,
         q2 ===> q3 = gamma,
         q3 ===> q4 = gamma;
    pvar q2 q3 = 2 * evar,
         q4 q1;
run;
```

Now, the PVAR statement has only the error variances of q2 and q3 constrained to be equal. The error variance of q4 and the variance of q1 are free parameters without constraints.

Output 29.7.11 shows some fit indices for the partially constrained model. The chi-square model fit test statistic is not significant. The SRMR is 0.3877 and the RMSEA is 0.1164. These are far from the conventional acceptance level of 0.05. However, the AIC, CAIC, and SBC values are all slightly smaller than the constrained indirect effects model, as shown in Output 29.7.7. In fact, these information-theoretic fit indices suggest that the partially constrained model is the best model among all models that have been considered.
Output 29.7.11  Model Fit of the Partially Constrained Model for the Sales Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
</tr>
</tbody>
</table>

Output 29.7.12 shows the parameter estimates of the partially constrained model. Again, all variance and error variance parameters are statistically significant. However, the path effects are only marginally significant.

Output 29.7.12  Parameter Estimates of the Partially Constrained Model for the Sales Data

<table>
<thead>
<tr>
<th>PATH List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
</tr>
<tr>
<td>q1 === q2</td>
</tr>
<tr>
<td>q2 === q3</td>
</tr>
<tr>
<td>q3 === q4</td>
</tr>
</tbody>
</table>

Variance Parameters

<table>
<thead>
<tr>
<th>Variance Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance Type</td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td>q3</td>
</tr>
<tr>
<td>q4</td>
</tr>
<tr>
<td>Exogenous q1</td>
</tr>
</tbody>
</table>

Which Model Should You Choose?

You fit various models in this example for the sales data. The fit summary of the models is shown in the following table:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>df</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>p-value</td>
<td>.</td>
<td>0.76</td>
<td>0.74</td>
<td>0.40</td>
<td>0.01</td>
</tr>
<tr>
<td>SRMSE</td>
<td>0</td>
<td>0.03</td>
<td>0.09</td>
<td>0.21</td>
<td>1.50</td>
</tr>
<tr>
<td>RMSEA</td>
<td>.</td>
<td>0.00</td>
<td>0.00</td>
<td>0.05</td>
<td>0.37</td>
</tr>
<tr>
<td>AIC</td>
<td>20.00</td>
<td>18.09</td>
<td>15.24</td>
<td>15.16</td>
<td>25.78</td>
</tr>
<tr>
<td>CAIC</td>
<td>36.39</td>
<td>32.84</td>
<td>26.71</td>
<td>23.36</td>
<td>30.70</td>
</tr>
<tr>
<td>SBC</td>
<td>26.39</td>
<td>23.84</td>
<td>19.71</td>
<td>18.36</td>
<td>27.70</td>
</tr>
</tbody>
</table>
As discussed previously, the model fit chi-square test statistic always favors models with a lot of parameters. It does not take model parsimony into account. In particular, a saturated model (Model 1) always has a perfect fit. However, it does not explain the data in a concise way. Therefore, the model fit chi-square statistic is not used here for comparing the competing models.

The standardized root mean square residual (SRMR) also does not take the model parsimony into account. It tells you how the fitted covariance matrix is different from the observed covariance matrix in a certain standardized way. Again, it always favors models with a lot of parameters. As shown in the preceding table, the more parameters (the fewer degrees of freedom) the model has, the smaller the SRMR is. A conventional criterion is to accept a model with SRMR less than 0.05. Applying this criterion, only the saturated model (Model 1) and the direct and indirect effects (Model 2) models are acceptable. The indirect effects model (Model 3) is marginally acceptable.

The root mean square error of approximation (RMSEA) fit index does take model parsimony into account. With the ‘RMSEA less than 0.05 criterion’, the constrained indirect effects and error variances model (Model 5) and the partially constrained model (Model 6) are not acceptable.

The information-theoretic fit indices such as the AIC, CAIC, and SBC also take model parsimony into account. All of these indices point to the partially constrained model (Model 6) as the best model among the competing models. However, because this model has a relatively bad absolute fit, as indicated by the large SRMR value (0.39), accepting this model is questionable. In addition, the information-theoretic fit indices of the indirect effects model (Model 3) and of the constrained indirect effects model (Model 4) are not too different from those of the partially constrained model (Model 6). The indirect effects model is especially promising because it has relatively small SRMR and RMSEA values. The drawback is that some path effect estimates in the indirect effects model are not significant. Perhaps collecting and analyzing more data might confirm these promising models with significant path effects.

You might not be able to draw a unanimous conclusion about the best model for the sales data of this example. Different fit indices in structural equation modeling do not always point to the same conclusions. The analyses in the current example show some of the complexity of structural equation modeling. Some interesting questions about model selections are:

- Do you choose a model based on a single fit criterion? Or, do you consider a set of model fit criteria to weigh competing models?
- Which fit index criterion is the most important for judging model fit?
- In selecting your “best” model, how do you take “chance” into account?
- How would you use your substantive theory to guide your model search?

The answers to these interesting research questions might depend on the context. Nonetheless, PROC CALIS can help you in the model selection process by computing various kinds of fit indices. (Only a few of these fit indices are shown in the output of this example. See the FITINDEX statement for a wide variety of fit indices that you can obtain from PROC CALIS.)

**Alternative PATH Model Specifications for Variances and Covariances**

The PATH modeling language of PROC CALIS is designed to map the path diagram representation into the PATH statement syntax efficiently. For any path that is denoted by a single-headed arrow in the path diagram, you can specify a path entry in the PATH statement. You can also specify double-headed arrows in the PATH statement.
Consider the preceding path diagram for the *partially constrained* model for the sales data. You use double-headed arrows to denote variances or error variances of the variables. The path diagram is shown in the following:

![Path Diagram](image)

As discussed previously, you can use the PVAR statement to specify these variances or error variances as in the following syntax:

```plaintext
pvar q2 q3 = 2 * evar,
     q4 q1;
```

Alternatively, you can specify these double-headed arrows directly as paths in the PATH statement, as shown in the following statements:

```plaintext
proc calis data=sales;
    path q1 ===> q2 = gamma,
         q2 ===> q3 = gamma,
         q3 ===> q4 = gamma,
         q2 <==> q2 = evar,
         q3 <==> q3 = evar,
         q4 <==> q4,
         q1 <==> q1;
run;
```

To specify the double-headed paths pointing to individual variables, you begin with the double-headed arrow notation `<===>`, followed by the list of variables. For example, in the preceding specification, the error variance of q4 and the variance of q1 are specified in the last path entry of the PATH statement. If you want to define the parameter names for the variances, you can add a parameter list after an equal sign in the path entries. For example, the error variances of q2 and q3 are denoted by the free parameter `evar` in a path entry in the PATH statement.

Alternatively, you can specify the double-headed arrow paths literally in a PATH statement, as shown in the following equivalent specification:

```plaintext
proc calis data=sales;
    path q1 ===> q2 = gamma,
         q2 ===> q3 = gamma,
         q3 ===> q4 = gamma,
         q2 <==> q2 = evar,
         q3 <==> q3 = evar,
         q4 <==> q4,
         q1 <==> q1;
run;
```
For example, the path entry \( q_1 \leftrightarrow q_1 \) specifies the variance of \( q_1 \). It is an unnamed free parameter in the model.

Output 29.7.13 show the parameter estimates for this alternative specification method. All these estimates match exactly those with the PVAR statement specification, as shown in Output 29.7.12. The only difference is that all estimation results are now presented under one PATH List, as shown in Output 29.7.13, instead of as two tables as shown in Output 29.7.12.

**Output 29.7.13** Path Estimates of the Partially Constrained Model for the Sales Data

<table>
<thead>
<tr>
<th>Path</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_1 \leftrightarrow q_2 )</td>
<td>gamma</td>
<td>0.35546</td>
<td>0.18958</td>
<td>1.8750</td>
<td>0.0608</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_2 \leftrightarrow q_3 )</td>
<td>gamma</td>
<td>0.35546</td>
<td>0.18958</td>
<td>1.8750</td>
<td>0.0608</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_3 \leftrightarrow q_4 )</td>
<td>gamma</td>
<td>0.35546</td>
<td>0.18958</td>
<td>1.8750</td>
<td>0.0608</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_2 \leftrightarrow q_2 )</td>
<td>evar</td>
<td>0.40601</td>
<td>0.11261</td>
<td>3.6056</td>
<td>0.0003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_3 \leftrightarrow q_3 )</td>
<td>evar</td>
<td>0.40601</td>
<td>0.11261</td>
<td>3.6056</td>
<td>0.0003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 \leftrightarrow q_4 )</td>
<td>_Parm1</td>
<td>2.29415</td>
<td>0.89984</td>
<td>2.5495</td>
<td>0.0108</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_1 \leftrightarrow q_1 )</td>
<td>_Parm2</td>
<td>0.33830</td>
<td>0.13269</td>
<td>2.5495</td>
<td>0.0108</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The double-headed arrow path syntax applies to covariance specification as well. For example, the following PATH statement specifies the covariances among variables x1–x3:

```
path x2 <==> x1,
    x3 <==> x1,  x3 <==> x2;
```

In the beginning of the current example, you use the following path diagram to represent the multiple regression model for the sales data:

```
\[ q_1 \]
  \[ ------- \]
  \[ q_2 \] \[ q_4 \]
  \[ ------- \]
  \[ q_3 \]
```

The following statements specify the multiple regression model:

```
proc calis data=sales;
    path   q1 q2 q3 ===> q4;
run;
```
You do not represent the covariances and variances among the exogenous variables explicitly in the path diagram, nor in the PATH statement specification. However, PROC CALIS generates them as free parameters by default. Some researchers might prefer to represent the exogenous variances and covariances explicitly in the path diagram, as shown in the following path diagram:

In the path diagram, there are three single-head arrows and seven double-headed arrows. These 10 paths represent the 10 parameters in the covariance structure model. To represent all these parameters in the PATH model specification, you can use the following statements:

```sas
proc calis data=sales;
  path  q1 ===> q4 ,
       q2 ===> q4 ,
       q3 ===> q4 ,
       q1  <==> q1 ,
       q2  <==> q2 ,
       q3  <==> q3 ,
       q1  <==> q2 ,
       q2  <==> q3 ,
       q1  <==> q3 ,
       q4  <==> q4 ;
run;
```

The first three path entries in the PATH statement reflect the single-headed paths in the path diagram. The next six path entries in the PATH statement reflect the double-headed paths among the exogenous variables q1–q3 in the path diagram. The last path entry in the PATH statement reflects the double-headed path attached to the endogenous variable q4 in the path diagram. With this specification, the parameter estimates for the multiple regression model are all shown in Output 29.7.14.
Output 29.7.14 Path Estimates of the Multiple Regression Model for the Sales Data

| Path | Parameter | Estimate | Standard Error | t Value | Pr > |t|
|------|-----------|----------|---------------|---------|------|
| q1  ===> q4 _Parm01 | 0.55980 | 0.64938 | 0.8621 | 0.3887 |
| q2  ===> q4 _Parm02 | 0.58946 | 0.84558 | 0.6971 | 0.4857 |
| q3  ===> q4 _Parm03 | 0.88290 | 0.51635 | 1.7099 | 0.0873 |
| q1  <==> q1 _Parm04 | 0.33830 | 0.13269 | 2.5495 | 0.0108 |
| q2  <==> q2 _Parm05 | 0.22466 | 0.08812 | 2.5495 | 0.0108 |
| q3  <==> q3 _Parm06 | 0.60633 | 0.23782 | 2.5495 | 0.0108 |
| q1  <==> q2 _Parm07 | 0.0001978 | 0.07646 | 0.00259 | 0.9979 |
| q2  <==> q3 _Parm08 | 0.12653 | 0.10821 | 1.1693 | 0.2423 |
| q1  <==> q3 _Parm09 | 0.03610 | 0.12601 | 0.2865 | 0.7745 |
| q4  <==> q4 _Parm10 | 1.84128 | 0.72221 | 2.5495 | 0.0108 |

These estimates are the same as those in Output 29.7.2, where the estimates are shown in three different tables, instead of in one table for all paths as in Output 29.7.14.

Sometimes, specification of some single-headed and double-headed paths can become very laborious. Fortunately, PROC CALIS provides shorthand notation for the PATH statement to make the specification more efficient. For example, a more concise way to specify the preceding multiple regression model is shown in the following statements:

```sas
proc calis data=sales;
path q1 q2 q3 ===> q4 ,
    <==> [q1-q3] ,
    <==> q4 ;
run;
```

The first path entry `q1 q2 q3 ===> q4` in the PATH statement represents the three single-headed arrows in the path diagram. The second path entry `==> [q1-q3]` generates the variances and covariances for the set of variables specified in the rectangular brackets. The last path entry represents the error variance of q4. Consequently, expanding the preceding shorthand specification generates the following specification:

```sas
proc calis data=sales;
path q1 ===> q4 ,
    q2 ===> q4 ,
    q3 ===> q4 ,
    q1 <==> q1 ,
    q2 <==> q1 ,
    q2 <==> q2 ,
    q3 <==> q1 ,
    q3 <==> q2 ,
    q3 <==> q3 ,
    q4 <==> q4 ;
run;
```

Notice that the third through ninth path entries correspond to the lower triangular elements of the covariance matrix for q1–q3.

**CAUTION:** The double-headed path specification does not represent a reciprocal relationship. That is, the following statement specifies the covariance between x2 and x1:
Chapter 29: The CALIS Procedure

Example 29.8: Measurement Error Models

In this example, you use PROC CALIS to fit some measurement error models. You use latent variables to define “true” scores variables that are measured without errors. You constrain parameters by using parameter names or fixed values in the PATH model specification.

Consider a simple linear regression model with dependent variable $y$ and predictor variable $x$. The path diagram for this simple linear regression model is depicted as follows:

Suppose you have the following SAS data set for the regression analysis of $y$ on $x$:

```
data measures;
  input x y @@;
datalines;
7.32266   13.2590  5.76977  10.7654  5.62881  11.5041
7.51832   12.3588  5.48877  11.2211  7.50323  13.3735
5.37445   9.6366   6.00419  11.7654  6.89546  13.1493
;```

This data set contains 30 observations for the $x$ and $y$ variables. You can fit the simple linear regression model to the measures data by the PATH model specification of PROC CALIS, as shown in the following statements:
Example 29.8: Measurement Error Models

```
proc calis data=measures;
  path
    x ===> y;
run;
```

Output 29.8.1 shows that the regression coefficient estimate (denoted as _Parm1 in the PATH List) is 1.1511 (standard error = 0.1002).

Output 29.8.1 Estimates of the Linear Regression Model for the Measures Data

<table>
<thead>
<tr>
<th>Path</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x ===&gt; y</td>
<td>_Parm1</td>
<td>1.15112</td>
<td>0.10016</td>
<td>11.4924</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can also do the simple linear regression by PROC REG by the following statement:

```
proc reg data=measures;
  model y = x;
run;
```

Output 29.8.2 shows that PROC REG essentially gives the same regression coefficient estimate with a similar standard error estimate. The discrepancy in the standard error estimates produced by the two procedures is due to the different variance divisors in computing standard errors in the two procedures. But the discrepancy is negligible when the sample size becomes large.

Output 29.8.2 PROC REG Estimates of the Linear Regression Model for the Measures Data

<table>
<thead>
<tr>
<th>Parameter Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>Intercept</td>
</tr>
<tr>
<td>x</td>
</tr>
</tbody>
</table>

There are two main differences between PROC CALIS and PROC REG regarding the parameter estimation results. First, PROC CALIS does not give the estimate of the intercept because by default PROC CALIS analyzes only the covariance structures. Therefore, it does not estimate the intercept. To obtain the intercept estimate, you can add the MEANSTR option in the PROC CALIS statement, as is shown in Example 29.9. Second, in Output 29.8.1 of PROC CALIS, the variance estimate of x and the error variance estimate of y are shown. The corresponding results are not shown as parameter estimates in the PROC REG results. In PROC CALIS, these two variances are model parameters in covariance structure analysis. PROC CALIS adds these variances as default parameters. You can also represent these two variance parameters by double-headed arrows in the path diagram, as shown in the following:
The two double headed-arrows attached to \( x \) and \( y \) represent the variances. Although it is not necessary to specify these default parameters, you can use the PVAR statement to specify them explicitly, as shown in the following statements:

```plaintext
proc calis data=measures meanstr;
  path
    x ===> y;
  pvar
    x y;
run;
```

In the PROC CALIS statement, you specify the MEANSTR option to request the analysis of mean structures together with covariance structures. Output 29.8.3 shows the estimation results.

**Output 29.8.3** Estimates of the Measurement Error Model with Error in \( x \)

| Path         | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|--------------|-----------|----------|----------------|---------|------|--|
| \( x \) ===> \( y \)  | _Parm1    | 1.15112  | 0.10016        | 11.4924 | <.0001 |

| Variety Type | Variable | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|--------------|----------|-----------|----------|----------------|---------|------|--|
| Exogenous    | \( x \)  | _Parm2    | 0.79962  | 0.20999       | 3.8079  | 0.0001 |
| Error        | \( y \)  | _Parm3    | 0.23265  | 0.06110       | 3.8079  | 0.0001 |

| Type | Variable | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|------|----------|-----------|----------|----------------|---------|------|--|
| Intercept | \( y \)  | _Add1     | 4.62455  | 0.69578       | 6.6466  | <.0001 |
| Mean    | \( x \)  | _Add2     | 6.88865  | 0.16605       | 41.4850 | <.0001 |

The regression coefficient estimate and the variance estimates are the same as those in Output 29.8.1. However, in Output 29.8.3, there is an additional table for the mean and intercept estimates. The intercept estimate for \( y \) is 4.6246 (standard error=0.6958), which match closely to the results obtained from PROC REG, as shown in Output 29.8.2.

**Measurement Error in \( x \)**

PROC CALIS can also handle more complicated regression situations where the variables are measured with errors. This is beyond the application of PROC REG.
Suppose that the predictor variable $x$ is measured with error and from prior studies you know that the size of the measurement error variance is about 0.019. You can use PROC CALIS to incorporate this information into the model. First, think of the measured variable $x$ as composed of two components: one component is the “true” score measure $F_x$ and the other is the measurement error $e_1$. Both of these components are not observed (that is, latent) but they sum up to yield $x$. That is,

$$x = F_x + e_1$$

Because $x$ is contaminated with measurement error, what you are interested in knowing is the regression effect of the true score $F_x$ on $x$. The following path diagram represents this regression scenario:

![Path Diagram]

In path diagrams, latent variables are usually represented by circles or ovals, while observed variables are represented by rectangles. In the current path diagram, $F_x$ is a latent variable and is represented by a circle. The other two variables are observed variables and are represented by rectangles. There are five arrows in the path diagram. Two of them are single-headed arrows that represent functional relationships, while the other three are double-headed arrows that represent variances or error variances. Two paths are labeled with fixed values. The path effect from $F_x$ to $x$ is fixed at 1, as assumed in the measurement error model. The error variance for measuring $x$ is fixed at 0.019 due to the prior knowledge about the measurement error variance. The remaining three arrows represent free parameters in the model: the regression coefficient of $y$ on $F_x$, the variance of $F_x$, and the error variance of $y$. The following statements specify the model for this path diagram:

```plaintext
proc calis data=measures;
  path
    x <= Fx = 1.,
    Fx ===> y;
  pvar
    x = 0.019,
    Fx, y;
run;
```

You specify all the single-headed paths in the PATH statement and all the double-headed arrows in the PVAR statement. For paths with fixed values, you put the equality at the back of the specifications to tell PROC CALIS about the fixed values. For example, the path coefficient in the path $x \leftarrow F_x$ is fixed at 1 and the error variance for $x$ is fixed at 0.019. All other specifications represent unnamed free parameters in the model.

Output 29.8.4 shows the estimation results. The effect of $F_x$ on $y$ is 1.1791 (standard error=0.1029). This effect is slightly greater than the corresponding effect (1.1511) of $x$ on $y$ in the preceding model where the measurement error of $x$ has not been taken into account, as shown in Output 29.8.3.

### Output 29.8.4: Estimates of the Measurement Error Model with Error in x

| Path    | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|---------|-----------|----------|----------------|---------|------|---|
| $x \leftarrow F_x$ |           | 1.00000  |               |         |      |   |
| $F_x \rightarrow y$ | _Parm1    | 1.17914  | 0.10288        | 11.4615 | <.0001 |
Measurement Errors in $x$ and $y$

Measurement error can occur in the $y$ variable too. Suppose that both $x$ and $y$ are measured with errors. From prior studies, the measurement error variance of $x$ is known to be 0.019 (as in the preceding modeling scenario) and the measurement error variance of $y$ is known to be 0.022. The following path diagram represents the current modeling scenario:

![Path Diagram](image)

In the current path diagram the true score variable $F_y$ and its measurement indicator $y$ have the same kind of relationship as the relationship between the true score variable $F_x$ and its measurement indicator $x$ in the previous description. The error variance for measuring $y$ is treated as a known constant 0.022. You can transcribe this path diagram easily to the following PROC CALIS specification:

```plaintext
proc calis data=measures;
    path
        x <= Fx = 1.,
        Fx ===> Fy ,
        Fy ===> y = 1.;
    pvar
        x = 0.019,
        y = 0.022,
        Fx Fy;
run;
```

Again, you specify all the single-headed paths in the PATH statement and the double-headed paths in the PVAR statement. You provide the fixed parameter values by appending the required equalities after the individual specifications.

Output 29.8.5 shows the estimation results of the model with measurement errors in both $x$ and $y$. The effect of $F_x$ on $F_y$ is 1.1791 (standard error=0.1029). This is essentially the same effect of $F_x$ on $y$ as in the preceding measurement model in which no measurement error in $y$ is assumed.
Output 29.8.5 Estimates of the Measurement Error Model with Errors in x and y

<table>
<thead>
<tr>
<th>Path</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x &lt;=Fx</td>
<td></td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fx ==&gt; Fy _Parm1</td>
<td></td>
<td>1.17914</td>
<td>0.10288</td>
<td>11.4615</td>
<td>&lt;0.001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fy ==&gt; y</td>
<td></td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variance Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td>y</td>
</tr>
<tr>
<td>Exogenous</td>
</tr>
<tr>
<td>Error</td>
</tr>
</tbody>
</table>

The estimated error variance for Fy in the current model is 0.1849 and the measurement error variance of y is fixed at 0.022, as shown in the last table of Output 29.8.5. The sum is 0.2069, which is the same amount of error variance for y in the preceding model with measurement error assumed only in x. Hence, the assumption of the measurement error in y does not change the structural effect of Fx on y (same amount of effect Fx on Fy, which is 1.1791). It only changes the variance components of y. In the preceding model with measurement error assumed only in x, the total error variance in y is 0.2069. In the current model, this total error variance is partitioned into the measurement error variance (which is fixed at 0.022) and the error variance in the regression on Fx (which is estimated at 0.1849).

Linear Regression Model as a Special Case of Structural Equation Model

By using the current measurement error model as an illustration, it is easy to see that the structural equation model is a more general model that includes the linear regression model as a special case. If you restrict the measurement error variances in x and y to zero, the measurement error model (which represents the structural equation model in this example) reduces to the linear regression model. That is, the path diagram becomes:

![Path diagram](image)

You can then specify the PATH model by the following statements:

```plaintext
proc calis data=measures;
   path
      x <= Fx = 1.,
      Fx ==> Fy ,
      Fy ==> y = 1.;
   pvar
      x = 0.,
      y = 0.,
      Fx Fy;
run;
```
Output 29.8.6 shows the estimation results of this measurement error model with zero measurement errors. The estimate of the regression coefficient is 1.1511, which is essentially the same result as in Output 29.8.2 by using PROC REG.

Output 29.8.6 Estimates of the Measurement Error Model with Zero Measurement Errors

| Path       | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|------------|-----------|----------|----------------|---------|------|---|
| x          | Fx        | 1.00000  |                |         |      |   |
| Fx         | Fy_Parm1  | 1.15112  | 0.10016        | 11.4924 | <0.001 |   |
| Fy         | y         | 1.00000  |                |         |      |   |

This example shows that you can apply PROC CALIS to fit measurement error models. You treat true scores variables as latent variables in the structural equation model. The linear regression model is a special case of the structural equation model (or measurement error model) where measurement error variances are assumed to be zero. Structural equation modeling by PROC CALIS is not limited to this simple modeling scenario. PROC CALIS can treat more complicated measurement error models. In Example 29.9 and Example 29.10, you fit measurement error models with parameter constraints and with more than one predictor. You can also fit measurement error models with correlated errors.

Example 29.9: Testing Specific Measurement Error Models

In Example 29.8, you used the PATH modeling language of PROC CALIS to fit some basic measurement error models. In this example, you continue to fit the same kind of measurement error models but you restrict some model parameters to test some specific hypotheses.

This example uses the same data set as is used in Example 29.8. This data set contains 30 observations for the x and y variables. The general measurement error model with measurement errors in both x and y is shown in the following path diagram:

```
  x   1.  Fx   1.  Fy   y
```

In the path diagram, two paths are fixed with a path coefficient of 1. They are required in the model for representing the relationships between true scores (latent) and measured indicators (observed). In Example 29.8, you consider several different modeling scenarios, all of which require you to make some parameter restrictions to estimate the models. You fix the measurement error variances to certain values that
are based on prior knowledge or studies. Without those fixed error variances, those models would have been overparameterized and the parameters would not have been estimable.

For example, in the current path diagram, five of the single- or double-headed paths are not labeled with fixed numbers. Each of these paths represents a free parameter in the model. However, in the covariance structure model analysis, you fit these free parameters to the three nonredundant elements of the sample covariance matrix, which is a $2 \times 2$ symmetric matrix. Hence, to have an identified model, you can at most have three free parameters in your covariance structure model. However, the path diagram shows that you have five free parameters in the model. You must introduce additional parameter constraints to make the model identified.

If you do not have prior knowledge about the measurement error variances (as those described in Example 29.8), then you might need to make some educated guesses about how to restrict the overparameterized model. For example, if $x$ and $y$ are of the same kind of measurements, perhaps you can assume that they have an equal amount of measurement error variance. Furthermore, if the measurement errors have been taken into account, in some physical science studies you might be able to assume that the relationship between the true scores $F_x$ and $F_y$ is almost deterministic, resulting in a near zero error variance of $F_y$.

The assumptions here are not comparable to prior knowledge or studies about the measurement error variances. If you suppose they are reasonable enough in a particular field, you can use these assumptions to give you an identified model to work with (at least as an exploratory study) when the required prior knowledge is lacking. The following path diagram incorporates these two assumptions in the measurement error model:

In the path diagram, you use $\text{evar}$ to denote the error variances of $x$ and $y$. This implicitly constrains the two error variances to be equal. The error variance of $F_y$ is labeled zero, indicating a fixed parameter value and a deterministic relationship between $x$ and $y$. You can transcribe this path diagram into the following PATH modeling specification:

```plaintext
proc calis data=measures;
  path
    x <== Fx = 1.,
    Fx ===> Fy ,
    Fy ===> y = 1.;
  pvar
    x = evar,
    y = evar,
    Fy = 0.,
    Fx;
run;
```

In the PVAR statement, you specify the same parameter name $\text{evar}$ for the error variances of $x$ and $y$. This way their estimates are constrained to be the same in the estimation. In addition, the error variance for $F_y$ is fixed at zero, which reflects the “near-deterministic” assumption about the relationship between $F_x$ and $F_y$. These two assumptions effectively reduce the overparameterized model by two parameters so that the new model is just-identified and estimable.

Output 29.9.1 shows the estimation results. The estimated effect of $F_x$ on $F_y$ is 1.3028 (standard error = 0.1134). The measurement error variances for $x$ and $y$ are both estimated at 0.0931 (standard error = 0.0244).
Output 29.9.1 Estimates of the Measurement Error Model with Equal Measurement Error Variances

<table>
<thead>
<tr>
<th>Path</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>x &lt;= Fx</td>
<td></td>
<td>1.0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fx ==&gt; Fy</td>
<td>_Parm1</td>
<td>1.30275</td>
<td>0.11336</td>
<td>11.4924</td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td>Fy ==&gt; y</td>
<td></td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Variance Parameters |
|----------------------|----------------|----------------|---------|------|---|
| Type     | Variable | Parameter | Estimate | Standard Error | t Value | Pr > |t |
| Error    | x        | evar      | 0.09307  | 0.02444       | 3.8079  | 0.0001 |
|          | y        | evar      | 0.09307  | 0.02444       | 3.8079  | 0.0001 |
|          | Fy       |           | 0         |                |         |      |   |
| Exogenous| Fx       | _Parm2    | 0.70655  | 0.20962        | 3.3706  | 0.0008 |

Testing the Effect of Fx on Fy

Suppose you are interested in testing the hypothesis that the effect of Fx on Fy (that is, the regression slope) is 1. The following path diagram represents the model under the hypothesis:

Now you label the path from Fx to Fy with a fixed constant 1, which reflects the hypothesis you want to test. You can transcribe the current path diagram easily into the following PROC CALIS specification:

```plaintext
proc calis data=measures;
   path
      x <= Fx = 1.,
      Fx ==> Fy = 1., /* Testing a fixed constant effect */
      Fy ==> y = 1. ;
   pvar
      x = evar,
      y = evar,
      Fy = 0.,
      Fx;
run;
```

Output 29.9.2 shows the model fit chi-square statistic. The model fit chi-square test here essentially is a test of the null hypothesis of the constant effect at 1 because the alternative hypothesis is a saturated model. The chi-square value is 8.1844 (df = 1, p = .0042), which is statistically significant. This means that the hypothesis of constant effect at 1 is rejected.
Output 29.9.2  Fit Summary for Testing Constant Effect

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
</tbody>
</table>

Output 29.9.3 shows the estimates under this restricted model. In the first table of Output 29.9.3, all path effects or coefficients are fixed at 1. In the second table of Output 29.9.3, estimates of the error variances are 0.1255 (standard error = 0.0330) for both x and y. The error variance of Fy is a fixed zero, as required in the hypothesis. The variance estimate of Fx is 0.9205 (standard error = 0.2587).

Output 29.9.3  Estimates of Constant Effect Measurement Error Model

<table>
<thead>
<tr>
<th>PATH List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
</tr>
<tr>
<td>Path</td>
</tr>
<tr>
<td>x</td>
</tr>
<tr>
<td>Fx</td>
</tr>
<tr>
<td>Fy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variance Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance Type</td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Exogenous</td>
</tr>
</tbody>
</table>

Testing a Zero Intercept

Suppose you are interested in testing the hypothesis that the intercept for the regression of Fy on Fx is zero, while the regression effect is freely estimated. Because the intercept parameter belongs to the mean structures, you need to specify this parameter in PROC CALIS to test the hypothesis.

There are two ways to include the mean structure analysis. First, you can include the MEANSTR option in the PROC CALIS statement. Alternatively, you can use the MEAN statement to specify the means and intercepts in the model. The following statements specify the model under the zero intercept hypothesis:

```plaintext
proc calis data=measures;
    path
      x      <= Fx = 1.,            / * regression effect is freely estimated */
      Fx    ==> Fy ,
      Fy    ==> y = 1.;
    pvar
      x = evar,
      y = evar,
      Fy = 0.,
      Fx;
    mean
      x y = 0. 0.             / * Intercepts are zero in the measurement error model */
```
In the PATH statement, the regression effect of $F_x$ on $F_y$ is freely estimated. In the MEAN statement, you specify the means or intercepts of the variables. Each variable in your measurement error model has either a mean or an intercept (but not both) to specify. If a variable is exogenous (independent), you can specify its mean in the MEAN statement. Otherwise, you can specify its intercept in the MEAN statement. Variables $x$ and $y$ in the measurement error model are both endogenous. They are measured indicators of their corresponding true scores $F_x$ and $F_y$. Under the measurement error model, their intercepts are fixed zeros. The intercept for $F_y$ is zero under the current hypothesized model. The mean of $F_x$ is freely estimated under the model. This parameter is specified in the MEAN statement but is not named.

Output 29.9.4 shows the model fit chi-square statistic. The chi-square value is 10.5397 ($df = 1$, $p = .0012$), which is statistically significant. This means that the zero intercept hypothesis for the regression of $F_y$ on $F_x$ is rejected.

**Output 29.9.4** Fit Summary for Testing Zero Intercept

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
</tbody>
</table>

Output 29.9.5 shows the estimates under the hypothesized model. The effect of $F_x$ on $F_y$ is 1.8169 (standard error = 0.0206). In the last table of Output 29.9.5, the estimate of the mean of $F_x$ is 6.9048 (standard error = 0.1388). The intercepts for all other variables are fixed at zero under the hypothesized model.

**Output 29.9.5** Estimates of the Zero Intercept Measurement Error Model

<table>
<thead>
<tr>
<th>Path List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>$x$</td>
</tr>
<tr>
<td>$F_x$</td>
</tr>
<tr>
<td>$F_y$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variance Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Exogenous</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Means and Intercepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Intercept</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Mean</td>
</tr>
</tbody>
</table>
Measurement Model with Means and Intercepts Freely Estimated

In the preceding model, you fit a restricted regression model with a zero intercept. You reject the null hypothesis and conclude that this intercept is significantly different from zero. The alternative hypothesis is a saturated model with the intercept freely estimated. The model under the alternative hypothesis is specified in the following statements:

```sas
proc calis data=measures;
  path
    x  <= Fx  = 1.,
    Fx  ===> Fy ,
    Fy  ===> y  = 1.;
  pvar
    x  = evar,
    y  = evar,
    Fy  = 0.,
    Fx;
  mean
    x  y  = 0. 0.,
    Fy  Fx;
run;
```

Output 29.9.6 shows that model fit chi-square statistic is zero. This is expected because you are fitting a measurement error model with saturated mean and covariance structures.

**Output 29.9.6** Fit Summary of the Saturated Measurement Model with Mean Structures

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>0.0000</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>0</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>.</td>
</tr>
</tbody>
</table>

Output 29.9.7 shows the estimates under the measurement model with saturated mean and covariance structures. The effect of Fx on Fy is 1.3028 (standard error=0.1134), which is considerably smaller than the corresponding estimate in the restricted model with zero intercept, as shown in Output 29.9.5. The intercept estimate of Fy is 3.5800 (standard error = 0.7864), with a significant t value of 4.55.

**Output 29.9.7** Estimates of the Saturated Measurement Model with Mean Structures

<table>
<thead>
<tr>
<th>PATH List</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>Parameter</td>
<td>Estimate</td>
<td>Standard Error</td>
<td>t Value</td>
<td>Pr &gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x  &lt;= Fx</td>
<td></td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fx  ===&gt; Fy</td>
<td>_Parm1</td>
<td>1.30275</td>
<td>0.11336</td>
<td>11.4924</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fy  ===&gt; y</td>
<td></td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variance Parameters</th>
<th></th>
<th></th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance Type</td>
<td>Variable</td>
<td>Parameter</td>
<td>Estimate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>x</td>
<td>evar</td>
<td>0.09307</td>
<td>0.02444</td>
<td>3.8079</td>
<td>0.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>evar</td>
<td>0.09307</td>
<td>0.02444</td>
<td>3.8079</td>
<td>0.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fy</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Exogenous Fx</td>
<td>_Parm2</td>
<td>0.70655</td>
<td>0.20962</td>
<td>3.3706</td>
<td>0.0008</td>
<td></td>
</tr>
</tbody>
</table>
In this example, you fit some measurement error models with some parameter constraints that reflect the hypothesized models of interest. You can set equality constraints by simply providing the same parameter names in the PATH model specification of PROC CALIS. You can also fix parameters to constants. In the MEAN statement, you can specify the intercepts and means of the variables in the measurement error models. You can apply all these techniques to more complicated measurement error models with multiple predictors, as shown in Example 29.10, where you also fit measurement error models with correlated errors.

**Example 29.10: Measurement Error Models with Multiple Predictors**

In Example 29.8 and Example 29.9, you fit various measurement error models with only one predictor. This example illustrates the case in which you have more than one predictor, all measured with errors. The measurement errors might also be correlated.

The data from 37 observations are summarized in a covariance matrix as shown in the following SAS DATA step:

```sas
data multiple(type=cov);
    input _type_ $ 1-4 _name_ $ 6-8 @10 y x1 x2 x3;
datalines;
mean y 0.93 1.33 1.34 4.11
cov y 1.31 . . .
cov x1 1.24 1.42 . .
cov x2 0.21 0.18 1.15 .
cov x3 3.91 4.21 0.58 14.11
;
```

In this data set, four variables are measured. Variables x1–x3 are predictors of y. Instead of the raw data, you can input the sample covariance matrix in the form of a SAS data set for PROC CALIS to analyze.

You assume all of these variables in the data set are measured with errors. From prior studies, you establish the knowledge about the measurement errors of these variables. You create the true score counterparts for each of these variables in the same manner as you do in Example 29.8 and Example 29.9. The following path diagram represents your measurement error model for the data:
In the path diagram, variables F1–F3 and Fy are latent variables that represent the true score for the measured indicators x1–x3 and y, respectively. All paths from the true scores to the corresponding measured indicators are labeled with the fixed constant 1, as required by the measurement model. Each measured indicator is attached with a double-headed arrow that indicates the error variance. Because you have knowledge about these measurement error variances, you put fixed constant values adjacent to these double-headed arrows. For example, the measurement error variance of y is 0.02 and the measurement error variance of x3 is 0.15. The path diagram also indicates that the paths from F1–F3 to Fy and the error variance for Fy are free parameters to estimate in the model.

Notice that for brevity the variances and covariances among the three exogenous true score variables F1–F3 are not represented in the path diagram. These six variance and covariance parameters could have been represented by double-headed arrows in the path diagram. However, because PROC CALIS always assumes the exogenous variances and covariances as default model parameters, this information is not represented to reduce clutter in the path diagram.

You can transcribe the path diagram easily to the following PATH model specification:

```sas
proc calis data=multiple nobs=37;
    path
        Fy <= F1 F2 F3,
        F1 ==> x1 = 1.,
        F2 ==> x2 = 1.,
        F3 ==> x3 = 1.,
        Fy ==> y = 1.;
pvar
        x1 x2 x3 y = .02 .03 .15 .02,
    run;
```

In the first entry of the PATH statement, you specify that F1–F3 predicts Fy. In the next four path entries you specify the measurement model for the true scores and how they are related to the observed variables. In the PVAR statement, you specify all the known measurement error variances for the observed variables. They are all fixed constants in the model. In the last entry in the PVAR statement, you specify the error variance of Fy as a free (unnamed) parameter. You could have omitted this entry because error variances for all endogenous variables in the PATH model are free parameters by default. Setting these default parameters explicitly as free parameters would not affect model fitting.

Output 29.10.1 shows the parameter estimates of the model. The path coefficient or effect from F2 to Fy is not significant, while the other two path coefficients are at least marginally significant.
Chapter 29: The CALIS Procedure

Output 29.10.1 Parameter Estimates of the Measurement Model with Multiple Predictors

<table>
<thead>
<tr>
<th>Path</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fy &lt;= F1</td>
<td>_Parm1</td>
<td>0.46507</td>
<td>0.22682</td>
<td>2.0503</td>
<td>0.0403</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fy &lt;= F2</td>
<td>_Parm2</td>
<td>0.04123</td>
<td>0.07069</td>
<td>0.5832</td>
<td>0.5597</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fy &lt;= F3</td>
<td>_Parm3</td>
<td>0.13812</td>
<td>0.07175</td>
<td>1.9249</td>
<td>0.0542</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1 ==&gt; x1</td>
<td></td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2 ==&gt; x2</td>
<td></td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F3 ==&gt; x3</td>
<td></td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fy ==&gt; y</td>
<td></td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variance Type</th>
<th>Variable</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>x1</td>
<td></td>
<td>0.02000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>x2</td>
<td></td>
<td>0.03000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>x3</td>
<td></td>
<td>0.15000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>y</td>
<td></td>
<td>0.02000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fy</td>
<td>_Parm4</td>
<td></td>
<td>0.16461</td>
<td>0.04522</td>
<td>3.6403</td>
<td>0.0003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exogenous</td>
<td>F1</td>
<td>_Add1</td>
<td>1.40000</td>
<td>0.33470</td>
<td>4.1829</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F2</td>
<td>_Add2</td>
<td>1.12000</td>
<td>0.27106</td>
<td>4.1320</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F3</td>
<td>_Add3</td>
<td>13.96000</td>
<td>3.32576</td>
<td>4.1975</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Covariances Among Exogenous Variables</th>
<th>Var1</th>
<th>Var2</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F2</td>
<td>F1</td>
<td>_Add4</td>
<td>0.18000</td>
<td>0.21508</td>
<td>0.8369</td>
<td>0.4027</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F3</td>
<td>F1</td>
<td>_Add5</td>
<td>4.21000</td>
<td>1.02416</td>
<td>4.1107</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>F3</td>
<td>F2</td>
<td>_Add6</td>
<td>0.58000</td>
<td>0.67829</td>
<td>0.8551</td>
<td>0.3925</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The second table of Output 29.10.1 shows the variance estimates. As specified in the model, all measurement error variances for the observed variables are fixed constants. The error variance of Fy is 0.1646 (standard error =0.0452). Although you do not specify them in the PATH model specification, variances of F1–F3 are free parameters in the model. The second table of Output 29.10.1 shows their estimates. The last table of Output 29.10.1 shows the covariances among the latent true scores. Only the covariance between F3 and F1 is significant.

PROC CALIS not only can handle measurement error variance with multiple true score predictors, but it also can handle correlated errors. Suppose that the measurement errors for variables x1 and x2 are correlated. From prior studies, you determine that their covariance is 0.01. The path diagram with this new piece of information added is shown in the following:
Example 29.10: Measurement Error Models with Multiple Predictors

In the path diagram, the double-headed arrow that connects $x_1$ and $x_2$ represents the covariance between the error terms for the two variables. The value attached to this double-headed arrow is 0.01, which represents a fixed constant in the model. The PATH model specification is similar to the preceding specification, with one more entry added in the PCOV statement, as shown in the following statements:

```plaintext
proc calis data=multiple nobs=37;
  path
    Fy <=== F1 F2 F3,
    F1 ===> x1 = 1.,
    F2 ===> x2 = 1.,
    F3 ===> x3 = 1.,
    Fy ===> y = 1.;
  pvar
    x1 x2 x3 y = .02 .03 .15 .02,
    Fy;
  pcov
    x1 x2 = 0.01;
run;
```

Except for the PCOV statement specification, everything else is the same as in the preceding specification. In the PCOV statement, you can specify covariance or error covariances between exogenous or endogenous variables. In the current model, because both $x_1$ and $x_2$ are endogenous in the model, the specification is for their error covariance, which is fixed at 0.01 as required.

Output 29.10.2 shows the parameter estimates of the measurement model with correlated errors. The estimates do not change much from the preceding analysis in which correlated errors is not assumed. Perhaps the correlation between the errors in the current model is so small that it is ignorable. The last table in Output 29.10.2 shows the covariance estimates among errors. This table is unique to the current model. It shows that the measurement errors for $x_1$ and $x_2$ have a covariance of 0.01, which is treated as a fixed constant in the current model.
Output 29.10.2  Parameter Estimates of the Measurement Model with Multiple Predictors: Correlated Errors

| PATH List                  | Standard Error | t Value | Pr > |t| |
|---------------------------|----------------|---------|------|---|
| Path          | Parameter | Estimate |        |     |   |
| Fy <= F1     | _Parm1    | 0.46839  | 0.22695| 2.0639| 0.0390|
| Fy <= F2     | _Parm2    | 0.04549  | 0.07074| 0.6431| 0.5202|
| Fy <= F3     | _Parm3    | 0.13694  | 0.07194| 1.9035| 0.0570|
| F1 => x1     |           | 1.00000           |       |
| F2 => x2     |           | 1.00000           |       |
| F3 => x3     |           | 1.00000           |       |
| Fy => y      |           | 1.00000           |       |

| Variance Parameters                  | Standard Error | t Value | Pr > |t| |
|--------------------------------------|----------------|---------|------|---|
| Type       | Variable | Parameter | Estimate |        |     |   |
| Error      | x1       |           | 0.02000 |       |       |
|           | x2       |           | 0.03000 |       |       |
|           | x3       |           | 0.15000 |       |       |
|           | y        |           | 0.02000 |       |       |
| Fy        | _Parm4   |           | 0.16421 | 0.04523| 3.6305| 0.0003|
| Exogenous | F1       | _Add1     | 1.40000 | 0.33470| 4.1829| <.0001|
|           | F2       | _Add2     | 1.12000 | 0.27106| 4.1320| <.0001|
|           | F3       | _Add3     | 13.96000| 3.32576| 4.1975| <.0001|

| Covariances Among Exogenous Variables | Standard Error | t Value | Pr > |t| |
|--------------------------------------|----------------|---------|------|---|
| Var1       | Var2     | Parameter | Estimate | Error | t Value | Pr > |t| |
| F2         | F1       | _Add4     | 0.17000 | 0.21508| 0.7904| 0.4293|
| F3         | F1       | _Add5     | 4.21000 | 1.02416| 4.1107| <.0001|
| F3         | F2       | _Add6     | 0.58000 | 0.67829| 0.8551| 0.3925|

| Covariances Among Errors | Standard Error | t Value | Pr > |t| |
|--------------------------|----------------|---------|------|---|
| Error of Error           | Estimate | Error | t Value | Pr > |t| |
| x1                       | x2       | 0.01000 |       |       |   |   |

This example shows how you can use PROC CALIS to fit measurement error models with multiple true score predictors. You can also fit models with correlated errors. The model specification tool is the PATH modeling language, which ties closely to the path diagram representations.

However, some researchers might prefer to use linear equations to represent the measurement error models. PROC CALIS provides the LINEQS modeling language for specifying the measurement error models, or mean and covariance structure models in general. Example 29.11 illustrates the LINEQS model specification of the measurement error models.
Example 29.11: Measurement Error Models Specified As Linear Equations

In Example 29.8, you fit a simple measurement error model with errors in both of the predictor variable \( x \) and the outcome variable \( y \). From prior studies, the measurement error variance of \( x \) is 0.019 and the measurement error variance of \( y \) is 0.022. You use the following path diagram to represent the model:

With this path diagram, you use the PATH modeling language of PROC CALIS to specify the model, as shown in the following:

```plaintext
proc calis data=measures;
  path
    x <=== Fx = 1.,
    Fx ===> Fy ,
    Fy ===> y = 1.;
  pvar
    x = 0.019,
    y = 0.022,
    Fx Fy;
run;
```

In the path diagram and in the PATH model specification, there are no explicit representations of the error terms in the model. You express the error variances of \( x \), \( y \), and \( Fy \) as partial variances of the endogenous variables. In the path diagram, you represent these partial variances by the double-headed arrows. Correspondingly, in the PATH statement of PROC CALIS, you specify these partial variances in the PVAR statement.

In practice, some researchers might prefer to express the error terms in the model explicitly. For example, with the error terms added to the preceding measurement error model, the new path diagram becomes:

In the path diagram, you add paths from error variables \( e1 \), \( e2 \), and \( d \) to the endogenous variables \( x \), \( y \), and \( Fy \), respectively. All these paths from the error terms have a fixed path coefficient of 1. The error variances are represented by double-headed arrows directly attached to them. For example, the variance of \( e1 \) is fixed at 0.019, and the variance of \( e2 \) is fixed at 0.022. The variance of \( d \), which is sometime called the disturbance, is a free unnamed parameter in the path diagram. Similarly, the variance of \( Fx \) is a free unnamed parameter in the model.

Corresponding to this new path diagram, you can use the LINEQS modeling language for specifying your model in PROC CALIS, as shown in the following statements:
Chapter 29: The CALIS Procedure

```sas
proc calis data=measures;
  lineqs
    x = 1. * Fx + e1,
    y = 1. * Fy + e2,
    Fy = * Fx + d;
  variance
    e1 = 0.019,
    e2 = 0.022,
    Fx d;
run;
```

The LINEQS model specification in PROC CALIS emphasizes the linear equation input. In each of the linear equations in the LINEQS statement, you specify an endogenous variable and how it is related to other variables. An endogenous variable in the path diagram is a variable that has at least one single-headed arrow pointing to it. You need to list all endogenous variables on the left-hand side of the linear equations of the LINEQS statement. In the current model, variables `x`, `y`, and `Fy` are endogenous, and therefore you specify three linear equations in the LINEQS statement. The first two equations represent the measurement model for the observed variables, while the third equation represents the structural equation of the model. Notice that in the third equation, you do not specify the path coefficient that is attached to `Fx`. PROC CALIS treats unspecified path coefficients as free parameters. The effect of `Fx` on `Fy` is freely estimated, as required in the path diagram representation.

In the VARIANCE statement, you specify the variances of the exogenous variables in the model. The specifications in the VARIANCE statement of the LINEQS model are very similar to those in the PVAR statement of the PATH model. The main difference is the use of error variable names in the VARIANCE statement. With the LINEQS model specification, you can only specify exogenous variables in the VARIANCE statement. Hence, you must specify the error variables `e1`, `e2`, and `d` in the VARIANCE statement of the LINEQS model, instead of the corresponding endogenous variables `x`, `y`, and `Fy` in the PVAR statement of the PATH model.

Output 29.11.1 shows the parameter estimates of the LINEQS model.

**Output 29.11.1** LINEQS Parameter Estimates of the Measurement Model for the Measures Data

<table>
<thead>
<tr>
<th>Linear Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x</code> = 1.0000 * <code>Fx</code> + 1.0000 * <code>e1</code></td>
</tr>
<tr>
<td><code>y</code> = 1.0000 * <code>Fy</code> + 1.0000 * <code>e2</code></td>
</tr>
<tr>
<td><code>Fy</code> = 1.1791 * <code>Fx</code> + 1.0000 * <code>d</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Effects in Linear Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td><code>x</code></td>
</tr>
<tr>
<td><code>y</code></td>
</tr>
<tr>
<td><code>Fy</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Estimates for Variances of Exogenous Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Type</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Latent</td>
</tr>
<tr>
<td>Disturbance</td>
</tr>
</tbody>
</table>
All these estimates are essentially the same as those obtained from the PATH model specification, as shown in Output 29.11.2.

**Output 29.11.2** PATH Parameter Estimates of the Measurement Model for the Measures Data

<table>
<thead>
<tr>
<th>Path</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Fx</td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fx</td>
<td>Fy _Parm1</td>
<td>1.17914</td>
<td>0.10288</td>
<td>11.4615</td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td>Fy</td>
<td>y</td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can use either the LINEQS or PATH model specification in PROC CALIS for your analysis problems. They give you the same estimation results.

So far the measurement error model is concerned with one predictor. With more predictors in the model, you might also want to model the correlated measurement errors in the x variables. You can analyze this kind of model by using the PATH model specification, as shown in Example 29.10. With measurement error terms explicitly assumed, you can also use the LINEQS model specification. This example illustrates how you can do that by using the same data set and the measurement error model with correlated errors in Example 29.10.

In the data set, you have four observed variables: x1–x3 and y. All are measured with errors, as represented by the following path diagram:

In the path diagram, F1–F3 and Fy represent true scores for the measurement indicators x1–x3 and y, respectively. You predict Fy by F1–F3, which represents the structural relationships in the model. Measurement error variances of the observed variables are treated as known and are represented by the double-headed arrows attached to the observed variables. For example, the error variance of x3 is 0.15. In addition, the error covariance between x1 and x2 is treated as known. The double-headed arrow that connects x1 and x2 represents the error covariance, and this covariance is fixed at 0.01 in the model.

You transcribe this path diagram representation into the following PATH model specification:
proc calis data=multiple nobs=37;
  path
    Fy <= F1 F2 F3,
    F1 ===> x1 = 1.,
    F2 ===> x2 = 1.,
    F3 ===> x3 = 1.,
    Fy ===> y = 1.;
  pvar
    x1 x2 x3 y = .02 .03 .15 .02,
    Fy;
  pcov
    x1 x2 = 0.01;
run;

To represent the error terms explicitly, you can add the error terms to the path diagram with some modifications, as shown in the following:

In the path diagram, you attach error variables e1–e3, ey, and d to the associated endogenous variables in the model. The error variances and covariances, which are attached to the endogenous variables directly, are now attached to the corresponding error variables. With this new path diagram, you can use the following LINEQS model specification for the model:

proc calis data=multiple nobs=37;
  lineqs
    Fy = * F1 + * F2 + * F3 + d,
    x1 = 1. * F1 + e1,
    x2 = 1. * F2 + e2,
    x3 = 1. * F3 + e3,
    y = 1. * Fy + ey;
  variance
    e1-e3 ey = .02 .03 .15 .02,
    d;
  cov
    e1 e2 = 0.01;
run;

Again, in each linear equation of the LINEQS statement, you specify the functional relationship of an endogenous variable with other variables, including the error variable. The first equation is the structural equation in the model. You want to estimate the effects of F1, F2, and F3 on Fy. The error or disturbance variable is d. In the next four equations, you relate the observed variables with their true scores counterparts.

In the VARIANCE statement, you specify the error variances with reference to the error variables in the path diagram. Four of the error variances are fixed constants, as required in the model. The last specification
represents a free parameter for the variance of \( d \). The specifications in the VARIANCE statement of the LINEQS model are similar to those in the PVAR statement of the PATH model specification. The difference is that in the PATH model specification the reference variables are the endogenous variables in the PATH model, while in the LINEQS model specification the reference variables are the associated error variables.

In the COV statement, you specify the covariance between the error variables \( e_1 \) and \( e_2 \). Again, this is similar to the corresponding specification of the PATH model, where the same error covariance is specified as the partial covariance between \( x_1 \) and \( x_2 \) in the PCOV statement.

Output 29.11.3 shows the parameter estimates that result from using the LINEQS model specification. Estimates in the equations, variances, and covariances are shown respectively.

### Output 29.11.3 Parameter Estimates of the Measurement Model with Multiple Predictors: LINEQS Model

#### Linear Equations

| Variable | Predictor | Parameter | Estimate | Standard Error | \( t \) Value | \( Pr > |t| \) |
|----------|-----------|-----------|----------|----------------|--------------|-------------|
| \( F_y \) | \( F_1 \) | _Parm1 | 0.46839 | 0.22695 | 2.0639 | 0.0390 |
| \( F_y \) | \( F_2 \) | _Parm2 | 0.04549 | 0.07074 | 0.6431 | 0.5202 |
| \( F_y \) | \( F_3 \) | _Parm3 | 0.13694 | 0.07194 | 1.9035 | 0.0570 |
| \( x_1 \) | \( F_1 \) |     | 1.00000 |       |       |       |
| \( x_2 \) | \( F_2 \) |     | 1.00000 |       |       |       |
| \( x_3 \) | \( F_3 \) |     | 1.00000 |       |       |       |
| \( y \)   | \( F_y \) |     | 1.00000 |       |       |       |

#### Effects in Linear Equations

#### Estimates for Variances of Exogenous Variables

| Type       | Variable | Parameter | Estimate | Standard Error | \( t \) Value | \( Pr > |t| \) |
|------------|----------|-----------|----------|----------------|--------------|-------------|
| Error      | \( e_1 \) |     | 0.02000 |       |       |       |
|            | \( e_2 \) |     | 0.03000 |       |       |       |
|            | \( e_3 \) |     | 0.15000 |       |       |       |
|            | \( e_y \) |     | 0.02000 |       |       |       |
| Disturbance| \( d \)  | _Parm4 | 0.16421 | 0.04523 | 3.6305 | 0.0003 |
| Latent     | \( F_1 \) | _Add1 | 1.40000 | 0.33470 | 4.1829 | <.0001 |
|            | \( F_2 \) | _Add2 | 1.12000 | 0.27106 | 4.1320 | <.0001 |
|            | \( F_3 \) | _Add3 | 13.96000 | 3.32576 | 4.1975 | <.0001 |

#### Covariances Among Exogenous Variables

| Var1 | Var2 | Parameter | Estimate | Standard Error | \( t \) Value | \( Pr > |t| \) |
|------|------|-----------|----------|----------------|--------------|-------------|
| \( e_1 \) | \( e_2 \) |     | 0.01000 |       |       |       |
| \( F_2 \) | \( F_1 \) | _Add4 | 0.17000 | 0.21508 | 0.7904 | 0.4293 |
| \( F_3 \) | \( F_1 \) | _Add5 | 4.21000 | 1.02416 | 4.1107 | <.0001 |
| \( F_3 \) | \( F_2 \) | _Add6 | 0.58000 | 0.67829 | 0.8551 | 0.3925 |
Output 29.11.4 shows the parameter estimates that result from using the PATH model specification. The estimates in the path list shown in Output 29.11.4 correspond to those of the equation output in Output 29.11.3. The variance estimates in Output 29.11.4 correspond to those variance estimates of the exogenous variables of the LINEQS model, as shown in Output 29.11.3. Finally, the last two tables in Output 29.11.4 correspond to the covariance estimates among the exogenous variables of the LINEQS model, as shown in Output 29.11.3. Again, the LINEQS and PATH model specification give you exactly the same estimation results, but in different output formats.

Output 29.11.4  Parameter Estimates of the Measurement Model with Multiple Predictors: PATH Model

| PATH List | Standard Error | t Value | Pr > |t| |
|-----------|----------------|---------|------|---|
| Fy <== F1  | Parm1          | 0.46839 | 0.22695 | 2.0639 | 0.0390 |
| Fy <== F2  | Parm2          | 0.04549 | 0.07074 | 0.6431 | 0.5202 |
| Fy <== F3  | Parm3          | 0.13694 | 0.07194 | 1.9035 | 0.0570 |
| F1 ==> x1  |               | 1.00000 |         |      |      |
| F2 ==> x2  |               | 1.00000 |         |      |      |
| F3 ==> x3  |               | 1.00000 |         |      |      |
| Fy ==> y   |               | 1.00000 |         |      |      |

| Variance Parameters | Standard Error | t Value | Pr > |t| |
|----------------------|----------------|---------|------|---|
| Error                | x1             | 0.02000 |       |      |
|                      | x2             | 0.03000 |       |      |
|                      | x3             | 0.15000 |       |      |
|                      | y              | 0.02000 |       |      |
| Fy                   | Parm4          | 0.16421 | 0.04523 | 3.6305 | 0.0003 |

| Exogenous | Standard Error | t Value | Pr > |t| |
|-----------|----------------|---------|------|---|
| F1        | Add1           | 1.40000 | 0.33470 | 4.1829 | <.0001 |
| F2        | Add2           | 1.12000 | 0.27106 | 4.1320 | <.0001 |
| F3        | Add3           | 13.96000 | 3.32576 | 4.1975 | <.0001 |

| Covariances Among Exogenous Variables | Standard Error | t Value | Pr > |t| |
|--------------------------------------|----------------|---------|------|---|
| F2 F1 Add4                           | 0.17000        | 0.21508 | 0.7904 | 0.4293 |
| F3 F1 Add5                           | 4.21000        | 1.02416 | 4.1107 | <.0001 |
| F3 F2 Add6                           | 0.58000        | 0.67829 | 0.8551 | 0.3925 |

| Covariances Among Errors | Standard Error | t Value | Pr > |t| |
|--------------------------|----------------|---------|------|---|
| x1 x2                    | 0.01000        |         |      |      |

In this example, you fit measurement error models by using the LINEQS and PATH model specifications of PROC CALIS. The two different model specification languages give you essentially the same estimation results. The measurement models can have multiple true scores predictors and correlated errors. The measurement error models considered so far have only one measured indicator for each true score latent variable. This is usually not the case in many psychometric or sociological applications where latent factors usually have several observed indicators. The confirmatory factor model is a typical example of this kind of
Example 29.12: Confirmatory Factor Models

This example shows how you can fit a confirmatory factor analysis model by the FACTOR modeling language. Thirty-two students take tests of their verbal and math abilities. Six tests are administered separately. Tests $x_1$–$x_3$ test their verbal skills and tests $y_1$–$y_3$ test their math skills.

The data are shown in the following DATA step:

```plaintext
data scores;
  input x1 x2 x3 y1 y2 y3;
datalines;
23 17 16 15 14 16
29 26 23 22 18 19
14 21 17 15 16 18
20 18 17 18 21 19
25 26 22 26 21 26
26 19 15 16 17 17
14 17 19 4 6 7
12 17 18 14 16 13
25 19 22 22 20 20
7 12 15 10 11 8
29 24 30 14 13 16
28 24 29 19 19 21
12 9 10 18 19 18
11 8 12 15 16 16
20 14 15 24 23 16
26 25 21 24 23 24
20 16 19 22 21 20
14 19 15 17 19 23
14 20 13 24 26 25
29 24 24 21 20 18
26 28 26 28 26 23
20 23 24 22 23 22
23 24 20 23 22 18
14 18 17 13 16 14
28 34 27 25 21 21
17 12 10 14 12 16
8 1 13 14 15 14
22 19 19 13 11 14
18 21 18 15 18 19
12 12 10 13 13 16
22 14 20 20 18 19
29 21 22 13 17 12
;
```

Because of the unambiguous nature of the tests, you hypothesize that this is a confirmatory factor model with two factors: one is the verbal ability factor and the other is the math ability factor. You can represent such a confirmatory factor model by the following path diagram:
In the path diagram, there are two clusters of variables. One cluster is for the *verbal* factor and the other is for the *math* factor. The single-headed arrows in the path diagram represent functional relationships between factors and the observed variables. The double-headed arrows that point to single variables represent variances of the factors or error variances of the observed variables. The double-headed arrow that connect the two factors represents their covariance. All but two of these arrows are not labeled with numbers. Each of the unlabeled arrows represents a free parameter in the confirmatory factor model. You label the double-headed arrows that attach to the two factors with the constant 1. This means that the variances of the factors are fixed at 1.0 in the model.

You can specify the confirmatory factor model by the FACTOR model language of PROC CALIS, as shown in the following statements:

```plaintext
proc calis data=scores;
  factor
    verbal ===> x1-x3,
    math ===> y1-y3;
  pvar
    verbal = 1.,
    math = 1.;
run;
```

In each of the entry of the FACTOR statement, you specify a latent factor, followed by a list of observed variables that are functionally related to the latent factor. For example, in the first entry, the *verbal* factor is related to variables \(x_1-x_3\), as shown by the single-headed arrows in the path diagram. In fact, all single-headed arrows in the path diagram are specified in the FACTOR statement. Notice that each entry of the FACTOR statement must take the format of

```
factor_name ===> variable_list
```

You cannot reverse the arrow specification as in the following:

```
variable_list <== factor_name
```

Nor you can have a specification such as the following:
variable_list ===> factor_name

However, you can specify the functional relationships between factors and variables in different entries. For example, you can specify the same confirmatory factor model by the following statements:

```
title "Basic Confirmatory Factor Model: Separate Path Entries";
title2 "FACTOR Model Specification";
proc calis data=scores;
  factor
    verbal ===> x1,
    verbal ===> x2,
    verbal ===> x3,
    math ===> y1,
    math ===> y2,
    math ===> y3;
  pvar
    verbal = 1.,
    math = 1.;
  fitindex noindextype on(only)=\{chisq df probchi rmsea srmr bentlercfi\};
run;
```

In the PVAR statement, which is for the specification of variances or error variances, you fix the variances of the latent factors to 1. This completes the model specification of the confirmatory factor model, although you do not specify other arrows in the path diagram as free parameters in these statements. The reason is that in the FACTOR modeling language, the variances and covariances among factors and the error variances of the observed variables are default parameters in the confirmatory factor model. It is not necessary to specify these parameters (or the corresponding arrows in the path diagram) explicitly if they are free parameters in the model. You can also specify these free parameters explicitly without affecting the estimation. However, if these parameters (or the corresponding double-headed arrows in the path diagram) are intended to be constrained parameters or fixed values, you must specify them explicitly. For example, in the current confirmatory factor model, you must provide explicit specifications for the variances of the verbal and the math factors because these parameters are fixed at 1.

Output 29.12.1 shows the modeling information and the variables in the confirmatory factor model.

**Output 29.12.1** Modeling Information and Variables of the CFA Model: Scores Data

**Simple Confirmatory Factor Model**

**FACTOR Model Specification**

<table>
<thead>
<tr>
<th>The CALIS Procedure</th>
<th>Covariance Structure Analysis: Model and Initial Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling Information</td>
<td>Maximum Likelihood Estimation</td>
</tr>
<tr>
<td>Data Set</td>
<td>WORK.SCORES</td>
</tr>
<tr>
<td>N Records Read</td>
<td>32</td>
</tr>
<tr>
<td>N Records Used</td>
<td>32</td>
</tr>
<tr>
<td>N Obs</td>
<td>32</td>
</tr>
<tr>
<td>Model Type</td>
<td>FACTOR</td>
</tr>
<tr>
<td>Analysis</td>
<td>Covariances</td>
</tr>
</tbody>
</table>
In the beginning of the output, PROC CALIS shows the data set, the number of observations, the model type, and the analysis type. The default analysis type in PROC CALIS is covariances (that is, covariance structures). If you want to analyze the correlation structures instead, you can use the CORR option in the PROC CALIS statement. Next, PROC CALIS shows the list of variables and factors in the model. As expected, the number of variables is 6 and the number of factors is 2.

Output 29.12.2 shows the initial model specifications of the confirmatory factor model.

**Output 29.12.1 continued**

<table>
<thead>
<tr>
<th>Variables in the Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
</tr>
<tr>
<td>Factors</td>
</tr>
<tr>
<td>Number of Variables = 6</td>
</tr>
<tr>
<td>Number of Factors = 2</td>
</tr>
</tbody>
</table>

Output 29.12.2 Initial Specification of the CFA Model: Scores Data

<table>
<thead>
<tr>
<th>Initial Factor Loading Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbal</td>
</tr>
<tr>
<td>x1</td>
</tr>
<tr>
<td>[ Parm1]</td>
</tr>
<tr>
<td>x2</td>
</tr>
<tr>
<td>[ Parm2]</td>
</tr>
<tr>
<td>x3</td>
</tr>
<tr>
<td>[ Parm3]</td>
</tr>
<tr>
<td>y1</td>
</tr>
<tr>
<td>[ Parm4]</td>
</tr>
<tr>
<td>y2</td>
</tr>
<tr>
<td>[ Parm5]</td>
</tr>
<tr>
<td>y3</td>
</tr>
<tr>
<td>[ Parm6]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial Factor Covariance Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbal</td>
</tr>
<tr>
<td>verbal</td>
</tr>
<tr>
<td>[ Add1]</td>
</tr>
<tr>
<td>math</td>
</tr>
<tr>
<td>[ Add1]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial Error Variances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Parameter Estimate</td>
</tr>
<tr>
<td>x1</td>
</tr>
<tr>
<td>x2</td>
</tr>
<tr>
<td>x3</td>
</tr>
<tr>
<td>y1</td>
</tr>
<tr>
<td>y2</td>
</tr>
<tr>
<td>y3</td>
</tr>
</tbody>
</table>

NOTE:
Parameters with prefix '_Add' are added by PROC CALIS.
Example 29.12: Confirmatory Factor Models

The first table of Output 29.12.2 shows the pattern of factor loadings of the variables on the two latent factors. As expected, \(x_1\)–\(x_3\) have nonzero loadings only on the verbal factor, while \(y_1\)–\(y_3\) have nonzero loadings on the math factor. PROC CALIS names these free parameters automatically with the “_Parm” prefix and unique numerical suffixes. There are six parameters in the factor loading matrix with six different parameter names.

The next table of Output 29.12.2 shows the covariance matrix of the factors. The variances of the factors are fixed at one, as shown on the diagonal of the covariance matrix. The covariance between the two factors is a free parameter named _Add1. You did not specify this covariance parameter explicitly in the factor model specification. By default, PROC CALIS assumes that latent factors are correlated. Default free parameters added by PROC CALIS have the _Add prefix for their names. If you do not want to assume the covariances among the factors, you must specify zero covariances in the COV statement. For example, the following statement specifies that the math and verbal factors have zero covariance:

\[
\text{COV} \\
\text{math verbal} = 0.;
\]

The last table of Output 29.12.2 shows the error variance parameters of the observed variables. By default PROC CALIS assumes these error variances are free parameters in the confirmatory factor model. These added parameters are named with the _Add prefix. However, as all other default parameters that are assumed by PROC CALIS, you can overwrite the default by using explicit specifications. You can specify the error variances of a confirmatory factor model explicitly in the PVAR statement. See specifications in Example 29.13.

Output 29.12.3 shows the fit summary of the confirmatory factor model for the scores data.

**Output 29.12.3**  Fit Summary of the CFA Model: Scores Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>9.8052</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>8</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.2790</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0571</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0853</td>
</tr>
<tr>
<td>Bentler Comparative Fit Index</td>
<td>0.9887</td>
</tr>
</tbody>
</table>

The model fit chi-square is 9.805 (\(df = 8, p = 0.279\)). This shows that statistically you cannot reject the confirmatory factor model for the test scores. However, the root mean square error of approximation (RMSEA) estimate is 0.0853, which is greater than the conventional 0.05 value for a good model fit. The standardized root mean square residual (SRMR) is 0.0571, which is close to the conventional 0.05 value for a good model fit. Bentler’s comparative fit index is 0.9887, which indicates a very good model fit. Overall, the model seems to be quite reasonable for the data.

Output 29.12.4 shows the loading and factor covariance estimates of the confirmatory factor model for the scores data. The first table shows the loading estimates, together with the standard error estimates and the \(t\) values. In structural equation modeling, the significance of the parameter estimates is usually inferred by comparing the \(t\) values with the critical value of a standardized normal variate (that is, the \(z\)-table). Therefore, estimates with associated (absolute) \(t\) values greater than 1.96 are significant at \(\alpha = .05\). In Output 29.12.4, all the \(t\) values for the loading estimates are greater than 2. This indicates that the prescribed relationships between the variables and the factors are significant.
Chapter 29: The CALIS Procedure

Output 29.12.4 Loading and Factor Covariance Estimates of the CFA Model: Scores Data

<table>
<thead>
<tr>
<th>Factor Loading Matrix:</th>
<th>Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>verbal</td>
</tr>
<tr>
<td>x1</td>
<td>5.8406</td>
</tr>
<tr>
<td></td>
<td>0.9962</td>
</tr>
<tr>
<td></td>
<td>5.8629</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td></td>
<td>[Parm1]</td>
</tr>
<tr>
<td>x2</td>
<td>5.8182</td>
</tr>
<tr>
<td></td>
<td>0.9537</td>
</tr>
<tr>
<td></td>
<td>6.1004</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td></td>
<td>[Parm2]</td>
</tr>
<tr>
<td>x3</td>
<td>4.6619</td>
</tr>
<tr>
<td></td>
<td>0.7814</td>
</tr>
<tr>
<td></td>
<td>5.9662</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td></td>
<td>[Parm3]</td>
</tr>
<tr>
<td>y1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.6998</td>
</tr>
<tr>
<td></td>
<td>7.5455</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td></td>
<td>[Parm4]</td>
</tr>
<tr>
<td>y2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.6220</td>
</tr>
<tr>
<td></td>
<td>6.7532</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td></td>
<td>[Parm5]</td>
</tr>
<tr>
<td>y3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.6341</td>
</tr>
<tr>
<td></td>
<td>5.9289</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td></td>
<td>[Parm6]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Factor Covariance Matrix:</th>
<th>Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>verbal</td>
</tr>
<tr>
<td>verbal</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>0.1429</td>
</tr>
<tr>
<td></td>
<td>3.6221</td>
</tr>
<tr>
<td></td>
<td>0.000292</td>
</tr>
<tr>
<td>math</td>
<td>0.5175</td>
</tr>
<tr>
<td></td>
<td>0.1429</td>
</tr>
<tr>
<td></td>
<td>3.6221</td>
</tr>
<tr>
<td></td>
<td>0.000292</td>
</tr>
</tbody>
</table>

The second table of Output 29.12.4 shows the covariance matrix of the verbal and the math factors. Because the factor variances are fixed at one, the covariance estimate is also the correlation between the two factors. Output 29.12.4 shows that the two factors are moderately correlated with a correlation estimate of 0.5175, which is statistically significant.

Output 29.12.5 shows the estimates of the error variances. All but the error variance of y1 are significant. This suggests that y1 might have an almost perfect relationship with the math factor.
Output 29.12.5  Error Variance Estimates of the CFA Model: Scores Data

| Variable | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|----------|-----------|----------|----------------|---------|-------|---|
| x1       | _Add2     | 11.52376 | 4.26398        | 2.7026  | 0.0069|
| x2       | _Add3     | 9.14503  | 3.83219        | 2.3864  | 0.0170|
| x3       | _Add4     | 6.68169  | 2.59770        | 2.5722  | 0.0101|
| y1       | _Add5     | 0.78580  | 1.29440        | 0.6071  | 0.5438|
| y2       | _Add6     | 2.88069  | 1.09395        | 2.6333  | 0.0085|
| y3       | _Add7     | 5.15573  | 1.46854        | 3.5108  | 0.0004|

Output 29.12.6 echoes this same fact. The R-squares in this table shows the percentages of variance of the variables that are overlapped with the factors. While all these percentages (0.74 – 0.97) are quite high for all variables, the percentage is especially high for y1. It shares 97% of the variance with the math factor. So, it appears that the observed variable y1 is almost a perfect indicator of the math factor.

Output 29.12.6  Squared Multiple Correlations of the CFA Model: Scores Data

<table>
<thead>
<tr>
<th>Variable</th>
<th>Error Variance</th>
<th>Total Variance</th>
<th>R-Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>11.52376</td>
<td>45.63609</td>
<td>0.7475</td>
</tr>
<tr>
<td>x2</td>
<td>9.14503</td>
<td>42.99597</td>
<td>0.7873</td>
</tr>
<tr>
<td>x3</td>
<td>6.68169</td>
<td>28.41532</td>
<td>0.7649</td>
</tr>
<tr>
<td>y1</td>
<td>0.78580</td>
<td>28.66835</td>
<td>0.9726</td>
</tr>
<tr>
<td>y2</td>
<td>2.88069</td>
<td>20.52319</td>
<td>0.8596</td>
</tr>
<tr>
<td>y3</td>
<td>5.15573</td>
<td>19.29032</td>
<td>0.7327</td>
</tr>
</tbody>
</table>

Alternative Identification Constraints

Setting the variances of the latent factors to 1 in the preceding FACTOR model specification makes the model identified. This is necessary because the scales of the latent factors are arbitrary and the constraints imposed on the factor variances fix the scales of the factors.

In practice, there is another way to fix the scales of the factors. For each factor, you can fix the loading of one of its measured indicators to a constant. This fixed loading value is usually set at 1. For example, you can represent the confirmatory factor model for the scores data by the following alternative path diagram:
This path diagram is essentially the same as the preceding one. However, the fixed constants adjacent to the double-headed arrows that attach to the two factors in the preceding path diagram are now moved to two of the single-headed paths in the current path diagram.

You can specify this path diagram by the following FACTOR model specification of PROC CALIS:

```plaintext
ods graphics on;
proc calis data=scores plots=pathdiagram;
  factor
    verbal ===> x1-x3 = 1. ,
    math ==> y1-y3 = 1. ;
run;
ods graphics off;
```

The PLOTS=PATHDIAGRAM option requests the path diagram output. In the FACTOR statement, you assign a fixed constant to each of the path entries. In the first entry, the constant 1 is assigned to the loading of x1 on the verbal factor, while all other loadings in this entry are (unnamed) free parameters. Similarly, in the second entry, the fixed constant 1 is assigned to the loading of y1 on the math factor, while all other loadings in this entry are (unnamed) free parameters. This completes the specification of the confirmatory factor model because all the double-headed arrows in the path diagram correspond to default free parameters in the FACTOR modeling language of PROC CALIS.

Output 29.12.7 shows some fit indices for the current confirmatory factor model for the scores data.

**Output 29.12.7** Fit Summary of the CFA Model with Alternative Identification Constraints: Scores Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>9.8052</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>8</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.2790</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0571</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0853</td>
</tr>
<tr>
<td>Bentler Comparative Fit Index</td>
<td>0.9887</td>
</tr>
</tbody>
</table>
The model fit chi-square is 9.805 \((df = 8, p = 0.279)\). This is the same model fit chi-square as that for the preceding CFA model specification with factor variances constrained to 1. In fact, all fit information in Output 29.12.7 are identical to Output 29.12.3.

Figure 29.12.8 shows the path diagram of the confirmatory factor model. The path diagram indicates significant estimates by attaching asterisks to the numerical values. Estimates that are flagged with one asterisk are significant at 0.05 \(\alpha\)-level. Estimates that are flagged with two asterisks are significant at 0.01 \(\alpha\)-level. The path diagram also shows a summary of fit statistics. For more information about specifying path diagram output, see the section “Path Diagrams: Layout Algorithms, Default Settings, and Customization” on page 1440.

**Output 29.12.8** Path Diagram and Fit Summary: Scores Data
Output 29.12.9 shows the parameter estimates under the current model specification. The loading of \( x_1 \) on the verbal factor is a fixed at 1, as required for the identification of the scale of the verbal factor. Similarly, the loading of \( y_1 \) on the math factor is a fixed at 1 for the identification of the scale of the math factor. All other loading estimates in Output 29.12.9 are not the same as those in the preceding model specification, as shown in Output 29.12.4. The reason is that the scales of the factors (as measured by the estimated standard deviations of the factors) in the two specifications are not the same. In the current model specification, the verbal factor has an estimated variance of 34.1123 and the math factor has an estimated variance of 27.8825, as shown in the second table of Output 29.12.9. Hence, the estimated standard deviations of these two factors are 5.8406 and 5.2804, respectively. But the standard deviations of the factors in the preceding confirmatory factor model specification are fixed at 1.

**Output 29.12.9** Loading and Factor Covariance Estimates of the CFA Model with Alternative Identification Constraints: Scores Data

<table>
<thead>
<tr>
<th>Factor Loading Matrix:</th>
<th>Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>verbal</td>
</tr>
<tr>
<td>( x_1 )</td>
<td>1.0000</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>0.9962</td>
</tr>
<tr>
<td></td>
<td>0.1576</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>0.7982</td>
</tr>
<tr>
<td></td>
<td>0.1286</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>( y_1 )</td>
<td>0</td>
</tr>
<tr>
<td>( y_2 )</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.0718</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>( y_3 )</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.0858</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
</tbody>
</table>


Example 29.12: Confirmatory Factor Models

**Output 29.12.9 continued**

<table>
<thead>
<tr>
<th>Factor Covariance Matrix:</th>
<th>Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>verbal</td>
</tr>
<tr>
<td>verbal</td>
<td>34.1123</td>
</tr>
<tr>
<td></td>
<td>11.6366</td>
</tr>
<tr>
<td></td>
<td>2.9315</td>
</tr>
<tr>
<td></td>
<td>0.003374</td>
</tr>
<tr>
<td></td>
<td>[Add1]</td>
</tr>
<tr>
<td>math</td>
<td>15.9585</td>
</tr>
<tr>
<td></td>
<td>6.7270</td>
</tr>
<tr>
<td></td>
<td>2.3723</td>
</tr>
<tr>
<td></td>
<td>0.0177</td>
</tr>
<tr>
<td></td>
<td>[Add3]</td>
</tr>
</tbody>
</table>

However, if you multiply the loading estimates in **Output 29.12.9** by the corresponding estimated factor standard deviation, you get the same set of loading estimates as in **Output 29.12.4**. For example, the loading of \(x_1\) on the **verbal** factor is 1.0 in **Output 29.12.9**. Multiplying this loading by the estimated standard deviation 5.8406 of the **verbal** factor gives you the same corresponding loading as in **Output 29.12.4**. Another example is the loading of \(y_3\) on the **math** factor. This loading is 0.7120 in **Output 29.12.9**. Multiplying this estimate by the estimated standard deviation 5.2804 of the **verbal** factor gives an estimate of 3.7596, which matches the corresponding loading estimate in **Output 29.12.4**. Therefore, the discrepancies in the loading estimates are due to different factor scales in the two specifications. The loading estimates in **Output 29.12.9** are simply rescaled version of the loading estimates in **Output 29.12.4**.

However, the scales of the factors do not affect the estimates of the error variances, as shown in **Output 29.12.10**. These estimates are the same as those for the preceding model specification, as shown in the **Output 29.12.5**.

**Output 29.12.10** Error Variance Estimates of the CFA Model with Alternative Identification Constraints: Scores Data

<table>
<thead>
<tr>
<th>Error Variances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>(x_1)</td>
</tr>
<tr>
<td>(x_2)</td>
</tr>
<tr>
<td>(x_3)</td>
</tr>
<tr>
<td>(y_1)</td>
</tr>
<tr>
<td>(y_2)</td>
</tr>
<tr>
<td>(y_3)</td>
</tr>
</tbody>
</table>

This example shows how you can fit a basic confirmatory factor model by the FACTOR modeling language of PROC CALIS. You can set the identification constraints and get statistically equivalent estimation results in two different ways. By setting up additional parameter constraints, you can also fit some variations of the basic confirmatory factor model. See Example 29.13 for illustrations of some restricted confirmatory factor models for the scores data.

When your data have missing values, with the default ML estimation method PROC CALIS deletes all observations with missing values for the analysis. This might result in a serious loss of information. Example 29.15 considers a hypothetical situation where some observations in the scores data have missing values in the observed variables. Only 16 observations have complete data. By using the full information
maximum likelihood (FIML) method for treating the missing data, Example 29.15 shows how you can fully use the information from the scores data set with missing values.

**Example 29.13: Confirmatory Factor Models: Some Variations**

This example shows how you can fit some variations of the basic confirmatory factor analysis model by the FACTOR modeling language. You apply these models to the scores data set that is described in Example 29.12. The data set contains six test scores of verbal and math abilities. Thirty-two students take the tests. Tests x1–x3 test their verbal skills and tests y1–y3 test their math skills.

**The Parallel Tests Model**

In classical measurement theory, test items for a latent factor are parallel if they have the same loadings on the factor and the same error variances (or reliability). Suppose for the scores data, the items within each of the verbal and the math factors are parallel. You can use the following path diagram to represent such a parallel tests model:

In the path diagram, the variances of the verbal and the math are both fixed at 1, as indicated by the constants 1.0 adjacent to the double-headed arrows that are attached to factors. You label all the single-headed paths in the path diagram by parameter names. For the three paths (loadings) from the verbal factor, you use the same parameter name load1. This means that these loadings are the same parameter. You also label the double-headed arrows that are attached to x1–x3 by the parameter name evar1. This means that the corresponding error variances for these three observed variables are exactly the same. Hence, x1–x3 are parallel tests for the verbal factor, as required by the current confirmatory factor model.

Similarly, you define parallel tests y1–y3 for the math factor by using load2 as the common factor loading parameter and evar2 as the common error variances for the observed variables.

Corresponding to this path diagram, you can specify the model by the following FACTOR model specification of PROC CALIS:
Example 29.13: Confirmatory Factor Models: Some Variations

```sas
proc calis data=scores;
   factor
      verbal ===> x1-x3 = load1 load1 load1,
      math  ===> y1-y3 = load2 load2 load2;
   pvar
      verbal = 1.,
      math  = 1.,
      x1-x3 = 3*evar1,
      y1-y3 = 3*evar2;
run;
```

In each entry of the FACTOR statement, you specify the factor-variables relationships, followed by a list of parameters. For example, the three loading parameters of \( x_1 - x_3 \) on the verbal factor are all named load1. This effectively constrains the corresponding loading estimates to be the same. Similarly, in the next entry you set equality constraints on the loading estimates \( y_1 - y_3 \) on the math factor by using the same parameter name load2.

To make the tests parallel, you also need to constrain the error variances for each variable cluster. In the PVAR statement, in addition to setting the factor variances to 1 for identification, you set all the error variances of \( x_1 - x_3 \) to be the same by using the same parameter name evar1. The notation 3*evar1 means that you want to specify evar1 three times, one time each for the error variances for the three observed variables in the variable list of the entry. Similarly, you set the equality of the error variances of \( y_1 - y_3 \) by using the same parameter name evar2.

Output 29.13.1 shows some fit indices of the parallel tests model for the scores data. The model fit chi-square is 26.128 (\( df = 16 \), \( p = 0.0522 \)). The SRMR value is 0.1537 and the RMSEA value is 0.1429. All these indices show that the model does not fit very well. However, Bentler’s CFI is 0.9366, which shows a good model fit.

```
Output 29.13.1 Model Fit of the Parallel Tests Model: Scores Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>26.1283</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>16</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.0522</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.1537</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.1429</td>
</tr>
<tr>
<td>Bentler Comparative Fit Index</td>
<td>0.9366</td>
</tr>
</tbody>
</table>
```

Output 29.13.2 shows the parameter estimates of the parallel tests model. The first table of Output 29.13.2 shows the required factor pattern for parallel tests. Variables \( x_1 - x_3 \) all have the same loading estimates on the verbal factor, and variables \( y_1 - y_3 \) all have the same loading estimates on the math factor. All loading estimates are statistically significant.
### Output 29.13.2 Parameter Estimates of the Parallel Tests Model: Scores Data

#### Factor Loading Matrix:

<table>
<thead>
<tr>
<th>Estimate/StdErr/t-value/p-value</th>
<th>verbal</th>
<th>math</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>5.4226</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.7655</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7.0833</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[load1]</td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td>5.4226</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.7655</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7.0833</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[load1]</td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>5.4226</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.7655</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7.0833</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[load1]</td>
<td></td>
</tr>
<tr>
<td>y1</td>
<td>0</td>
<td>4.4001</td>
</tr>
<tr>
<td></td>
<td>0.5926</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7.4246</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[load2]</td>
<td></td>
</tr>
<tr>
<td>y2</td>
<td>0</td>
<td>4.4001</td>
</tr>
<tr>
<td></td>
<td>0.5926</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7.4246</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[load2]</td>
<td></td>
</tr>
<tr>
<td>y3</td>
<td>0</td>
<td>4.4001</td>
</tr>
<tr>
<td></td>
<td>0.5926</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7.4246</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[load2]</td>
<td></td>
</tr>
</tbody>
</table>

#### Factor Covariance Matrix:

<table>
<thead>
<tr>
<th>Estimate/StdErr/t-value/p-value</th>
<th>verbal</th>
<th>math</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbal</td>
<td>1.0000</td>
<td>0.5024</td>
</tr>
<tr>
<td></td>
<td>0.1497</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.3569</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.000788</td>
<td>[Add1]</td>
</tr>
<tr>
<td>math</td>
<td>0.5024</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>0.1497</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.3569</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.000788</td>
<td>[Add1]</td>
</tr>
</tbody>
</table>

#### Error Variances

| Variable | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|----------|-----------|----------|----------------|---------|------|---|
| x1       | evar1     | 9.61122  | 1.72623        | 5.5678  | <.0001|   |
| x2       | evar1     | 9.61122  | 1.72623        | 5.5678  | <.0001|   |
| x3       | evar1     | 9.61122  | 1.72623        | 5.5678  | <.0001|   |
| y1       | evar2     | 3.46673  | 0.62264        | 5.5678  | <.0001|   |
| y2       | evar2     | 3.46673  | 0.62264        | 5.5678  | <.0001|   |
| y3       | evar2     | 3.46673  | 0.62264        | 5.5678  | <.0001|   |
In the second table of Output 29.13.2, the factor covariance (or correlation) estimate is 0.5024, showing moderate relationship between the verbal and the math factors. The last table of Output 29.13.2 shows the error variances of the variables. As required by the parallel tests model, the error variance estimates of x1–x3 are all 9.6112, and the error variance estimates of y1–y3 are all 3.4667.

The Tau-Equivalent Tests Model

Because the parallel tests model does not fit well, you are looking for a less constrained model for the scores data. The tau-equivalent tests model is such a model. It requires only the equality of factor loadings but not the equality of error variances within each factor. The following path diagram represents the tau-equivalent tests model for the scores data:

```
  x1  x2  x3  y1  y2  y3
  |    |    |    |    |    |
  ↓    ↓    ↓    ↓    ↓    ↓
load1 load1 load1 load2 load2 load2

  verbal      math
  1.0         1.0
```

This path diagram is much the same as that for the parallel tests model except that now you do not use parameter names to label the double-headed arrows that are attached to the observed variables. This means that you allow the corresponding error variances to be free parameters in the tau-equivalent tests model. You can use the following FACTOR model specification of PROC CALIS to specify the tau-equivalent tests model for the scores data:

```
proc calis data=scores;
  factor
    verbal ===> x1-x3 = load1 load1 load1,
    math   ===> y1-y3 = load2 load2 load2;
  pvar
    verbal = 1.,
    math   = 1.;
run;
```

This specification is the same as that for the parallel tests model except that you remove the specifications about the error variances in the PVAR statement in the current tau-equivalent model. This effectively allows the error variances of the observed variables to be (default) free parameters in the model.

Output 29.13.3 shows some model fit indices of the tau-equivalent tests model for the scores data. The chi-square is 22.0468 (df = 12, p = 0.037). The SRMR is 0.1398 and the RMSEA is 0.1643. The comparative fit index (CFI) is 0.9371. Except for the CFI value, all other values do not support a good model fit. This model has a degrees of freedom of 12, which is less restrictive (has more parameters) than the parallel tests model, which has a degrees of freedom of 16, as shown in Output 29.13.1. However, it seems that the tau-equivalent tests model is still too restrictive for the data.
Output 29.13.3 Model Fit of the Tau-Equivalent Tests Model: Scores Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
</tr>
<tr>
<td>Bentler Comparative Fit Index</td>
</tr>
</tbody>
</table>

Output 29.13.4 shows the parameter estimates. The first table of Output 29.13.4 shows the required pattern of factor loadings under the tau-equivalent tests model. The third table of Output 29.13.4 shows the error variance estimates. The error variance parameters are no longer constrained under the tau-equivalent tests model. Each has a unique estimate.

Output 29.13.4 Parameter Estimates of the Tau-Equivalent Tests Model: Scores Data

<table>
<thead>
<tr>
<th>Factor Loading Matrix: Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbal</td>
</tr>
<tr>
<td>x1           5.2418</td>
</tr>
<tr>
<td>0.7374</td>
</tr>
<tr>
<td>7.1085</td>
</tr>
<tr>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[load1]</td>
</tr>
<tr>
<td>x2           5.2418</td>
</tr>
<tr>
<td>0.7374</td>
</tr>
<tr>
<td>7.1085</td>
</tr>
<tr>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[load1]</td>
</tr>
<tr>
<td>x3           5.2418</td>
</tr>
<tr>
<td>0.7374</td>
</tr>
<tr>
<td>7.1085</td>
</tr>
<tr>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[load1]</td>
</tr>
<tr>
<td>y1           0</td>
</tr>
<tr>
<td>0.5932</td>
</tr>
<tr>
<td>7.4953</td>
</tr>
<tr>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[load2]</td>
</tr>
<tr>
<td>y2           0</td>
</tr>
<tr>
<td>0.5932</td>
</tr>
<tr>
<td>7.4953</td>
</tr>
<tr>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[load2]</td>
</tr>
<tr>
<td>y3           0</td>
</tr>
<tr>
<td>0.5932</td>
</tr>
<tr>
<td>7.4953</td>
</tr>
<tr>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[load2]</td>
</tr>
</tbody>
</table>
The Partially Constrained Parallel Tests Model

Because both the parallel tests and tau-equivalent tests models do not fit the data well, you can explore an alternative model for the scores data. Suppose that for each factor only two (but not all) of their measured variables (tests) are parallel. For example, suppose you know that tests $x_1$ and $x_2$ are very similar to each other (for example, both are speeded tests with forced-choice answers), while $x_3$ is a little different in the way it is administered (for example, open-ended questions). Although all tests are designed for measuring the verbal factor, only $x_1$ and $x_2$ are parallel tests while $x_3$ is congeneric to the verbal factor. Similarly, suppose you can argue that $y_2$ and $y_3$ are parallel tests while $y_1$ is only congeneric to the math factor.

The current modeling idea is represented by the following path diagram:
In the path diagram, x1 and x2 have the same parameter load1 for the paths from the verbal factor. Their error variances are also the same, as labeled with the evar1 parameter adjacent to the double-headed arrows that are attached to the variables. The test x3 has distinct parameter names for its associated path and the attached double-headed arrow. The corresponding loading and error variance parameters are alpha and phi, respectively. Similarly, with the use of specific parameter names, you define y2 and y3 as parallel tests for the math factor, while y1 is congeneric to the same factor but with distinct loading and error variance parameters. Lastly, you fix the variances of the factors to 1.0 for identification of the factor scales.

You can specify such a partially constrained parallel tests model by the following FACTOR model specification of PROC CALIS:

```plaintext
proc calis data=scores;
  factor
    verbal ===> x1-x3 = load1 load1 alpha,
    math ===> y1-y3 = beta load2 load2;
  pvar
    verbal = 1.,
    math = 1.,
    x1-x3 = evar1 evar1 phi,
    y1-y3 = theta evar2 evar2;
run;
```

First, in the FACTOR statement, you name the loading parameters that reflect the parallel tests constraints. For example, the loading parameters of x1 and x2 on the verbal factor are both named load1. This means that they are the same. However, the loading parameter of x3 on the verbal factor is named alpha, which means that it is a separate parameter. Similarly, you apply the load2 parameter name to the loading parameters of y2 and y3 on the math factor, but the loading parameter of y1 on the math factor is a distinct parameter named beta.

In the PVAR statement, the two factor variances are set to a constant 1 for the identification of latent factor scales. Next, you use the same naming techniques as in the FACTOR statement to constrain some parts of the error variances. As a result, together with the specifications in the FACTOR statement, x1 and x2 are parallel tests for the verbal factor and y2 and y3 are parallel tests for the math factor, while x3 and y1 are only congeneric tests for their respective factors.

Output 29.13.5 shows some fit indices of the partially constrained parallel tests model. The model fit chi-square is 12.6784 (df = 12, p = 0.3928). The SRMR is 0.0585 and the RMSEA is close to 0.0427. The comparative fit index (CFI) is 0.9958. All these fit indices point to a quite reasonable model fit for the scores data.

**Output 29.13.5 Model Fit of the Partially Constrained Parallel Tests Model: Scores Data**

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>12.6784</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>12</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.3928</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0585</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0427</td>
</tr>
<tr>
<td>Bentler Comparative Fit Index</td>
<td>0.9958</td>
</tr>
</tbody>
</table>

Notice that the current model actually has the same degrees of freedom as that of the tau-equivalent tests model, as shown in Output 29.13.3. Both models have nine parameters. But the current partially constrained
Example 29.13: Confirmatory Factor Models: Some Variations

Parallel tests model is definitely a better model for the data. This shows that sometimes you do not have to add more parameters to improve the model fit. Structurally different models might explain the data quite differently, even though they might use the same number of parameters.

Output 29.13.6 show the parameter estimates of the partially constrained parallel tests model for the scores data. The estimates in the factor loading matrix and error variances table confirm the prescribed nature of the tests—that is, x1 and x2 are parallel tests for the verbal factor and y2 and y3 are parallel tests for the math factor.

Output 29.13.6 Parameter Estimates of the Partially Constrained Parallel Tests Model: Scores Data

<table>
<thead>
<tr>
<th>Factor Loading Matrix:</th>
<th>Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>verbal</td>
</tr>
<tr>
<td>x1</td>
<td>5.8306 0.8593 6.7853 &lt;.0001 [load1]</td>
</tr>
<tr>
<td>x2</td>
<td>5.8306 0.8593 6.7853 &lt;.0001 [load1]</td>
</tr>
<tr>
<td>x3</td>
<td>4.6623 0.7814 5.9664 &lt;.0001 [alpha]</td>
</tr>
<tr>
<td>y1</td>
<td>0 5.2784 0.7010 7.5294 &lt;.0001 [beta]</td>
</tr>
<tr>
<td>y2</td>
<td>0 3.9789 0.5732 6.9419 &lt;.0001 [load2]</td>
</tr>
<tr>
<td>y3</td>
<td>0 3.9789 0.5732 6.9419 &lt;.0001 [load2]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Factor Covariance Matrix:</th>
<th>Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>verbal</td>
</tr>
<tr>
<td>verbal</td>
<td>1.0000 0.5203 0.1425 3.6497 0.000263 [Add1]</td>
</tr>
<tr>
<td>math</td>
<td>0.5203 0.1425 3.6497 0.000263 [Add1]</td>
</tr>
</tbody>
</table>

Example 29.14: Residual Diagnostics and Robust Estimation

This example illustrates case-level residual diagnostics and robust estimation with PROC CALIS. The data set is available in Mardia, Kent, and Bibby (1979). It contains 88 responses on five subjects: mechanics, vector, algebra, analysis, and statistics. Yuan and Hayashi (2010) use the same data set to illustrate the uses of residual diagnostics in the context of robust estimation. Inspired by their illustrations, this example mirrors their one-factor model, with some modifications.

Case-Level Residual Diagnostics

The following DATA step specifies and inputs five observed variables for the data set mardia:

```sas
data mardia;
   input mechanics vector algebra analysis statistics;
datalines;
  77.000 82.000 67.000 67.000 81.000
  63.000 78.000 80.000 70.000 81.000
  75.000 73.000 71.000 66.000 81.000
  .  .  .  .  .
  .  .  .  .  .
  { More Data }
  .  .  .  .  .
  .  .  .  .  .
  .  .  .  .  .
  .  .  .  .  .
;
```

You specify a one-factor model for the five variables by using the following statements:

```sas
ods graphics on;
proc calis data=mardia residual plots=caseresid;
   path
       fact1 ===> mechanics vector algebra analysis statistics = 1.;
run;
ods graphics off;
```
Example 29.14: Residual Diagnostics and Robust Estimation

In the PATH statement, all observed variables are indicators of the latent factor fact1. The first factor loading for mechanics is fixed at 1 for identification of the latent variable scale, while the remaining four loadings are free parameters in the model. The RESIDUAL option that is specified in the PROC CALIS statement requests residual analysis. Because raw data are input, case-level residual analysis is done. To request all the available ODS Graphics output for case-level residual diagnostics, you specify the PLOTS=CASERESID option. To enable ODS Graphics, you specify the ODS GRAPHICS ON statement before running the CALIS procedure.

Output 29.14.1 shows some of the fit statistics of the one-factor model. The model fit chi-square is 8.9782 ($df = 5, p = 0.1099$). Therefore, you fail to reject the one-factor model for the data. The SRMR is 0.0411, which shows a good fit. But the RMSEA is 0.0956, which does not show that the one-factor model is a very good approximating model for the data. Output 29.14.2 shows the loading estimates. All are statistically significant.

### Output 29.14.1 Model Fit of Mardia Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
</tr>
</tbody>
</table>

### Output 29.14.2 Estimates of Loadings

<table>
<thead>
<tr>
<th>PATH List</th>
<th>Path</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fact1</td>
<td>mechanics</td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fact1</td>
<td>vector</td>
<td>_Parm1</td>
<td>0.83770</td>
<td>0.16598</td>
<td>5.0469</td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fact1</td>
<td>algebra</td>
<td>_Parm2</td>
<td>0.93064</td>
<td>0.15369</td>
<td>6.0552</td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fact1</td>
<td>analysis</td>
<td>_Parm3</td>
<td>1.09480</td>
<td>0.19611</td>
<td>5.5826</td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fact1</td>
<td>statistics</td>
<td>_Parm4</td>
<td>1.19229</td>
<td>0.22309</td>
<td>5.4444</td>
<td>&lt;.0001</td>
<td></td>
</tr>
</tbody>
</table>

Output 29.14.3 shows the classifications of outlier and leverage observations in a two-dimensional space. Four regions are clearly shown.
The upper and the lower regions are separated by a horizontal line that represents the criterion for outlier detection. This criterion is set by a certain \( \alpha \)-level that has a probability meaning. Loosely speaking, if the model is true and the distribution of the residuals is multivariate normal, you would expect that about \( \alpha \times 100\% \) of observations will have their residual M-distances above the criterion. The larger the \( \alpha \) value, the more liberal the outlier detection criterion. This \( \alpha \)-level is set to 0.01 by default. You can change this default by using the \texttt{ALPHAOUT=} option.

Notice that all residuals and the criterion are measured in terms of M-distances. Hence, these measures are always positive. \textbf{Output 29.14.3} classifies observations 81 and 28 as outliers (in the residual dimension). In addition to these two outlying observations, PROC CALIS also labels the next five observations with the largest residual M-distances. \textbf{Output 29.14.3} shows that observations 33, 54, 56, 61, and 66 are the closest to being labeled as outliers.

The right and left regions are separated by a vertical line that represents the criterion for leverage observations. Again, this criterion is set by a certain \( \alpha \)-level that has a probability meaning. If the model is true and the distribution of the predictor variables (in this case, the factor variable \texttt{fact1}) is multivariate normal, you would expect that about \( \alpha \times 100\% \) of observations will have their leverage M-distances above the criterion. The larger the \( \alpha \) value, the more liberal the leverage observation detection criterion. This \( \alpha \)-level is also set to 0.01 by default. You can change this default by using the \texttt{ALPHALEV=} option.
Output 29.14.3 classifies observations 2, 87, and 88 as leverage points or observations. In addition to these three observations, PROC CALIS also labels the next five observations with the largest leverage M-distances. Output 29.14.3 shows that observations 1, 3, 6, 8, and 86 are the closest to being labeled as leverage observations.

The lower left region in Output 29.14.3 contains observations that are neither outliers nor leverage observations. This region contains the majority of the observations. The upper right region in Output 29.14.3 contains observations that are classified as both outliers and leverage observations. For the current model, no observations are classified as such.

To supplement the graphical depiction of outliers and leverage observations, PROC CALIS shows two tables that contain more numerical information. Output 29.14.4 shows the seven observations that have the largest residual M-distances. The table reflects the outlier classification in Output 29.14.3 numerically. Two outliers are flagged in the table. However, none of the seven observations are leverage observations. Output 29.14.5 shows the eight observations that have the largest leverage M-distances. Again, this table reflects the classification of leverage observations in Output 29.14.3 numerically. Three leverage observations are flagged, but none of these observations are residual outliers at the same time.

**Output 29.14.4  Largest Residual Outliers**

<table>
<thead>
<tr>
<th>Case Number</th>
<th>Residual (M-Distance)</th>
<th>Outlier</th>
<th>Leverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>4.60511 *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>3.69953 *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>3.43068</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>3.17659</td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>3.15104</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>3.06094</td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>3.03262</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Output 29.14.5  Largest Leverage Observations**

<table>
<thead>
<tr>
<th>Case Number</th>
<th>Leverage (M-Distance)</th>
<th>Leverage</th>
<th>Outlier</th>
</tr>
</thead>
<tbody>
<tr>
<td>87</td>
<td>2.96438 *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>2.72441 *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2.69293 *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2.15519</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2.04014</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1.87894</td>
<td></td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>1.72950</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1.59242</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Output 29.14.6 shows the histogram and its kernel density function of the residual M-distances. Theoretically, the density should look like a $\chi$-distribution (that is, the distribution of the square root of the corresponding $\chi^2$ variate). While this histogram provides an overall impression of the empirical distribution of the residual M-distances, it is not very useful in diagnosing the observations that do not conform to the theoretical distribution of the residual M-distances. To this end, Output 29.14.7 and Output 29.14.8 are more useful.

**Output 29.14.6** Distribution of Case-Level Residuals

![Distribution of Case-Level Residuals (M-distances)](image)

Output 29.14.7 shows the so-called Q-Q plot for the residual M-distances. The Y coordinates represent the observed quantiles, and the X coordinates represent the theoretical quantiles. Output 29.14.8 shows the so-called P-P plots for the residual M-distances. The Y coordinates represent the observed percentiles, and the X coordinates represent the theoretical percentiles. If the observed residual M-distances distribute exactly like the theoretical $\chi$-distribution, all observations should lie on the straight lines with a slope of 1 in both plots. In Output 29.14.7 and Output 29.14.8, observations with the largest departures from the straight lines are labeled.
In terms of quantiles, the Q-Q plot in **Output 29.14.7** shows that observations 81, 28, and 54 have the largest departures from the theoretical distribution for the residual M-distances. Notice that observations 81 and 28 were also identified as outliers previously.
In terms of percentiles, the P-P plot in Output 29.14.8 shows that several observations in the middle of the distribution have the largest departures from the theoretical distribution. However, it is not clear from the plot which observations have the largest departures.

To supplement the Q-Q and the P-P plots with numerical information, PROC CALIS outputs the table in Output 29.14.9, which shows the observations with the largest departures from the theoretical residual M-distance distribution in terms of quantiles and percentiles. The observations in the table are arranged by their percentile regions. Approximately 10% of the observations with the largest departures in terms of quantile and in terms of percentile, respectively, are shown in this table. However, the final percentage of observations shown in the table might not add up to 20%, because some observations could have large departures in terms of both quantile and percentile. Output 29.14.9 shows that observations 1, 15, 30, and 72 are this kind of observation. However, none of these are residual outliers.
Output 29.14.9  Largest Departures from the Theoretical Residual Distribution  

<table>
<thead>
<tr>
<th>Percentile Region</th>
<th>Case Number</th>
<th>Quantile Deviation</th>
<th>Percentile Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(.65,.90)</td>
<td>26</td>
<td>-0.05302</td>
<td></td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>-0.05836</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-0.05863</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>-0.13087</td>
<td>-0.06509</td>
</tr>
<tr>
<td></td>
<td>87</td>
<td>-0.05257</td>
<td></td>
</tr>
<tr>
<td></td>
<td>71</td>
<td>-0.05145</td>
<td></td>
</tr>
<tr>
<td>(.90,.95)</td>
<td>30</td>
<td>-0.11973</td>
<td>-0.05198</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-0.12344</td>
<td>-0.05065</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>-0.13739</td>
<td>-0.05498</td>
</tr>
<tr>
<td>&gt;.95</td>
<td>64</td>
<td>-0.12550</td>
<td></td>
</tr>
<tr>
<td></td>
<td>66</td>
<td>0.13803</td>
<td></td>
</tr>
<tr>
<td></td>
<td>54</td>
<td>0.20039</td>
<td></td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>0.32398</td>
<td></td>
</tr>
<tr>
<td></td>
<td>81</td>
<td>0.99843</td>
<td></td>
</tr>
</tbody>
</table>

Output 29.14.10 shows the residual on fit plot for the variable algebra. Notice that unlike the preceding residual diagnostic plots where the M-distances (nonnegative) for residuals are plotted, the residuals (which can be positive or negative) themselves are plotted against the predicted values for algebra in the current residual on fit (or residual on predicted) plot. If the linear relationship as described by the one-factor model is reasonable, then the residuals should show no systematic trend with the predicted values. That is, the residuals should be randomly scattered around the plot in the range of the predicted values. The pattern of residuals in Output 29.14.10 looks sufficiently “random,” and therefore it gives support to the prescribed linear relationship between the variable algebra and the factor fact1. PROC CALIS also outputs the residual on fit plots for other dependent observed variables. To conserve space, they are not shown here. You can also control the amount of graphical output by specifying the desirable set of dependent observed variables for the residual on fit plots. For example, the following requests the residual on fit plots for the variables mechanics and analysis only, even though there are five dependent observed variables in the model:

```
PLOTS=RESBYPRED(VAR=mechanics analysis)
```

See the PLOTS= option for details.
Direct Robust Estimation

Usually, the outlier detection process in residual diagnostics implies some treatments after their detections. For example, in the preceding analysis, observations 28 and 81 were detected as outliers. You would naturally think that the next step is to do an additional model fitting with these two outlying observations removed. While this thinking is intuitively appealing, the outlier issues are usually more complicated than can be resolved by such a “pick-and-remove” strategy. One issue is the so-called masking effect of outliers. In the preceding analysis, outliers are detected based on a criterion that is a function of the chosen criterion level (for example, the ALPHAOOUT= value set by the researcher) and the parameter estimates. However, because the outliers have contributed to the estimation of the parameters, you might wonder whether the outlier detection process has been contaminated by the outliers themselves. If such a masking effect is present and an additional analysis is done after employing the pick-and-remove strategy for the outliers, you would be likely to discover more outliers. As a result, you might wonder whether the outlier removal process should be carried out indefinitely.

Robust estimation provides an alternative way to resolve the issue of masking effect. In robust estimation, outlying observations are downweighted simultaneously with the estimation. Iterative steps are used to reweight the observations according to the updated parameter estimates. Hence, with robust estimation you
Example 29.14: Residual Diagnostics and Robust Estimation

do not need to make a decision about removing outliers because they are already downweighted during the estimation. The issue of masking effect is avoided.

PROC CALIS provides two major methods of robust estimation—direct and two-stage. This section illustrates the direct robust estimation method, in which the model is estimated simultaneously with the (re)weighting of the observations. The next section illustrates the two-stage estimation method, in which the robust mean and covariance matrices obtained in the first stage are analyzed by the maximum likelihood (ML) estimation in the second stage. Both methods are based on the methodology proposed by Yuan and Zhong (2008) and Yuan and Hayashi (2010).

The following statements request direct robust estimation of the model for the mardia data set:

```plaintext
ods graphics on;
proc calis data=mardia residual robust plots=caseresid pcorr;
    path
        fact1 ===> mechanics vector algebra analysis statistics = 1. ;
run;
ods graphics off;
```

The ROBUST option invokes the direct robust estimation method. The direct robust method reweights the observations according to the magnitudes of the residuals during iterative steps. For this reason, you can also specify ROBUST=RES to signify direct robust estimation with residual (M-distance) weighting. Following Yuan and Hayashi (2010), there are two variants of the direct robust estimation. One treats the disturbances (that is, error terms for endogenous latent factors) as factors to estimate (see the ROBUST=RES(F) option), and the other treats the disturbances as true error terms (see the ROBUST=RES(E) option). See the ROBUST= option and the section “Robust Estimation” on page 1470 for details. These variants would not make any difference for the current one-factor model estimation, and the ROBUST option that is used in the specification is the same as ROBUST=RES(E).

You also request case-level residual analysis by using the RESIDUAL option, even though the intention now is to identify rather than to remove the outliers.

Output 29.14.11 shows the fit summary of the direct robust estimation. The model fit chi-square is 6.0031 ($df = 5, p = 0.3059$). Again, you fail to reject the one-factor model for the data. Unlike the regular ML estimation in the preceding analysis, both SRMR (0.0321) and RMSEA (0.0480) show good model fit. This result can be attributed to the downweighting of the outlying observations in the direct robust estimation. Output 29.14.12 shows that all loading estimates are statistically significant.

**Output 29.14.11** Model Fit of Mardia Data with Direct Robust Estimation

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>6.0031</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>5</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.3059</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0370</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0480</td>
</tr>
</tbody>
</table>

**NOTE:** Saturated mean structure parameters are excluded from the computations of fit indices.
Output 29.14.12  Estimates of Loadings with Direct Robust Estimation

| Path     | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|----------|-----------|----------|----------------|---------|-------|---|
| fact1 = mechanics |            | 1.00000  |                |         |       |   |
| fact1 = vector | _Parm1     | 0.83279  | 0.15680        | 5.3111  | <.0001|   |
| fact1 = algebra | _Parm2    | 0.92854  | 0.14763        | 6.2895  | <.0001|   |
| fact1 = analysis | _Parm3   | 1.08665  | 0.18855        | 5.7632  | <.0001|   |
| fact1 = statistics | _Parm4  | 1.18799  | 0.21487        | 5.5289  | <.0001|   |

Output 29.14.12 shows the outlier and leverage classifications by the direct robust estimation. This plot has a similar pattern to the one obtained from the preceding regular ML estimation (see Output 29.14.3). Both plots show that observations 28 and 81 are the only outliers. Thus, the results from the direct robust estimation eliminate the concern of the masking effect.

Output 29.14.13  Outliers and Leverage Observations with Direct Robust Estimation

As a by-product of the direct robust estimation, robust (unstructured) covariance and mean matrices are computed with the PCORR option. Output 29.14.14 shows the robust covariance and mean matrices obtained for the current direct robust estimation for the mardia data set. These matrices are essentially computed by
the usual formulas for covariance and mean matrices, but with weights obtained from the final estimates of the direct robust estimation of the model. For the current analysis, outlying observations (28 and 81) receive weights of less than 1 whereas all other observations receive weights of 1 in the computation.

**Output 29.14.14** Robust Covariance and Mean Matrices with Direct Robust Estimation

<table>
<thead>
<tr>
<th>Robust Covariance Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>mechanics</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>mechanics</td>
</tr>
<tr>
<td>vector</td>
</tr>
<tr>
<td>algebra</td>
</tr>
<tr>
<td>analysis</td>
</tr>
<tr>
<td>statistics</td>
</tr>
</tbody>
</table>

**Determinant** 33723890038 **Ln** 24.241472

<table>
<thead>
<tr>
<th>Robust Means</th>
</tr>
</thead>
<tbody>
<tr>
<td>mechanics</td>
</tr>
<tr>
<td>39.11015</td>
</tr>
</tbody>
</table>

**Two-Stage Robust Estimation**

In addition to the direct robust estimation method, PROC CALIS implements another robust estimation method for structural equation models. The two-stage robust estimation method estimates the robust means and covariance matrices in the first stage. It then fits the model to the robust mean and covariance matrix by using the ML method in the second stage. During the first stage, the robust method downweights the outlying observations in all variable dimensions. The robust properties are then propagated to the second stage of estimation.

In PROC CALIS, you can use the ROBUST=SAT option to invoke the two-stage robust estimation method, as shown in the following statements for fitting the one-factor model to the *mardia* data set:

```plaintext
proc calis data=mardia robust=sat pcorr;
    path
        fact1 ===> mechanics vector algebra analysis statistics = 1. ;
run;
```

With the PCORR option, the (unstructured) robust mean and covariance matrices are displayed in Output 29.14.15. Although the robust covariance and mean matrices obtained here are numerically similar to those obtained from the direct robust estimation (see Output 29.14.14), they are obtained with totally different methods. The robust mean and covariance matrices in the direct robust estimation are the by-products of the model estimation, while the robust mean and covariance matrices in the two-stage robust estimation are the initial inputs for model estimation.
Output 29.14.15 Robust Covariance and Mean Matrices with Two-Stage Robust Estimation

<table>
<thead>
<tr>
<th>Robust Covariance Matrix</th>
<th>mechanics</th>
<th>vector</th>
<th>algebra</th>
<th>analysis</th>
<th>statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>mechanics</td>
<td>298.83513</td>
<td>119.28292</td>
<td>99.65134</td>
<td>105.47098</td>
<td>116.68242</td>
</tr>
<tr>
<td>vector</td>
<td>119.28292</td>
<td>165.11323</td>
<td>84.25310</td>
<td>94.28855</td>
<td>98.70260</td>
</tr>
<tr>
<td>algebra</td>
<td>99.65134</td>
<td>84.25310</td>
<td>110.93012</td>
<td>110.46759</td>
<td>121.18480</td>
</tr>
<tr>
<td>analysis</td>
<td>105.47098</td>
<td>94.28855</td>
<td>110.46759</td>
<td>219.56424</td>
<td>155.24377</td>
</tr>
<tr>
<td>statistics</td>
<td>116.68242</td>
<td>98.70260</td>
<td>121.18480</td>
<td>155.24377</td>
<td>298.80146</td>
</tr>
</tbody>
</table>

Determinant 36001000868 Ln 24.306813

Output 29.14.16 shows the fit summary of the two-stage robust estimation. The model fit chi-square is 7.2941 ($df = 5, p = 0.1997$). As in direct robust estimation, you fail to reject the one-factor model for the data. Unlike the case with direct robust estimation, only SRMR (0.0366) shows a good model fit. RMSEA is 0.0726 and is not very close to a good model fit.

Output 29.14.16 Model Fit of Mardia Data with Two-Stage Robust Estimation

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>7.2941</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>5</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.1997</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0422</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0726</td>
</tr>
</tbody>
</table>

NOTE: Saturated mean structure parameters are excluded from the computations of fit indices.

Output 29.14.16 shows the estimates of loadings with the two-stage robust estimation. All the estimates are statistically significant. All the numerical results are similar to that of the direct robust estimation in Output 29.14.12.
Output 29.14.17  Estimates of Loadings with Two-Stage Robust Estimation

<table>
<thead>
<tr>
<th>Path</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fact1 ===&gt; mechanics</td>
<td></td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fact1 ===&gt; vector</td>
<td>_Parm1</td>
<td>0.84322</td>
<td>0.16538</td>
<td>5.0986</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fact1 ===&gt; algebra</td>
<td>_Parm2</td>
<td>0.93683</td>
<td>0.15485</td>
<td>6.0500</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fact1 ===&gt; analysis</td>
<td>_Parm3</td>
<td>1.10397</td>
<td>0.19849</td>
<td>5.5617</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fact1 ===&gt; statistics</td>
<td>_Parm4</td>
<td>1.21080</td>
<td>0.22686</td>
<td>5.3371</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The outlier and leverage observation detection in two-stage robust estimation is essentially the same method as that in direct robust estimation. Therefore, the corresponding results for the two-stage robust estimation are not shown here. See the section “Residual Diagnostics with Robust Estimation” on page 1497 for details about case-level residual analysis with robust estimation.

Example 29.15: The Full Information Maximum Likelihood Method

This example shows how you can fully utilize all available information from the data when there is a high proportion of observations with random missing value. You use the full information maximum likelihood method for model estimation.

In Example 29.12, 32 students take six tests. These six tests are indicator measures of two ability factors: verbal and math. You conduct a confirmatory factor analysis in Example 29.12 based on a data set without any missing values. The path diagram for the confirmatory factor model is shown the following:
Suppose now due to sickness or unexpected events, some students cannot take part in one of these tests. Now, the data test contains missing values at various locations, as indicated by the following DATA step:

```plaintext
data missing;
  input x1 x2 x3 y1 y2 y3;
  datalines;
23   .  16  15  14  16
29   26  23  22  18  19
14   21   .  15  16  18
20   18  17  18  21  19
25   26  22   .  21  26
26   19  15  16  17  17
   .  17  19   4  6  7
12   17  18  14  16   .
25   19  22  22  20  20
  7  12  15  10  11  8
29   24   .  14  13  16
28   24  29  19  19  21
12   9  10  18  19   .
11   .  12  15  16  16
20   14  15  24  23  16
26   25   .  24  23  24
20   16  19  22  21  20
14   .  15  17  19  23
14  20  13  24   .  .
29   24  24  21  20  18
26   .  26  28  26  23
20   23  24  22  23  22
23   24  20  23  22  18
14   .  17   .  16  14
28   34  27  25  21  21
17  12  10  14  12  16
   .  1  13  14  15  14
22  19  19  13  11  14
18  21   .  15  18  19
12  12  10  13  13  16
22  14  20  20  18  19
29  21  22  13  17  .
;
```

This data set is similar to the scores data set used in Example 29.12, except that some values are replaced at random with missing values. You can still fit the same confirmatory factor analysis model described in Example 29.12 to this data set by the default maximum likelihood (ML) method, as shown in the following statement:
The data set, the number of observations, the model type, and analysis type are shown in the first table of Output 29.15.1. Although PROC CALIS reads all 32 records in the data set, only 16 of these records are used. The remaining 16 records contain at least one missing value in the tests. They are discarded from the analysis. Therefore, the maximum likelihood method only uses those 16 observations without missing values.

Output 29.15.1  Modeling Information of the CFA Model: Missing Data

Confirmatory Factor Model With \Dataset(Missing) Data: ML
FACTOR Model Specification

The CALIS Procedure
Covariance Structure Analysis: Model and Initial Values

<table>
<thead>
<tr>
<th>Modeling Information</th>
<th>Maximum Likelihood Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set</td>
<td>WORK.MISSING</td>
</tr>
<tr>
<td>N Records Read</td>
<td>32</td>
</tr>
<tr>
<td>N Records Used</td>
<td>16</td>
</tr>
<tr>
<td>N Obs</td>
<td>16</td>
</tr>
<tr>
<td>Model Type</td>
<td>FACTOR</td>
</tr>
<tr>
<td>Analysis</td>
<td>Covariances</td>
</tr>
</tbody>
</table>
Output 29.15.2 shows the parameter estimates.

**Output 29.15.2** Parameter Estimates of the CFA Model: Missing Data

<table>
<thead>
<tr>
<th>Factor Loading Matrix:</th>
<th>Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>verbal</td>
</tr>
<tr>
<td>x1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.1110</td>
</tr>
<tr>
<td></td>
<td>1.3110</td>
</tr>
<tr>
<td></td>
<td>3.8984</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[Parm1]</td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.6261</td>
</tr>
<tr>
<td></td>
<td>1.2561</td>
</tr>
<tr>
<td></td>
<td>4.4790</td>
</tr>
<tr>
<td>[Parm2]</td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.8739</td>
</tr>
<tr>
<td></td>
<td>1.1410</td>
</tr>
<tr>
<td></td>
<td>4.2717</td>
</tr>
<tr>
<td>[Parm3]</td>
<td></td>
</tr>
<tr>
<td>y1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>[Parm4]</td>
<td></td>
</tr>
<tr>
<td>y2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>[Parm5]</td>
<td></td>
</tr>
<tr>
<td>y3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>[Parm6]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Factor Covariance Matrix:</th>
<th>Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>verbal</td>
</tr>
<tr>
<td>verbal</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>0.1464</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[Add1]</td>
<td></td>
</tr>
<tr>
<td>math</td>
<td>0.7050</td>
</tr>
<tr>
<td></td>
<td>0.1464</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[Add1]</td>
<td></td>
</tr>
</tbody>
</table>
Most of the factor loading estimates shown in Output 29.15.2 are similar to those estimated from the data set without missing values, as shown in Output 29.12.4. The loading estimate of $y_3$ on the math factor shows the largest discrepancy. With only half of the data used in the current estimation, this loading estimate is 2.6338 in the current analysis, while it is 3.7596 if no data were missing, as shown in Output 29.12.4. Another obvious difference between the two sets of results is that the standard error estimates for the loadings are consistently larger in the current analysis than in the analysis in Example 29.12 where there are no missing data. This is expected because you have only half of the data set available in the current analysis.

Similarly, the estimates for the factor covariance and error variances are mostly similar to those in the analysis with complete data, but the standard error estimates in the current analysis are consistently higher.

The maximum likelihood method, as implemented in PROC CALIS, deletes all observations with at least one missing value in the estimation. In a sense, the partially available information of these deleted observations is wasted. This greatly reduces the efficiency of the estimation, which results in higher standard error estimates.

To fully utilize all available information from the data set with the presence of missing values, you can use the full information maximum likelihood (FIML) method in PROC CALIS, as shown in the following statements:

```plaintext
proc calis method=fiml data=missing;
  factor
    verbal ===> x1-x3,
    math  ===> y1-y3;
  pvar
    verbal = 1.,
    math   = 1.;
run;
```

In the PROC CALIS statement, you use METHOD=FIML to request the full information maximum likelihood method. Instead of deleting observations with missing values, the full information maximum likelihood method uses all available information in all observations. Output 29.15.3 shows some modeling information of the FIML estimation of the confirmatory factor model on the missing data.
PROC CALIS shows you that the number of complete observations is 16 and the number of incomplete observations is 16 in the data set. All these observations are included in the estimation. The analysis type is ‘Means and Covariances’ because with full information maximum likelihood, the sample means have to be analyzed during the estimation.

For the full information maximum likelihood estimation, PROC CALIS outputs several tables to summarize the missing data patterns and statistics. Output 29.15.4 shows the proportions of data that are present for the variables, individually or jointly by pairs.

The diagonal elements of the table in Output 29.15.4 show the proportions of data coverage by each of the variables. The off-diagonal elements show the proportions of joint data coverage by all possible pairs of variables. For example, the first diagonal element of the table shows that about 94% of the observations...
have \( x_1 \) values that are not missing. This percentage value is referred to as the proportion coverage for \( x_1 \) or the proportion coverage for computing the means of \( x_1 \). The off-diagonal element for \( x_1 \) and \( x_2 \) shows that about 78% of the observations have nonmissing values for both their \( x_1 \) and \( x_2 \) values. This percentage value is referred to as the joint proportion coverage of \( x_1 \) and \( x_2 \) or the proportion coverage for computing the covariance between \( x_1 \) and \( x_2 \). The larger the coverage proportions this table shows, the more relative information the data contain for estimating the corresponding moments.

To summarize the proportion coverage, Output 29.15.4 shows that on average about 91% of the data are nonmissing for computing the means, and about 82% of the data are nonmissing for computing the covariances.

Output 29.15.5 shows the lowest coverage proportions of the means and the covariances.

**Output 29.15.5** Ranking the Lowest Coverage Proportions: Missing Data

<table>
<thead>
<tr>
<th>Rank Order of the 3 Smallest Variable (Mean) Coverages</th>
<th>Variable</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>x2</td>
<td>0.8438</td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>0.8750</td>
<td></td>
</tr>
<tr>
<td>y3</td>
<td>0.8750</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rank Order of the 7 Smallest Covariance Coverages</th>
<th>Var1</th>
<th>Var2</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>x3</td>
<td>x2</td>
<td>0.7188</td>
<td></td>
</tr>
<tr>
<td>y3</td>
<td>x2</td>
<td>0.7188</td>
<td></td>
</tr>
<tr>
<td>y3</td>
<td>x3</td>
<td>0.7500</td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td>x1</td>
<td>0.7813</td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>x1</td>
<td>0.8125</td>
<td></td>
</tr>
<tr>
<td>y1</td>
<td>x2</td>
<td>0.8125</td>
<td></td>
</tr>
<tr>
<td>y1</td>
<td>x3</td>
<td>0.8125</td>
<td></td>
</tr>
</tbody>
</table>

The first table of Output 29.15.5 shows that \( x_2 \) has the lowest proportion coverage at about 84%, and \( x_3 \) and \( y_3 \) are the next at about 88%. The second table of Output 29.15.5 shows that the joint proportion coverage by the \( x_3-x_2 \) pair and the \( y_3-x_2 \) pair are the lowest at about 72%, followed by the \( y_3-x_3 \) pair at 75%. These two tables are useful to diagnose which variables most lack the information for estimation. For this data set, these tables show that estimation related to the moments of \( x_2 \), \( x_3 \), and \( y_3 \) suffers the missing data problem the most. However, because the worst proportion coverage is still higher than 70%, the missingness problem does not seem to be very serious based on percentage.

In Output 29.15.6, PROC CALIS outputs two tables that show an overall picture of the missing patterns in the data set.
Chapter 29: The CALIS Procedure

Output 29.15.6 The Most Frequent Missing Patterns and Their Mean Profiles: Missing Data

<table>
<thead>
<tr>
<th>Rank Order of the 5 Most Frequent Missing Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of Distinct Patterns with Missing Values = 7</td>
</tr>
<tr>
<td>Pattern</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

NOTE: Nonmissing Pattern Proportion = 0.5000 (N=16)

<table>
<thead>
<tr>
<th>Means of the Nonmissing and the Most Frequent Missing Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing Pattern</td>
</tr>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>x1</td>
</tr>
<tr>
<td>x2</td>
</tr>
<tr>
<td>x3</td>
</tr>
<tr>
<td>y1</td>
</tr>
<tr>
<td>y2</td>
</tr>
<tr>
<td>y3</td>
</tr>
</tbody>
</table>

The first table of Output 29.15.6 shows that “x.xxxx” and “xx.xxx” are the two most frequent missing patterns in the data set. Each has a frequency of 4. An “x” in the missing pattern denotes a nonmissing value, while a “.” denotes a missing value. Hence, the first pattern has all missing values for the second variable, and the second pattern has all missing values for the third variable. Each of these two missing patterns accounts for 12.5% of the total observations. Together, the five missing patterns shown in Output 29.15.6 account for about 43.8% of the total observations. The note after this table shows that 50% of the total observations do not have any missing values.

To determine exactly which variables are missing in the missing patterns, it is useful to consult the second table in Output 29.15.6. In this table, the variable means of the most frequent missing patterns are shown, together with the variable means of the nonmissing pattern for comparisons. Missing means in this table show that the corresponding variables are not present in the missing patterns. For example, the column labeled “Nonmissing” is for the group of 16 observations that do not have any missing values. Each of the variable means is computed based on 16 observations. The next column labeled “1” is the first missing pattern that has four observations. The variable mean for x2 is missing for this missing pattern group, while each of the other variable means is computed based on four observations. Comparing these means with those in the nonmissing group, it shows that the means for x1, x3, and y1 in the first missing pattern are smaller than those in the nonmissing group, while the means for y2 and y3 are greater. This comparison does not seem to suggest any systematic bias in the means of the first missing pattern group.

However, the nonmissing means in the third missing pattern (the column labeled “3”) do show a consistent downward bias, as compared with the means in the nonmissing group. This might mean that respondents with low scores in x1–x3, y1, and y2 tend not to respond to y3 for some reason. Similarly, the fourth missing pattern shows a consistent downward bias in x2, x3, and y1–y3. Whether these patterns suggest a systematic (or nonrandom) pattern of missingness must be judged in the substantive context. Nonetheless, the numerical results if Output 29.15.6 provide some insight on this matter.
The tables shown in Output 29.15.6 do not show all the missing patterns. In general, PROC CALIS shows only the most frequent or dominant missing patterns so that the output results are more focused. By default, if the total number of missing patterns in a data set is below six, then PROC CALIS shows all the missing patterns. If the total number of missing patterns is at least six, PROC CALIS shows up to 10 missing patterns provided that each of these missing patterns accounts for at least 5% of the total observations. The 10 missing patterns is the default maximum number of missing patterns to show, and the 5% is the default proportion threshold for a missing pattern to display. You can override the default maximum number of missing patterns by the MAXMISSPAT= option and the proportion threshold by the TMISSPAT= option.

Output 29.15.7 shows the parameter estimates by the FIML estimation.

**Output 29.15.7** Parameter Estimates of the CFA Model with FIML: Missing Data

<table>
<thead>
<tr>
<th>Factor Loading Matrix:</th>
<th>Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>verbal</td>
</tr>
<tr>
<td>x1</td>
<td>5.5003 1.0025 5.4867 &lt;.0001</td>
</tr>
<tr>
<td>_Parm1</td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td>5.7134 0.9956 5.7385 &lt;.0001</td>
</tr>
<tr>
<td>_Parm2</td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>4.4417 0.7669 5.7918 &lt;.0001</td>
</tr>
<tr>
<td>_Parm3</td>
<td></td>
</tr>
<tr>
<td>y1</td>
<td>0 4.9277 0.6798 7.2491 &lt;.0001</td>
</tr>
<tr>
<td>_Parm4</td>
<td></td>
</tr>
<tr>
<td>y2</td>
<td>0 4.1215 0.5716 7.2100 &lt;.0001</td>
</tr>
<tr>
<td>_Parm5</td>
<td></td>
</tr>
<tr>
<td>y3</td>
<td>0 3.3834 0.6145 5.5058 &lt;.0001</td>
</tr>
<tr>
<td>_Parm6</td>
<td></td>
</tr>
</tbody>
</table>
First, you can compare the current FIML results with the results in Example 29.12, where maximum likelihood method is used with the complete data set. Overall, the estimates of loadings, factor covariance, and error variances are similar in the two analyses. Next, you compare the current FIML results with the results in Output 29.15.2, where the default ML method is applied to the same data set with missing values. Except for the standard error estimate of the factor covariance, which are very similar with ML and FIML, the standard error estimates with FIML are consistently smaller than those with ML in Output 29.15.2. This means that with FIML, you improve the estimation efficiency by including the partial information in those observations with missing values.

When you have a data set with no missing values, the ML and FIML methods, as implemented in PROC CALIS, are theoretically the same. Both are equally efficient and produce similar estimates (see Example 29.16). FIML and ML are the same estimation technique that maximizes the likelihood function under the multivariate normal distribution. However, in PROC CALIS, the distinction between ML and FIML concerns different treatments of the missing values. With METHOD=ML, all observations with one or more missing values are discarded from the analysis. With METHOD=FIML, all observations with at least one nonmissing value are included in the analysis.

### Output 29.15.7 continued

<table>
<thead>
<tr>
<th>Factor Covariance Matrix: Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbal</td>
</tr>
<tr>
<td>verbal</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>math</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

### Error Variances

<table>
<thead>
<tr>
<th>Variable</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>_Add08</td>
<td>12.72770</td>
<td>4.77627</td>
<td>2.6648</td>
<td>0.0077</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td>_Add09</td>
<td>9.35994</td>
<td>4.48806</td>
<td>2.0855</td>
<td>0.0370</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>_Add10</td>
<td>5.67393</td>
<td>2.69872</td>
<td>2.1025</td>
<td>0.0355</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y1</td>
<td>_Add11</td>
<td>1.86768</td>
<td>1.36676</td>
<td>1.3665</td>
<td>0.1718</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y2</td>
<td>_Add12</td>
<td>1.49942</td>
<td>0.97322</td>
<td>1.5407</td>
<td>0.1234</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y3</td>
<td>_Add13</td>
<td>5.24973</td>
<td>1.54121</td>
<td>3.4062</td>
<td>0.0007</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 29.16: Comparing the ML and FIML Estimation

This example uses the complete data set from Example 29.12 to illustrate how the maximum likelihood (ML) and full information maximum likelihood (FIML) methods are theoretically equivalent when you apply them to data set without missing values. In Example 29.15, you apply a confirmatory factor model to a data set with missing values. You find that with METHOD=FIML, you can get more stable estimates than with METHOD=ML (which is the default estimation method). Near the end of Example 29.15, you learn that ML and FIML are theoretically equivalent estimation methods when you apply them to data sets without missing values.

However, the ML and FIML methods have two major computational differences in their implementations in PROC CALIS. First, with METHOD=FIML the first-order properties (that is, the means of the variables) of the data are automatically included in the analysis. However, by default you analyze only the second-order properties (that is, the covariances of the variables) with METHOD=ML. Second, the biased sample covariance formula (with N as the variance divisor) is used with METHOD=FIML, while the unbiased sample covariance formula (with DF=N – 1 as the variance divisor) is used with METHOD=ML. See the section “Relationships among Estimation Criteria” on page 1473 for more details about the similarities and differences between the ML and FIML methods.

If you take care of these two differences between ML and FIML in PROC CALIS, you can obtain exactly the same results with these two methods when you apply them to data sets without missing values.

For example, with the complete data set scores from Example 29.12, you specify the FIML estimation in the following statements:

```sas
proc calis method=fiml data=scores;
  factor
    verbal ===> x1-x3,
    math  ===> y1-y3;
  pvar
    verbal = 1.,
    math   = 1.;
run;
```

An equivalent specification with the ML method is shown in the following statements:

```sas
proc calis method=ml meanstr vardef=n data=scores;
  factor
    verbal ===> x1-x3,
    math  ===> y1-y3;
  pvar
    verbal = 1.,
    math   = 1.;
run;
```
In the PROC CALIS statement, you specify two options to make the ML estimation exactly equivalent to the FIML estimation in PROC CALIS. First, the MEANSTR option requests the first-order properties (the mean structures) to be analyzed with the covariance structures. Second, the VARDEF=N option defines the variance divisor to N, instead of the default DF, which is the same as N–1. These two options make the ML estimation equivalent to the FIML estimation.

Output 29.16.1 and Output 29.16.2 show some fit summary statistics under the FIML and ML methods, respectively.

**Output 29.16.1** Model Fitting by the FIML Method: Scores Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fit Function</td>
<td>31.7837</td>
</tr>
<tr>
<td>Chi-Square</td>
<td>10.1215</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>8</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.2566</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0571</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0910</td>
</tr>
<tr>
<td>Bentler Comparative Fit Index</td>
<td>0.9872</td>
</tr>
</tbody>
</table>

*NOTE: Saturated mean structure parameters are excluded from the computations of fit indices.*

**Output 29.16.2** Model Fitting by the ML Method: Scores Data

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fit Function</td>
<td>0.3163</td>
</tr>
<tr>
<td>Chi-Square</td>
<td>10.1215</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>8</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.2566</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0504</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0910</td>
</tr>
<tr>
<td>Bentler Comparative Fit Index</td>
<td>0.9872</td>
</tr>
</tbody>
</table>

Except for the fit function values, both FIML and ML methods produce the same set of fit statistics. The difference in the fit function values is expected because the FIML function has a constant term which is derived from the likelihood function. This constant term does not depend on the model parameters. Hence, the FIML and ML discrepancy functions that are used in PROC CALIS are equivalent when VARDEF=N is used in the ML method for analyzing mean and covariance structures.

The parameter estimates are shown in Output 29.16.3 and Output 29.16.4 for the FIML and ML methods, respectively. Except for very tiny numerical differences in some estimates, the FIML and ML estimates match.
### Example 29.16: Comparing the ML and FIML Estimation

#### Output 29.16.3 Parameter Estimates by the FIML Method: Scores Data

**Factor Loading Matrix:**

<table>
<thead>
<tr>
<th>Parameter Estimates</th>
<th>verbal</th>
<th>math</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>5.7486</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.9651</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.9567</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ Parm1]</td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td>5.7265</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.9239</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.1980</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ Parm2]</td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>4.5886</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.7570</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.0618</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ Parm3]</td>
<td></td>
</tr>
<tr>
<td>y1</td>
<td>0</td>
<td>5.1972</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.6779</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.6662</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td></td>
<td>[ Parm4]</td>
<td></td>
</tr>
<tr>
<td>y2</td>
<td>0</td>
<td>4.1342</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.6025</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.8612</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td></td>
<td>[ Parm5]</td>
<td></td>
</tr>
<tr>
<td>y3</td>
<td>0</td>
<td>3.7004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.6143</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.0237</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td></td>
<td>[ Parm6]</td>
<td></td>
</tr>
</tbody>
</table>

**Factor Covariance Matrix:**

<table>
<thead>
<tr>
<th>Parameter Estimates</th>
<th>verbal</th>
<th>math</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbal</td>
<td>1.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5175</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1406</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.6804</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.000233</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ Add01]</td>
<td></td>
</tr>
<tr>
<td>math</td>
<td>0.5175</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>0.1406</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.6804</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.000233</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ Add01]</td>
<td></td>
</tr>
</tbody>
</table>

**Intercepts**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>_Add02</td>
<td>19.90625</td>
<td>1.17540</td>
<td>16.9357</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td>_Add03</td>
<td>18.81250</td>
<td>1.14089</td>
<td>16.4893</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>_Add04</td>
<td>18.68750</td>
<td>0.92749</td>
<td>20.1486</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y1</td>
<td>_Add05</td>
<td>17.90625</td>
<td>0.93161</td>
<td>19.2208</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y2</td>
<td>_Add06</td>
<td>17.84375</td>
<td>0.78823</td>
<td>22.6377</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y3</td>
<td>_Add07</td>
<td>17.75000</td>
<td>0.76419</td>
<td>23.2272</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Output 29.16.3 continued

<table>
<thead>
<tr>
<th>Variable</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>_Add08</td>
<td>11.16406</td>
<td>4.06574</td>
<td>2.7459</td>
<td>0.0060</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td>_Add09</td>
<td>8.85978</td>
<td>3.65403</td>
<td>2.4247</td>
<td>0.0153</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>_Add10</td>
<td>6.47248</td>
<td>2.47685</td>
<td>2.6132</td>
<td>0.0090</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y1</td>
<td>_Add11</td>
<td>0.76135</td>
<td>1.23420</td>
<td>0.6169</td>
<td>0.5373</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y2</td>
<td>_Add12</td>
<td>2.79060</td>
<td>1.04306</td>
<td>2.6754</td>
<td>0.0075</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y3</td>
<td>_Add13</td>
<td>4.99466</td>
<td>1.40025</td>
<td>3.5670</td>
<td>0.0004</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Output 29.16.4 Parameter Estimates by the ML Method: Scores Data

<table>
<thead>
<tr>
<th>verbal</th>
<th>math</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>5.7486 0.9651 5.9567 &lt;.0001 [Parm1]</td>
</tr>
<tr>
<td>x2</td>
<td>5.7265 0.9239 6.1981 &lt;.0001 [Parm2]</td>
</tr>
<tr>
<td>x3</td>
<td>4.5885 0.7570 6.0617 &lt;.0001 [Parm3]</td>
</tr>
<tr>
<td>y1</td>
<td>0 5.1972 0.6779 7.6662 &lt;.0001 [Parm4]</td>
</tr>
<tr>
<td>y2</td>
<td>0 4.1341 0.6025 6.8612 &lt;.0001 [Parm5]</td>
</tr>
<tr>
<td>y3</td>
<td>0 3.7004 0.6143 6.0238 &lt;.0001 [Parm6]</td>
</tr>
</tbody>
</table>
The equivalence between METHOD=ML and METHOD=FIML implies that if you do not have any missing data in your data, you can just use METHOD=ML because it is computationally more efficient than the FIML method.

While the equivalence between ML and FIML is established here with the use of the VARDEF= and MEANSTR options (for data without missing values), it is not necessary in practice to use these options with METHOD=ML. The VARDEF= option is used in this example only to demonstrate the theoretical equivalence between METHOD=ML and METHOD=FIML. The VARDEF= option has very little effect if you have at least a moderate sample size (for example, 30 or more observations).

Merely adding the MEANSTR option to an analysis for data without missing values amounts to adding a saturated mean structure to a covariance structure analysis. In this case, the MEANSTR option only gives you more estimates that pertain to the mean structures, but the parameter estimates that pertain to the covariance structures do not change. Therefore, use the MEANSTR option only when you need to estimate certain mean structure parameters or when you fit models with nonsaturated mean structures.

However, use METHOD=FIML when there are missing values in your data and you need to use every bit of information from the incomplete observations with random missing values.
Example 29.17: Path Analysis: Stability of Alienation

The following covariance matrix from Wheaton et al. (1977) has served to illustrate the performance of several implementations for the analysis of structural equation models. Two different models have been analyzed by an early implementation of LISREL and are mentioned in Jöreskog (1978). You can also find a more detailed discussion of these models in the LISREL VI manual (Jöreskog and Sörbom 1985). A slightly modified model for this covariance matrix is included in the EQS 2.0 manual (Bentler 1985, p. 28). However, for the analysis with the EQS implementation, the SEI variable is rescaled by a factor of 0.1 to make the matrix less ill-conditioned. Since the Levenberg-Marquardt or Newton-Raphson optimization techniques are used with PROC CALIS, rescaling the data matrix is not necessary and, therefore, is not done here. The results reported here reflect the estimates based on the original covariance matrix.

The path diagram of this model is displayed in Figure 29.1 and is reproduced in the following:

You use the PATH modeling language of PROC CALIS to specify this path model, as shown in the following statements:
Since no METHOD= option is used in the PROC CALIS statement, maximum likelihood estimates are computed by default.

In the PATH statement, you specify the functional relationships of the variables in the model. These functional relationships are represented as single-headed paths in the path diagram. There are five entries in the PATH statement. You specify the relationships between the latent constructs and the observed variables in the first three path entries. For example, the first entry states that Anomie and Powerless67 are measured indicators of the latent variable Alien67. The path effects or coefficients from the latent factor to these measured indicators
are fixed at 1.0 and 0.833, respectively. Similarly, in the next two path entries, you define the relationships between the latent factors Alien71 and SES and their measured indicators. The last two path entries in the PATH statement represent the functional relationships among the latent variables in the model. SES has effects on Alien67 and Alien71. These effect parameters are labeled or named with gamma1 and gamma2, respectively. Alien67 also has an effect on Alien71, with the effect parameter named beta.

In the PVAR statement, you specify the variance or error variance parameters in the model. These parameters correspond to the double-headed arrows pointing to the individual variables in the path diagram. In the first six entries of the PVAR statement, you specify the error variance parameters of the observed variables. You also give names to these parameters that correspond to the notation in the path diagram. Although you can choose any names for the parameters, it is important to remember that parameters with the same name are identical and will have the same estimates. For example, the error variances of Anomie67 and Anomie71 are the same parameter named theta1. Similarly, you constrain the error variances of Powerless67 and Powerless71. However, the error variance parameters of Education and SEI are unique. They are not constrained with other parameters in the model because they have unique parameter names. Next, you specify the error variance parameters of Alien67 and Alien71. They also have unique parameter names and therefore they are not constrained with any other parameters in the model. Lastly, you specify the variance parameter phi of SES.

In the PCOV statement, you specify the covariances or error covariances among variables in the model. These parameters correspond to the double-headed arrows pointing to distinct pairs of variables in the path diagram. Observed variables Anomie67 and Anomie71 have correlated errors and you specify this error covariance parameter as theta5. Similarly, observed variables Powerless67 and Powerless71 have correlated errors and you also specify this error covariance parameter as theta5. This way, the two error covariances are constrained to be equal.

PROC CALIS can produce a high-quality residual histogram that is useful for showing the distribution of residuals. Before you request the residual histogram, ODS Graphics must be enabled. For example, you can specify the ODS GRAPHICS ON statement, as shown in the preceding statements before the PROC CALIS statement. Then, the residual histogram is requested by the plots=residuals option in the PROC CALIS statement. PROC CALIS can also produce a high-quality path diagram for the model. You can use the PATHDIAGRAM statement to request the path diagram and to specify related options.

Output 29.17.1 displays the modeling information and variables in the analysis.

Output 29.17.1 Model Specification and Variables

PATH Model Specification

The CALIS Procedure
Covariance Structure Analysis: Model and Initial Values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling Information</td>
<td></td>
</tr>
<tr>
<td>Maximum Likelihood</td>
<td></td>
</tr>
<tr>
<td>Estimation</td>
<td></td>
</tr>
<tr>
<td>Data Set</td>
<td>WORK.WHEATON</td>
</tr>
<tr>
<td>N Obs</td>
<td>932</td>
</tr>
<tr>
<td>Model Type</td>
<td>PATH</td>
</tr>
<tr>
<td>Analysis</td>
<td>Covariances</td>
</tr>
</tbody>
</table>
Output 29.17.1 continued

<table>
<thead>
<tr>
<th>Variables in the Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Endogenous</strong></td>
</tr>
<tr>
<td><strong>Latent</strong></td>
</tr>
<tr>
<td><strong>Exogenous</strong></td>
</tr>
<tr>
<td><strong>Latent</strong></td>
</tr>
</tbody>
</table>

Number of Endogenous Variables = 8  
Number of Exogenous Variables = 1

Output 29.17.1 shows that the data set Wheaton was used with 932 observations. The model is specified with the PATH modeling language. Variables in the model are classified into different categories according to their roles. All manifest variables are endogenous in the model. Also, three latent variables are hypothesized in the model: Alien67, Alien71, and SES. While Alien67 and Alien71 are endogenous, SES is exogenous in the model.

Output 29.17.2 echoes the initial specification of the PATH model.

**Output 29.17.2 Initial Estimates**

<table>
<thead>
<tr>
<th>Initial Estimates for PATH List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
</tr>
<tr>
<td>Anomie67 &lt;= Alien67</td>
</tr>
<tr>
<td>Powerless67 &lt;= Alien67</td>
</tr>
<tr>
<td>Anomie71 &lt;= Alien71</td>
</tr>
<tr>
<td>Powerless71 &lt;= Alien71</td>
</tr>
<tr>
<td>Education &lt;= SES</td>
</tr>
<tr>
<td>SEI &lt;= SES</td>
</tr>
<tr>
<td>Alien67 &lt;= SES</td>
</tr>
<tr>
<td>Alien71 &lt;= SES</td>
</tr>
<tr>
<td>Alien71 &lt;= Alien67</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial Estimates for Variance Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance Type</td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td>Powerless67</td>
</tr>
<tr>
<td>Anomie71</td>
</tr>
<tr>
<td>Powerless71</td>
</tr>
<tr>
<td>Education</td>
</tr>
<tr>
<td>SEI</td>
</tr>
<tr>
<td>Alien67</td>
</tr>
<tr>
<td>Alien71</td>
</tr>
<tr>
<td>Exogenous SES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial Estimates for Covariances Among Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error of Error</td>
</tr>
<tr>
<td>Anomie67</td>
</tr>
<tr>
<td>Powerless67</td>
</tr>
</tbody>
</table>
The numerical values for estimates in Output 29.17.2 are the initial values that you input in the model specification. A blank value for the associated parameter name for a numerical estimate indicates that the estimate is a fixed value, which would not be changed in the estimation. For example, the first five paths have fixed path coefficients, and their fixed values are shown in the Estimate column. For numerical estimates that have specified parameter names, the numerical values serve as initial values, which would be changed during the estimation. Output 29.17.2 does not actually have this type of specification. Missing values ‘.’ are specified as initial values for all the free parameters that are specified in the model. For example, lambda, gamma1, theta1, and psi1, among others, are free parameters that do not have specified initial values. PROC CALIS automatically generates the initial values of these parameters.

You can examine this output to ensure that the desired model is being analyzed. PROC CALIS outputs the initial specifications or the estimation results in the order you specify in the model, unless you use reordering options such as ORDERSPEC and ORDERALL. Therefore, the input order of specifications is important—it determines how your output would look.

Simple descriptive statistics are displayed in Output 29.17.3.

### Output 29.17.3 Descriptive Statistics

<table>
<thead>
<tr>
<th>Simple Statistics</th>
<th>Mean</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67</td>
<td>0</td>
<td>3.44006</td>
</tr>
<tr>
<td>Powerless67</td>
<td>0</td>
<td>3.06007</td>
</tr>
<tr>
<td>Anomie71</td>
<td>0</td>
<td>3.54006</td>
</tr>
<tr>
<td>Powerless71</td>
<td>0</td>
<td>3.16006</td>
</tr>
<tr>
<td>Education</td>
<td>0</td>
<td>3.10000</td>
</tr>
<tr>
<td>SEI</td>
<td>0</td>
<td>21.21999</td>
</tr>
</tbody>
</table>

Because the input data set contains only the covariance matrix, the means of the manifest variables are assumed to be zero. Note that this has no impact on the estimation, unless a mean structure model is being analyzed.

Initial estimates are necessary in all kinds of optimization problems. You can provide these initial estimates or let PROC CALIS to generate them automatically. As shown in Output 29.17.2, you did not provide any initial estimates for the parameters. PROC CALIS uses a combination of well-behaved mathematical methods to complete the initial estimation. The initial estimation methods for the current analysis are shown in Output 29.17.4.

### Output 29.17.4 Optimization Starting Point

<table>
<thead>
<tr>
<th>Initial Estimation Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Instrumental Variables Method</td>
</tr>
<tr>
<td>2 McDonald Method</td>
</tr>
<tr>
<td>3 Two-Stage Least Squares</td>
</tr>
</tbody>
</table>
In this example, the instrumental variable Method, the McDonald and Hartmann method, and the two-stage least squares method have been used for initial estimation. In the same output, the vector of initial parameter estimates and their gradients are also shown. The initial objective function value is 0.0366.

Output 29.17.5 displays the optimization information, including technical details, iteration history and convergence status.
The convergence status is important for the validity of your solution. In most cases, you should interpret your results only when the solution is converged. In this example, you obtain a converged solution, as shown in the message at the bottom of the table. The final objective function value is 0.01448, which is the minimized function value during the optimization. If problematic solutions such as nonconvergence are encountered, PROC CALIS issues an error message.

The fit summary statistics are displayed in Output 29.17.6. By default, PROC CALIS displays all available fit indices and modeling information.

### Output 29.17.6  Fit Summary

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modeling Info</strong></td>
<td></td>
</tr>
<tr>
<td>Number of Observations</td>
<td>932</td>
</tr>
<tr>
<td>Number of Variables</td>
<td>6</td>
</tr>
<tr>
<td>Number of Moments</td>
<td>21</td>
</tr>
<tr>
<td>Number of Parameters</td>
<td>12</td>
</tr>
<tr>
<td>Number of Active Constraints</td>
<td>0</td>
</tr>
<tr>
<td>Baseline Model Function Value</td>
<td>2.2894</td>
</tr>
<tr>
<td>Baseline Model Chi-Square</td>
<td>2131.4327</td>
</tr>
<tr>
<td>Baseline Model Chi-Square DF</td>
<td>15</td>
</tr>
<tr>
<td>Pr &gt; Baseline Model Chi-Square</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td><strong>Absolute Index</strong></td>
<td></td>
</tr>
<tr>
<td>Fit Function</td>
<td>0.0145</td>
</tr>
<tr>
<td>Chi-Square</td>
<td>13.4851</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>9</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.1419</td>
</tr>
<tr>
<td>Z-Test of Wilson &amp; Hilferty</td>
<td>1.0754</td>
</tr>
<tr>
<td>Hoelter Critical N</td>
<td>1169</td>
</tr>
<tr>
<td>Root Mean Square Residual (RMR)</td>
<td>0.2281</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0150</td>
</tr>
<tr>
<td>Goodness of Fit Index (GFI)</td>
<td>0.9953</td>
</tr>
<tr>
<td><strong>Parsimony Index</strong></td>
<td></td>
</tr>
<tr>
<td>Adjusted GFI (AGFI)</td>
<td>0.9890</td>
</tr>
<tr>
<td>Parsimonious GFI</td>
<td>0.5972</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0231</td>
</tr>
<tr>
<td>RMSEA Lower 90% Confidence Limit</td>
<td>0.0000</td>
</tr>
<tr>
<td>RMSEA Upper 90% Confidence Limit</td>
<td>0.0470</td>
</tr>
<tr>
<td>Probability of Close Fit</td>
<td>0.9705</td>
</tr>
<tr>
<td>ECVI Estimate</td>
<td>0.0405</td>
</tr>
<tr>
<td>ECVI Lower 90% Confidence Limit</td>
<td>0.0357</td>
</tr>
<tr>
<td>ECVI Upper 90% Confidence Limit</td>
<td>0.0556</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>37.4851</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>107.5330</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>95.5330</td>
</tr>
<tr>
<td>McDonald Centrality</td>
<td>0.9976</td>
</tr>
<tr>
<td><strong>Incremental Index</strong></td>
<td></td>
</tr>
<tr>
<td>Bentler Comparative Fit Index</td>
<td>0.9979</td>
</tr>
<tr>
<td>Bentler-Bonett NFI</td>
<td>0.9937</td>
</tr>
<tr>
<td>Bentler-Bonett Non-normed Index</td>
<td>0.9965</td>
</tr>
<tr>
<td>Bollen Normed Index Rho1</td>
<td>0.9895</td>
</tr>
<tr>
<td>Bollen Non-normed Index Delta2</td>
<td>0.9979</td>
</tr>
<tr>
<td>James et al. Parsimonious NFI</td>
<td>0.5962</td>
</tr>
</tbody>
</table>
First, the fit summary table starts with some basic modeling information, as shown in Output 29.17.6. You can check the number of observations, number of variables, number of moments being fitted, number of parameters, number of active constraints in the solution, and the independent model chi-square and its degrees of freedom in this modeling information category. Next, three types of fit indices are shown: absolute, parsimony, and incremental.

The absolute indices are fit measures that you interpret them without referring to any baseline model. These indices do not adjust for model parsimony. They always favor models with a large number of parameters. The chi-square test statistic is the best-known absolute index in this category. In this example, the $p$-value of the chi-square is 0.1419, which is greater than the conventional 0.05 value. From the statistical hypothesis testing point of view, you cannot reject this model. The Z-test of Wilson and Hilferty is also insignificant at $\alpha = .05$, which echoes the result of the chi-square test. You can consult other absolute indices as well. Although it seems that there are no clear conventional levels for these absolute indices to indicate an acceptable model fit, you can always use these indices to compare the relative fit among competing models.

Next, the parsimony fit indices take the model parsimony into account. These indices adjust the model fit by the degrees of freedom (or the number of the parameters) of the model in certain ways. The advantage of these indices is that merely increasing the number of parameters in the model might not necessarily lead to better model fit measures. These fit indices penalize models with large numbers of parameters. There is no universal way to interpret all these indices. However, for the relatively well-known RMSEA estimate, by convention values under 0.05 indicate good model fit. The RMSEA value for this example is 0.0231, and so this is a very good model fit. For interpretations of other parsimony indices, you can consult the original articles for these indices.

Last, the incremental fit indices are computed based on comparing the target model fit against the fit of a baseline model, which is usually the so-called uncorrelatedness model where all manifest variables are assumed to be uncorrelated. This is the baseline model that PROC CALIS uses. The baseline model fit statistic is shown under the ‘Modeling Info’ category of the same fit summary table. In this example, the model fit chi-square of the baseline model is 2131.43, with 15 degrees of freedom. The incremental indices show how well the hypothesized model improves over the baseline model for the data. Various incremental fit indices have been proposed. In the fit summary table, there are six of such fit indices. Large values for these indices are desired. It has been suggested that values greater than .9 for these indices indicate acceptable model fit. In this example, all incremental indices but James et al. parsimonious NFI show that the hypothesized model fits well.

There is no consensus as to which fit index is the best to judge model fit. Probably, with artificial data and model, all fit indices can be shown defective in some aspects of measuring model fit. Conventional wisdom is to look at all fit indices and determine whether the majority of them are close to the desirable ranges of values. In this example, almost all fit indices are good, and so it is safe to conclude that the model fits well.

Nowadays, most researchers pay less attention to the model fit chi-square statistic because it tends to reject all meaningful models with minimum departures from the truth. Although the model fit chi-square test statistic is an impeccable statistical inference tool when the underlying statistical assumptions are satisfied, for practical purposes it is just too powerful to accept any useful and reasonable models with only tiny imperfections. Some fit indices are more popular than others. Standardized RMR, RMSEA estimate, adjusted AGFI, and Bentler’s comparative fit index are frequently reported in empirical research for judging model fit. In this example, all these measures show good model fit of the hypothesized model. While there are certainly legitimate reasons why these fit indices are more popular than others, they are out of the current scope of discussion.
Figure 29.17.7 shows the path diagram of the unstandardized solution. The path diagram indicates significant estimates by attaching asterisks to the numerical values. Estimates that are flagged with one asterisk are significant at 0.05 $\alpha$-level. Estimates that are flagged with two asterisks are significant at 0.01 $\alpha$-level. The path diagram also shows a summary of fit statistics. For more information about specifying path diagram output, see the section “Path Diagrams: Layout Algorithms, Default Settings, and Customization” on page 1440.

**Output 29.17.7** Path Diagram and Fit Summary

PROC CALIS can perform a detailed residual analysis. Large residuals might indicate misspecification of the model. In **Output 29.17.8**, raw residuals are reported and ranked.
Output 29.17.8  Raw Residuals and Ranking

<table>
<thead>
<tr>
<th>Raw Residual Matrix</th>
<th>Anomie67</th>
<th>Powerless67</th>
<th>Anomie71</th>
<th>Powerless71</th>
<th>Education</th>
<th>SEI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67</td>
<td>-0.06997</td>
<td>0.03642</td>
<td>-0.01116</td>
<td>-0.15200</td>
<td>0.32892</td>
<td>0.47786</td>
</tr>
<tr>
<td>Powerless67</td>
<td>0.03642</td>
<td>0.01261</td>
<td>0.15600</td>
<td>0.01135</td>
<td>-0.41712</td>
<td>-0.19108</td>
</tr>
<tr>
<td>Anomie71</td>
<td>-0.01116</td>
<td>0.15600</td>
<td>-0.08381</td>
<td>-0.00854</td>
<td>0.22464</td>
<td>0.07976</td>
</tr>
<tr>
<td>Powerless71</td>
<td>-0.15200</td>
<td>0.01135</td>
<td>-0.00854</td>
<td>0.14067</td>
<td>-0.23832</td>
<td>-0.59248</td>
</tr>
<tr>
<td>Education</td>
<td>0.32892</td>
<td>-0.41712</td>
<td>0.22464</td>
<td>-0.23832</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>SEI</td>
<td>0.47786</td>
<td>-0.19108</td>
<td>0.07976</td>
<td>-0.59248</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
</tbody>
</table>

Average Absolute Residual  0.153940
Average Off-diagonal Absolute Residual  0.195044

<table>
<thead>
<tr>
<th>Rank Order of the 10 Largest Raw Residuals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Var1</td>
</tr>
<tr>
<td>SEI</td>
</tr>
<tr>
<td>SEI</td>
</tr>
<tr>
<td>Education</td>
</tr>
<tr>
<td>Education</td>
</tr>
<tr>
<td>Education</td>
</tr>
<tr>
<td>Education</td>
</tr>
<tr>
<td>SEI</td>
</tr>
<tr>
<td>Anomie71</td>
</tr>
<tr>
<td>Powerless71</td>
</tr>
<tr>
<td>Powerless71</td>
</tr>
</tbody>
</table>

Because of the differential scaling of the variables, it is usually more useful to examine the standardized residuals instead. In Output 29.17.9, for example, the table for the 10 largest asymptotically standardized residuals is displayed.
The model performs the poorest concerning the covariances of Education with all measures of Powerless and Anomie. This might suggest a misspecification of the functional relationships of Education with other variables in the model. However, because the model fit is quite good, such a possible misspecification should not be a serious concern in the analysis.

The histogram of the asymptotically standardized residuals is displayed in Output 29.17.10, which also shows the normal and kernel approximations.
The residual distribution looks quite symmetrical. It shows a small to medium departure from the normal distribution, as evidenced by the discrepancies between the kernel and the normal distribution curves.

**Output 29.17.11** shows the estimation results.

<table>
<thead>
<tr>
<th>PATH List</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATH</td>
</tr>
<tr>
<td>Anomie67</td>
</tr>
<tr>
<td>Powerless67</td>
</tr>
<tr>
<td>Anomie71</td>
</tr>
<tr>
<td>Powerless71</td>
</tr>
<tr>
<td>Education</td>
</tr>
<tr>
<td>SEI</td>
</tr>
<tr>
<td>Alien67</td>
</tr>
<tr>
<td>Alien71</td>
</tr>
<tr>
<td>Alien71</td>
</tr>
</tbody>
</table>
The paths, variances and partial (or error) variances, and covariances and partial covariances are shown. When you have fixed parameters such as the first five path coefficients in the output, the standard errors and \( t \) values are all blanks. For free or constrained estimates, standard errors and \( t \) values are computed. Researchers in structural equation modeling usually use the value 2 as an approximate critical value for the observed \( t \) values. The reason is that the estimates are asymptotically normal, and so the two-sided critical point with \( \alpha = 0.05 \) is 1.96, which is close to 2. Using this criterion, all estimates shown in Output 29.17.11 are significantly different from zero, supporting the presence of these parameters in the model.

Squared multiple correlations are shown in Output 29.17.12.

**Output 29.17.12** Squared Multiple Correlations

<table>
<thead>
<tr>
<th>Variable</th>
<th>Error Variance</th>
<th>Total Variance</th>
<th>R-Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67</td>
<td>3.60796</td>
<td>11.90397</td>
<td>0.6969</td>
</tr>
<tr>
<td>Anomie71</td>
<td>3.60796</td>
<td>12.61581</td>
<td>0.7140</td>
</tr>
<tr>
<td>Education</td>
<td>2.99366</td>
<td>9.61000</td>
<td>0.6885</td>
</tr>
<tr>
<td>Powerless67</td>
<td>3.59488</td>
<td>9.35139</td>
<td>0.6156</td>
</tr>
<tr>
<td>Powerless71</td>
<td>3.59488</td>
<td>9.84533</td>
<td>0.6349</td>
</tr>
<tr>
<td>SEI</td>
<td>259.57639</td>
<td>450.28798</td>
<td>0.4235</td>
</tr>
<tr>
<td>Alien67</td>
<td>5.67046</td>
<td>8.29601</td>
<td>0.3165</td>
</tr>
<tr>
<td>Alien71</td>
<td>4.51479</td>
<td>9.00786</td>
<td>0.4988</td>
</tr>
</tbody>
</table>

For each endogenous variable in the model, the corresponding squared multiple correlation is computed by:

\[
1 - \frac{\text{error variance}}{\text{total variance}}
\]

In regression analysis, this is the percentage of explained variance of the endogenous variable by the predictors. However, this interpretation is complicated or even uninterpretable when your structural equation model
has correlated errors or reciprocal casual relations. In these situations, it is not uncommon to see negative R-squares. Negative R-squares do not necessarily mean that your model is wrong or the model prediction is weak. Rather, the R-square interpretation is questionable in these situations.

When your variables are measured on different scales, comparison of path coefficients cannot be made directly. For example, in Output 29.17.11, the path coefficient for path Education \( \rightleftharpoons \) SES is fixed at one, while the path coefficient for path SEI \( \rightleftharpoons \) SES is 5.369. It would be simple-minded to conclude that the effect of SES on SEI is greater than that SES on Education. Because SEI and Education are measured on different scales, direct comparison of the corresponding path coefficients is simply inappropriate.

In alleviating this problem, some might resort to the standardized solution for a better comparison. In a standardized solution, because the variances of manifest variables and systematic predictors are all standardized to ones, you hope the path coefficients are more comparable. In this example, PROC CALIS standardizes your results in Output 29.17.13.

**Output 29.17.13 Standardized Results**

| Path     | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|----------|-----------|----------|----------------|---------|-------|---|
| Anomie67 | Alien67   | 0.83481  | 0.01093        | 76.3531 | <.0001|
| Powerless67 | Alien67  | 0.78459  | 0.01163        | 67.4776 | <.0001|
| Anomie71 | Alien71   | 0.84499  | 0.01031        | 81.9796 | <.0001|
| Powerless71 | Alien71  | 0.79678  | 0.01107        | 71.9626 | <.0001|
| Education| SES       | 0.82975  | 0.03172        | 26.1599 | <.0001|
| SEI      | SES       | 0.65079  | 0.03019        | 21.5533 | <.0001|
| Alien67  | SES       | -0.56257 | 0.03456        | -16.2796| <.0001|
| Alien71  | SES       | -0.20642 | 0.04483        | -4.6043 | <.0001|
| Alien71  | Alien67   | 0.56920  | 0.04066        | 14.0000 | <.0001|

| Variance Type | Variable   | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|---------------|------------|-----------|----------|----------------|---------|-------|---|
| Error         | Anomie67   | theta1    | 0.30309  | 0.01825        | 16.6031 | <.0001|
|               | Powerless67| theta2    | 0.38442  | 0.01825        | 21.0695 | <.0001|
|               | Anomie71   | theta1    | 0.28599  | 0.01742        | 16.4178 | <.0001|
|               | Powerless71| theta2    | 0.36514  | 0.01764        | 20.6942 | <.0001|
|               | Education  | theta3    | 0.31152  | 0.05264        | 5.9182  | <.0001|
|               | SEI        | theta4    | 0.57647  | 0.03930        | 14.6680 | <.0001|
|               | Alien67    | psi1      | 0.68352  | 0.03888        | 17.5797 | <.0001|
|               | Alien71    | psi2      | 0.50121  | 0.03321        | 15.0897 | <.0001|
| Exogenous SES | phi        | 1.00000   |          |                |         |       |   |

| Error of Error of | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|-------------------|-----------|----------|----------------|---------|-------|---|
| Anomie67          | Anomie71  | theta5   | 0.07391        | 0.01013 | 7.2957 | <.0001|
| Powerless67       | Powerless71| theta5   | 0.09440        | 0.01274 | 7.4092 | <.0001|
Now, the standardized path coefficient for path Education $\Rightarrow$ SES is 0.830, while the standardized path coefficient for path SEI $\Rightarrow$ SES is 0.651. So the standardized effect of SES on SEI is actually smaller than that of SES on Education.

Furthermore, in PROC CALIS the standardized estimates are computed with standard error estimates and $t$ values so that you can make statistical inferences on the standardized estimates as well.

PROC CALIS might differ from other software in its standardization scheme. Unlike other software that might standardize the path coefficients that attach to the error terms (unsystematic sources), PROC CALIS keeps these path coefficients at ones (not shown in the output). Unlike other software that might also standardize the corresponding error variances to ones, the error variances in the standardized solution of PROC CALIS are rescaled so as to keep the mathematical consistency of the model.

Essentially, in PROC CALIS only variances of manifest and non-error-type latent variables are standardized to ones. The error variances are rescaled, but not standardized. For example, in the standardized solution shown in Output 29.17.13, the error variances for all endogenous variables are not ones (see the middle portion of the output). Only the variance for the latent variable SES is standardized to one. See the section “Standardized Solutions” on page 1500 for the logic of the standardization scheme adopted by PROC CALIS.

In appearance, the standardized solution is like a correlational analysis on the standardized manifest variables with standardized exogenous latent factors. Unfortunately, this statement is over-simplified, if not totally inappropriate. In standardizing a solution, the implicit equality constraints are likely destroyed. In this example, the unstandardized error variances for Anomie67 and Anomie71 are both 3.608, represented by a common parameter theta1. However, after standardization, these error variances have different values at 0.303 and 0.286, respectively. In addition, fixed parameter values are no longer fixed in a standardized solution (for example, the first five paths in the current example). The issue of standardization is common to all other SEM software and beyond the current discussion. PROC CALIS provides the standardized solution so that users can interpret the standardized estimates whenever they find them appropriate.

---

**Example 29.18: Simultaneous Equations with Mean Structures and Reciprocal Paths**

The supply-and-demand food example of Kmenta (1971, pp. 565, 582) is used to illustrate PROC CALIS for the estimation of intercepts and coefficients of simultaneous equations in econometrics. The model is specified by two simultaneous equations containing two endogenous variables $Q$ and $P$, and three exogenous variables $D$, $F$, and $Y$:

\[
Q_t(\text{demand}) = \alpha_1 + \beta_1 P_t + \gamma_1 D_t \\
Q_t(\text{supply}) = \alpha_2 + \beta_2 P_t + \gamma_2 F_t + \gamma_3 Y_t
\]

for $t = 1, \ldots, 20.$
To analyze this model in PROC CALIS, the second equation needs to be written in another form. For instance, in the LINEQS model each endogenous variable must appear on the left-hand side of exactly one equation. To satisfy this requirement, you can rewrite the second equation as an equation for $P_t$ as:

$$
P_t = -\frac{\alpha_2}{\beta_2} + \frac{1}{\beta_2} Q_t - \frac{\gamma_2}{\beta_2} F_t - \frac{\gamma_3}{\beta_2} Y_t
$$

or, equivalently reparameterized as:

$$
P_t = \theta_1 + \theta_2 Q_t + \theta_3 F_t + \theta_4 Y_t
$$

where

$$
\theta_1 = -\frac{\alpha_2}{\beta_2}, \quad \theta_2 = \frac{1}{\beta_2}, \quad \theta_3 = -\frac{\gamma_2}{\beta_2}, \quad \theta_4 = -\frac{\gamma_3}{\beta_2}
$$

This new equation for $P_t$ together with the first equation for $Q_t$ suggest the following LINEQS model specification in PROC CALIS:

```plaintext
title 'Food example of KMENTA(1971, p.565 & 582)';
data food;
  input Q P D F Y;
  label Q='Food Consumption per Head'
       P='Ratio of Food Prices to General Price'
       D='Disposable Income in Constant Prices'
       F='Ratio of Preceding Years Prices'
       Y='Time in Years 1922-1941';
datalines;
  98.485 100.323 87.4 98.0 1
  99.187 104.264 97.6 99.1 2
  102.163 103.435 96.7 99.1 3
  101.504 104.506 98.2 98.1 4
  104.240 98.001 99.8 110.8 5
  103.243 99.456 100.5 108.2 6
  103.993 101.066 103.2 105.6 7
  99.900 104.763 107.8 109.8 8
  100.350 96.446 96.6 108.7 9
  102.820 91.228 88.9 100.6 10
  95.435 93.085 75.1 81.0 11
  92.424 98.801 76.9 68.6 12
  94.535 102.908 84.6 70.9 13
  98.757 98.756 90.6 81.4 14
  105.797 95.119 103.1 102.3 15
  100.225 98.451 105.1 105.0 16
  103.522 86.498 96.4 110.5 17
  99.929 104.016 104.4 92.5 18
  105.223 105.769 110.7 89.3 19
  106.232 113.490 127.1 93.0 20
;
```
Chapter 29: The CALIS Procedure

proc calis data=food pshort nostand;
lineqs
  Q = alpha1 * Intercept + beta1 * P + gamma1 * D + E1,
  P = theta1 * Intercept + theta2 * Q + theta3 * F + theta4 * Y + E2;
variance
  E1-E2 = eps1-eps2;
cov
  E1-E2 = eps3;
bounds
  eps1-eps2 >= 0. ;
run;

The LINEQS modeling language is used in this example because its specification is similar to the original equations. In the LINEQS statement, you essentially input the two model equations for Q and P. Parameters for intercepts and regression coefficients are also specified in the equations. Note that Intercept in the two equations is treated as a special variable that contains ones for all observations. Intercept is not a variable in the data set, nor do you need to create such a variable in your data set. Hence, the variable Intercept does not represent the intercept parameter itself. Instead, the intercept parameters for the two equations are the coefficients attached to Intercept. In this example, the intercept parameters are alpha1 and theta1, respectively, in the two equations. As required, error terms E1 and E2 are added to complete the equation specification.

In the VARIANCE statement, you specify eps1 and eps2, respectively, for the variance parameters of the error terms. In the COV, you specify eps3 for the covariance parameter between the error terms. In the BOUNDS statement, you set lower bounds for the error variances so that estimates of eps1 and eps2 would be nonnegative.

In this example, the PSHORT and the NOSTAND options are used in the PROC CALIS statement. The PSHORT option suppresses a large amount of the output. For example, initial estimates are not printed and simple descriptive statistics and standard errors are not computed. The NOSTAND option suppresses the printing of the standardized results. Because the default printing in PROC CALIS might produce a large amount of output, using these printing options make your output more concise and readable. Whenever appropriate, you may consider using these printing options.

The estimated equations are shown in Output 29.18.1.

Output 29.18.1 Linear Equations

<table>
<thead>
<tr>
<th>Linear Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q = 93.6193 (<strong>) Intercept -0.2295 (</strong>) P + 0.3100 (**) D + 1.0000 E1</td>
</tr>
<tr>
<td>P = -218.9 (ns) Intercept + 4.2140 (<strong>) Q + -0.9305 (</strong>) F + -1.5579 (**) Y + 1.0000 E2</td>
</tr>
</tbody>
</table>

The estimates of intercepts and regression coefficients are shown directly in the equations. Any number in an equation followed by an asterisk is an estimate. For the estimates in equations, the parameter names are shown underneath the associated variables. Any number in an equation not followed by an asterisk is a fixed value. For example, the value 1.0000 attached to the error term in each of the output equation is fixed. Also, for fixed coefficients there are no parameter names underneath the associated variables.

All but the intercept estimates in the equation for predicting P are statistically significant at $\alpha = 0.05$ (when using an approximate critical value of 2). The t ratio for $\theta_{theta1}$ is $-1.590$, which implies that this intercept might have been zero in the population. However, because you have reparameterized the original model to use the LINEQS model specification, transformed parameters like $\theta_{theta1}$ in this model might not be of
primary interest. Therefore, you might not need to pay any attention to the significance of the theta1 estimate. There is a way to use the original econometric parameters to specify the LINEQS model. It is discussed in the later part of this example.

Estimates for variance, covariance, and mean parameters are shown in Output 29.18.2.

**Output 29.18.2** Variance, Covariance, and Mean Parameters

| Variable Type | Variable Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|---------------|--------------------|----------|----------------|---------|------|---|
| Error         | E1                  | 3.51274  | 1.20204        | 2.9223  | 0.0035|
|               | E2                  | 105.06749| 83.89446       | 1.2524  | 0.2104|
| Observed      | _Add1              | 139.96029| 45.40911       | 3.0822  | 0.0021|
| F             | _Add2              | 161.51355| 52.40192       | 3.0822  | 0.0021|
| Y             | _Add3              | 35.00000 | 11.35550       | 3.0822  | 0.0021|

| Var1 | Var2 | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|------|------|-----------|----------|----------------|---------|------|---|
| E1   | E2   | eps3      | -18.8727 | -2.1496        | 0.0316  |
| F    | D    | _Add4     | 74.02539 | 38.44699       | 1.9254  | 0.0542|
| Y    | D    | _Add5     | 22.99211 | 16.90102       | 1.3604  | 0.1737|
| Y    | F    | _Add6     | -21.58158| 17.94544       | -1.2026 | 0.2291|

| Variable Type | Variable Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|---------------|--------------------|----------|----------------|---------|------|---|
| Observed      | _Add7              | 97.53500 | 2.71410        | 35.9364 | <.0001|
| F             | _Add8              | 96.62500 | 2.91560        | 33.1407 | <.0001|
| Y             | _Add9              | 10.50000 | 1.35724        | 7.7363  | <.0001|

Parameters with a name prefix _Add are added automatically by PROC CALIS. These parameters are added as free parameters to complete the model specification. In PROC CALIS, variances and covariances among the set of exogenous manifest variables must be parameters. You either specify them explicitly or let the CALIS procedure to add them. If you need to constrain or to fix these parameters, then you must specify them explicitly. When your model also fits the mean structures, the same principle applies to the means of the exogenous manifest variables. In this example, because variables D, F, and Y are all exogenous manifest variables, their associated means, variances and covariances must be parameters in the model.

The squared multiple correlations for the equations are shown in Output 29.18.3.

**Output 29.18.3** Squared Multiple Correlations

<table>
<thead>
<tr>
<th>Variable</th>
<th>Error Variance</th>
<th>Total Variance</th>
<th>R-Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>3.51274</td>
<td>14.11128</td>
<td>0.7511</td>
</tr>
<tr>
<td>P</td>
<td>105.06749</td>
<td>35.11850</td>
<td>-1.9918</td>
</tr>
</tbody>
</table>
For endogenous variable \( P \), the R-square is \(-1.9918\), which is obviously an invalid value. In fact, because there are correlated errors (between \( E_1 \) and \( E_2 \)) and reciprocal paths (paths to and from \( Q \) and \( P \)), the model departs from the regular assumptions of multiple regression analysis. As a result, you should not interpret the R-squares for this example.

**Specifying the LINEQS with the Original Econometric Parameters**

If you are interested in estimating the parameters in the original econometric model (that is, \( \alpha_2, \beta_2, \gamma_2, \) and \( \gamma_3 \)), the previous reparameterized LINEQS model does not serve your purpose well enough. However, using the relations between these original parameters with the \( \theta \) parameters in the reparameterized LINEQS model, you can set up some “super-parameters” in the LINEQS model, as shown in the following statements:

```sas
proc calis data=Food pshort nostand;
  lineqs
    Q = alpha1 * Intercept + beta1 * P + gamma1 * D + E1,
    P = theta1 * Intercept + theta2 * Q + theta3 * F + theta4 * Y + E2;
  variance
    E1-E2 = eps1-eps2;
  cov
    E1-E2 = eps3;
  bounds
    eps1-eps2 >= 0.;
  parameters alpha2 (50.) beta2 gamma2 gamma3 (3*.25);
    theta1 = -alpha2 / beta2;
    theta2 = 1 / beta2;
    theta3 = -gamma2 / beta2;
    theta4 = -gamma3 / beta2;
run;
```

In this new specification, only the `PARAMETERS` statement and the SAS programming statements following it are new. In the `PARAMETERS` statement, you define super-parameters `alpha2`, `beta2`, `gamma2`, and `gamma3`, and put initial values for them in parentheses. These parameters are the original econometric parameters of interest. The SAS programming statements that follow the `PARAMETERS` statement are used to define the functional relationships of the super-parameters with the parameters in the LINEQS model. Consequently, in this new specification, \( \theta_1 \), \( \theta_2 \), \( \theta_3 \), and \( \theta_4 \) are no longer independent parameters in the model, as they are in the previous reparameterized model. Instead, \( \alpha_2 \), \( \beta_2 \), \( \gamma_2 \), and \( \gamma_3 \) are independent parameters in this new specification. By fitting this new model, you get the same set of estimates as those in the previous LINEQS model. In addition, you get estimates of the super-parameters, as shown in **Output 29.18.4**.

**Output 29.18.4 Additional Parameters**

| Type     | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|----------|-----------|----------|----------------|---------|------|---|
| Independent | alpha2  | 51.94452 | 11.70002 | 4.4397 | <.0001 |
|          | beta2   | 0.23731  | 0.09877 | 2.4026 | 0.0163 |
|          | gamma2  | 0.22082  | 0.04161 | 5.3070 | <.0001 |
|          | gamma3  | 0.36971  | 0.07060 | 5.2365 | <.0001 |
You can now interpret the results in terms of the original econometric parameterization. As shown in Output 29.18.4, all these estimates are significant, despite the fact that one of the transformed parameter estimates in the linear equations of the LINEQS model is not. You can obtain almost equivalent results by applying the SAS/ETS procedure SYSLIN on this problem.

**Alternative Ways to Specify Your LINEQS Model**

In specifying the linear equations in the LINEQS model, it might become cumbersome when you need to name a lot of parameters into the equations. If the parameters in your model are unconstrained, you need to very careful to use unique parameter names to distinguish the free parameters because parameters with the same name are identical and will have the same estimate. To make model specification easier and to avoid accidental constraints, PROC CALIS provides an efficient way to specify these free parameters. That is, you can simply omit the parameter names in the specification. For example, in the first specification of the current example, except for the boundary constraints on the error variance parameters, all other parameters in the model are not constrained, as shown in the following statements:

```sas
proc calis data=food pshort nostand;
  lineqs
    Q = alpha1 * Intercept + beta1 * P + gamma1 * D + E1,
    P = theta1 * Intercept + theta2 * Q + theta3 * F + theta4 * Y + E2;
  variance
    E1-E2 = eps1-eps2;
  cov
    E1-E2 = eps3;
  bounds
    eps1-eps2 >= 0. ;
run;
```

Parameters such as `alpha1`, `beta1`, and so on are unique parameter names in the specific locations of the model. They are free parameters. Hence, you can use the following equivalent specification:

```sas
proc calis data=food pshort nostand;
  lineqs
    Q = Intercept + * P + * D + * E1,
    P = Intercept + * Q + * F + * Y + E2;
  variance
    E1-E2 = eps1-eps2;
  cov
    E1 E2;
  bounds
    eps1-eps2 >= 0. ;
run;
```

Only the parameters `eps1` and `eps2` remain in this equivalent specification. You omit the specification of all other parameter names. But the estimation results are the same, as shown in Output 29.18.5.

**Output 29.18.5** Estimation Results With Generated Parameter Names

**Linear Equations**

- \( Q = 93.6193 \) Intercept \(-0.2295\) P \(+0.3100\) D \(+1.0000\) E1
- \( P = -218.9\) Intercept \(+4.2140\) Q \(-0.9305\) F \(-1.5579\) Y \(+1.0000\) E2
The estimation results in Output 29.18.5 are the same as those in Output 29.18.2 and Output 29.18.3 with the original LINEQS model specification, only now PROC CALIS generates the parameter names with the _Parm in the results, as shown in Output 29.18.5. Note that you retain the parameter names eps1 and eps2 because you need to refer to them in the BOUNDS statement.
Example 29.19: Fitting Direct Covariance Structures

In the section “Direct Covariance Structures Analysis” on page 1192, the MSTRUCT modeling language is used to specify a model with direct covariance structures. In the model, four variables from the data set of Wheaton et al. (1977) are used. The analysis is carried out in this example to investigate the tenability of the hypothesized covariance structures.

The four variables used are: Anomie67, Powerless67, Anomie71, and Powerless71. The hypothesized covariance matrix is structured as:

$$
\Sigma = \begin{pmatrix}
\phi_1 & \theta_1 & \theta_2 & \theta_1 \\
\theta_1 & \phi_2 & \theta_1 & \theta_3 \\
\theta_2 & \theta_1 & \phi_1 & \theta_1 \\
\theta_1 & \theta_3 & \theta_1 & \phi_2 \\
\end{pmatrix}
$$

where:

- $\phi_1$: variance of anomie
- $\phi_2$: variance of powerlessness
- $\theta_1$: covariance between anomie and powerlessness
- $\theta_2$: covariance between anomie measures
- $\theta_3$: covariance between powerlessness measures

In this example, you hypothesize the covariance structures directly, as opposed to those models with implied covariance structures from path models (see Example 29.17), structural equations (see Example 29.18), or other types of models. The basic assumption of the direct covariance structures in this example is that Anomie and Powerless were invariant over the measurement periods employed. This implies that the time of measurement did not change the variances and covariances of the measures. Therefore, both Anomie67 and Anomie71 have the same variance parameter $\phi_1$, and both Powerless67 and Powerless71 have the same variance parameter $\phi_2$. These two parameters, $\phi_1$ and $\phi_2$, are hypothesized on the diagonal of the covariance matrix $\Sigma$. In the same structured covariance matrix, $\theta_1$ represents the covariance between Anomie and Powerless, without regard to the time of measurement. The $\theta_2$ parameter represents the covariance between the Anomie measures, or the reliability of the Anomie measure. Similarly, the $\theta_2$ parameter represents the covariance between the Powerless measures, or the reliability of the Powerless measure.

As explained in the section “Direct Covariance Structures Analysis” on page 1192, you can use the MSTRUCT modeling language to specify the hypothesized covariance structures directly, as shown in the following statements:
proc calis nobs=932 data=Wheaton psummary;
   fitindex on(only)=\{chisq df probchi\} outfit=savefit;
   mstruct
   var = Anomie67 Powerless67 Anomie71 Powerless71;
   matrix _COV_ [1,1] = phi1,
   [2,2] = phi2,
   [3,3] = phi1,
   [4,4] = phi2,
   [2,1] = theta1,
   [3,1] = theta2,
   [3,2] = theta1,
   [4,1] = theta1,
   [4,2] = theta3,
   [4,3] = theta1;
run;

In the MSTRUCT statement you specify the variables in the VAR= list. The order of variables in this VAR= list is assumed to be the same as that in the row and column of the hypothesized covariance matrix. Next, in the MATRIX statement you specify parameters as entries in the hypothesized covariance matrix _COV_. Only the lower diagonal elements need to be specified because covariance matrices, by nature, are symmetric. Redundant specification of the upper triangular elements are unnecessary as PROC CALIS has the information accounted for. You can also set initial estimates by putting parenthesized numbers after the parameter names. But in this example you let PROC CALIS determine all the initial estimates.

In the PROC CALIS statement, the PSUMMARY option is used. As a global display option, this option suppresses a lot of displayed output and requests only the fit summary table be printed. This way you can eliminate quite a lot of displayed output that is not of your primary interest. In this example, the specification of the covariance structures is straightforward, and you do not need any output regarding the initial estimation or standardized solution. Suppose that you are not even concerned with the estimates of the parameters because you are not yet sure if this model is good enough for the data. All you want to know at this stage is whether the hypothesized covariance structures fit the data well. Therefore, the PSUMMARY option would serve your purpose well in this example.

In fact, even the fit summary table can be trimmed down quite a bit if you only want to look at certain specific fit indices. In the FITINDEX statement of this example, the ON(ONLY)= option turns on the printing of the model fit chi-square, its df, and p-value only. This does not mean that you must lose the information of all other fit indices. In addition to the printed output, you can save all fit indices in an output data set. To this end, you can use the OUTFIT= option in the FITINDEX statement. In this example, you save the results of all fit indices in a SAS data set called savefit.

Output 29.19.1 shows the entire printed output.

**Output 29.19.1** Testing Direct Covariance Structures

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>221.5798</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>5</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>&lt;.0001</td>
</tr>
</tbody>
</table>

The displayed output is very concise. It contains only a fit summary table with three statistics. The p-value for the model fit chi-square test indicates that the hypothesized structures should be rejected at $\alpha = 0.05$. Therefore, this rather restrictive direct covariance structure model does not fit the data well. A less restrictive covariance structure model is needed to explain the variances and covariances.
All fit indices are saved in the `savefit` data set. To view it, you can use the following statement:

```plaintext
proc print data=savefit;
run;
```

Output 29.19.2 shows all indices, their types and values of all fit indices and information.

### Output 29.19.2 Saved Fit Indices

**Analysis of Direct Covariance Structures**

**Testing Model by the MSTRUCT Language**

<table>
<thead>
<tr>
<th>Obs</th>
<th>TYPE</th>
<th>IndexCode</th>
<th>FitIndex</th>
<th>FitValue</th>
<th>PrintChar</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ModelInfo</td>
<td>101</td>
<td>Number of Observations</td>
<td>932.00</td>
<td>932</td>
</tr>
<tr>
<td>2</td>
<td>ModelInfo</td>
<td>103</td>
<td>Number of Variables</td>
<td>4.00</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>ModelInfo</td>
<td>104</td>
<td>Number of Moments</td>
<td>10.00</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>ModelInfo</td>
<td>105</td>
<td>Number of Parameters</td>
<td>5.00</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>ModelInfo</td>
<td>106</td>
<td>Number of Active Constraints</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>ModelInfo</td>
<td>111</td>
<td>Baseline Model Function Value</td>
<td>1.68</td>
<td>1.6799</td>
</tr>
<tr>
<td>7</td>
<td>ModelInfo</td>
<td>113</td>
<td>Baseline Model Chi-Square</td>
<td>1563.94</td>
<td>1563.9442</td>
</tr>
<tr>
<td>8</td>
<td>ModelInfo</td>
<td>114</td>
<td>Baseline Model Chi-Square DF</td>
<td>6.00</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>ModelInfo</td>
<td>115</td>
<td>Pr &gt; Baseline Model Chi-Square</td>
<td>0.00</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>10</td>
<td>Absolute</td>
<td>201</td>
<td>Fit Function</td>
<td>0.24</td>
<td>0.2380</td>
</tr>
<tr>
<td>11</td>
<td>Absolute</td>
<td>203</td>
<td>Chi-Square</td>
<td>221.58</td>
<td>221.5798</td>
</tr>
<tr>
<td>12</td>
<td>Absolute</td>
<td>204</td>
<td>Chi-Square DF</td>
<td>5.00</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>Absolute</td>
<td>205</td>
<td>Pr &gt; Chi-Square</td>
<td>0.00</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>14</td>
<td>Absolute</td>
<td>211</td>
<td>Z-Test of Wilson &amp; Hilferty</td>
<td>12.25</td>
<td>12.2533</td>
</tr>
<tr>
<td>15</td>
<td>Absolute</td>
<td>212</td>
<td>Hoelter Critical N</td>
<td>47.00</td>
<td>47</td>
</tr>
<tr>
<td>16</td>
<td>Absolute</td>
<td>213</td>
<td>Root Mean Square Residual (RMR)</td>
<td>0.76</td>
<td>0.7649</td>
</tr>
<tr>
<td>17</td>
<td>Absolute</td>
<td>214</td>
<td>Standardized RMR (SRMR)</td>
<td>0.07</td>
<td>0.0701</td>
</tr>
<tr>
<td>18</td>
<td>Absolute</td>
<td>215</td>
<td>Goodness of Fit Index (GFI)</td>
<td>0.90</td>
<td>0.9036</td>
</tr>
<tr>
<td>19</td>
<td>Parsimony</td>
<td>301</td>
<td>Adjusted GFI (AGFI)</td>
<td>0.81</td>
<td>0.8071</td>
</tr>
<tr>
<td>20</td>
<td>Parsimony</td>
<td>302</td>
<td>Parsimonious GFI</td>
<td>0.75</td>
<td>0.7530</td>
</tr>
<tr>
<td>21</td>
<td>Parsimony</td>
<td>303</td>
<td>RMSEA Estimate</td>
<td>0.22</td>
<td>0.2157</td>
</tr>
<tr>
<td>22</td>
<td>Parsimony</td>
<td>304</td>
<td>RMSEA Lower 90% Confidence Limit</td>
<td>0.19</td>
<td>0.1920</td>
</tr>
<tr>
<td>23</td>
<td>Parsimony</td>
<td>305</td>
<td>RMSEA Upper 90% Confidence Limit</td>
<td>0.24</td>
<td>0.2404</td>
</tr>
<tr>
<td>24</td>
<td>Parsimony</td>
<td>306</td>
<td>Probability of Close Fit</td>
<td>0.00</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>25</td>
<td>Parsimony</td>
<td>307</td>
<td>ECVI Estimate</td>
<td>0.25</td>
<td>0.2488</td>
</tr>
<tr>
<td>26</td>
<td>Parsimony</td>
<td>308</td>
<td>ECVI Lower 90% Confidence Limit</td>
<td>0.20</td>
<td>0.2003</td>
</tr>
<tr>
<td>27</td>
<td>Parsimony</td>
<td>309</td>
<td>ECVI Upper 90% Confidence Limit</td>
<td>0.31</td>
<td>0.3053</td>
</tr>
<tr>
<td>28</td>
<td>Parsimony</td>
<td>310</td>
<td>Akaike Information Criterion</td>
<td>231.58</td>
<td>231.5798</td>
</tr>
<tr>
<td>29</td>
<td>Parsimony</td>
<td>311</td>
<td>Bozdogan CAIC</td>
<td>260.77</td>
<td>260.7665</td>
</tr>
<tr>
<td>30</td>
<td>Parsimony</td>
<td>312</td>
<td>Schwarz Bayesian Criterion</td>
<td>255.77</td>
<td>255.7665</td>
</tr>
<tr>
<td>31</td>
<td>Parsimony</td>
<td>313</td>
<td>McDonald Centrality</td>
<td>0.89</td>
<td>0.8903</td>
</tr>
<tr>
<td>32</td>
<td>Incremental</td>
<td>401</td>
<td>Bentler Comparative Fit Index</td>
<td>0.86</td>
<td>0.8610</td>
</tr>
<tr>
<td>33</td>
<td>Incremental</td>
<td>402</td>
<td>Bentler-Bonett NFI</td>
<td>0.86</td>
<td>0.8583</td>
</tr>
<tr>
<td>34</td>
<td>Incremental</td>
<td>403</td>
<td>Bentler-Bonett Non-normed Index</td>
<td>0.83</td>
<td>0.8332</td>
</tr>
<tr>
<td>35</td>
<td>Incremental</td>
<td>404</td>
<td>Bollen Normed Index Rho1</td>
<td>0.83</td>
<td>0.8300</td>
</tr>
<tr>
<td>36</td>
<td>Incremental</td>
<td>405</td>
<td>Bollen Non-normed Index Delta2</td>
<td>0.86</td>
<td>0.8611</td>
</tr>
<tr>
<td>37</td>
<td>Incremental</td>
<td>406</td>
<td>James et al. Parsimonious NFI</td>
<td>0.72</td>
<td>0.7153</td>
</tr>
</tbody>
</table>
The results of various fit indices from this output data set confirm that the hypothesized model does not fit the data well.

As an aside, it is noted with some shorthand notation, the specification of the MSTRUCT model parameters that use the MATRIX statements can be made a little more precise for the current example. This is shown as follows:

```plaintext
proc calis nobs=932 data=Wheaton psummary;
  mstruct
    var = Anomie67 Powerless67 Anomie71 Powerless71;
    matrix _COV_ [1,1] = phi1 phi2 phi1 phi2,
                       [2, ] = thetal,
                       [3, ] = theta2 thetal,
                       [4, ] = thetal theta3 thetal;
    fitindex on(only)= [chisq df probchi] outfit=savefit;
run;
```

In the first entry of the MATRIX statement, the notation [1,1] represents that the parameter list specified after the equal sign starts with the [1,1] element of the _COV_ matrix and proceeds down the diagonal. In the next three entries, the notations [2,], [3,], and [4,] represent that parameter lists start with the first elements of the second, third, and fourth rows, respectively, and proceed to the next (right) elements on the same rows. See the syntax of the MATRIX statement on page 1295 for more details about this kind of shorthand notation.

This example shows how you can use the MSTRUCT modeling language to test specific covariance patterns. You need to define the parameters of the covariance patterns explicitly by the MATRIX statements. See Example 29.4 and Example 29.21 for more applications.

However, some commonly-used covariance and mean patterns are built into PROC CALIS. For these covariance and mean patterns, you can simply use the COVPATTERN= and the MEANPATTERN= options without the need to specify the parameters in the MATRIX statements. See the COVPATTERN= and the MEANPATTERN= options for the supported covariance and mean patterns. See Example 29.5 and Example 29.22 for applications.

---

**Example 29.20: Confirmatory Factor Analysis: Cognitive Abilities**

In this example, cognitive abilities of 64 students from a middle school were measured. The fictitious data contain nine cognitive test scores. Three of the scores were for reading skills, three others were for math skills, and the remaining three were for writing skills. The covariance matrix for the nine variables was obtained. A confirmatory factor analysis with three factors was conducted. The following is the input data set:
Confirmatory Factor Model with Uncorrelated Factors

You first fit a confirmatory factor model with uncorrelated factors to the data, as shown in the following statements:

```plaintext
proc calis data=cognitive1 nobus=64 modification;
   factor
      Read_Factor ===> reading1-reading3,
      Math_Factor ===> math1-math3,
      Write_Factor ===> writing1-writing3;
   pvar
      Read_Factor Math_Factor Write_Factor = 3 * 1.;
   cov
      Read_Factor Math_Factor Write_Factor = 3 * 0.;
run;
```

In the PROC CALIS statement, the number of observations is specified with the NOBS= option. With the MODIFICATION in the PROC CALIS statement, LM (Lagrange Multiplier) tests are conducted. The results of LM tests can suggest the inclusion of additional parameters for a better model fit.

The FACTOR modeling language is most handy when you specify confirmatory factor models. You use the FACTOR statement to invoke the FACTOR modeling language. Entries in the FACTOR statement are for specifying factor-variables relationships and are separated by commas. In each entry, you first specify a latent factor, followed by the right arrow sign ===>(you can use >, =>, ===>, or ===>). Then you specify the observed variables that have nonzero loadings on the factor. For example, in the first entry of FACTOR statement, you specify that latent factor Read_Factor has nonzero loadings (free parameters) on variables reading1–reading3. Optionally, you can specify the parameter list after you specify the factor-variable relationships. For example, you can name the loading parameters as in the following specification:

```plaintext
factor
   Read_Factor ===> reading1-reading3 = load1-load3;
```

This way, you name the factor loadings with parameter names load1, load2, and load3, respectively. However, in the current example, because the loading parameters are all unconstrained, you can just let PROC CALIS to generate the parameter names for you. In this example, there are three factors: Read_Factor, Math_Factor,
and Write_Factor. These factors have simple cluster structures with the nine observed variables. Each observed variable has only one loading on exactly one factor.

In the PVAR statement, you can specify the variances of the factors and the error variances of the observed variables. The factor variances in this model are all fixed at 1.0 for identification purposes. You do not need to specify the error variances of the observed variables in the current model because PROC CALIS assumes these are free parameters by default.

In the COV statement, you specify that the covariances among the factors are fixed zeros. There are three covariances among the three latent factors and therefore you put $3 \times 0$. for their fixed values. This means that the factors in the current model are uncorrelated. Note that you must specify uncorrelated factors explicitly in the COV statement because all latent factors are correlated by default.

In Output 29.20.1, the initial model specification is echoed in matrix form. The observed variables and factors are also displayed.

**Output 29.20.1 Uncorrelated Factor Model Specification**

<table>
<thead>
<tr>
<th>Variables in the Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variables</strong></td>
</tr>
<tr>
<td><strong>Factors</strong></td>
</tr>
</tbody>
</table>

| Number of Variables | = 9 |
| Number of Factors   | = 3 |

<table>
<thead>
<tr>
<th>Initial Factor Loading Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read_Factor        Math_Factor Write_Factor</td>
</tr>
<tr>
<td>reading1           .          0          0          [Parm1]</td>
</tr>
<tr>
<td>[Parm2]    0          0          [Parm3]</td>
</tr>
<tr>
<td>[Parm4]    0          0</td>
</tr>
<tr>
<td>math1              0          .          0          [Parm5]</td>
</tr>
<tr>
<td>[Parm6]    0          0</td>
</tr>
<tr>
<td>math2              0          .          0          [Parm7]</td>
</tr>
<tr>
<td>[Parm8]    0          0</td>
</tr>
<tr>
<td>math3              0          .          0          [Parm9]</td>
</tr>
<tr>
<td>[Parm10]   0          0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial Factor Covariance Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read_Factor        Math_Factor  Write_Factor</td>
</tr>
<tr>
<td>Read_Factor        1.00000       0          0</td>
</tr>
<tr>
<td>Math_Factor        0            1.00000    0</td>
</tr>
<tr>
<td>Write_Factor       0            0          1.00000</td>
</tr>
</tbody>
</table>
In the table for initial factor loading matrix, the nine loading parameters are shown to have simple cluster relations with the factors. In the table for initial factor covariance matrix, the diagonal matrix shows that the factors are not correlated. The diagonal elements are fixed at ones so that this matrix is also a correlation matrix for the factors. In the table for initial error variances, the nine variance parameters are shown. As described previously, these error variances are generated by PROC CALIS as default parameters.

In Output 29.20.2, initial estimates are generated by the instrumental variable method and the McDonald method.

**Output 29.20.1 continued**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Parameter</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>reading1</td>
<td>_Add1</td>
<td>.</td>
</tr>
<tr>
<td>reading2</td>
<td>_Add2</td>
<td>.</td>
</tr>
<tr>
<td>reading3</td>
<td>_Add3</td>
<td>.</td>
</tr>
<tr>
<td>math1</td>
<td>_Add4</td>
<td>.</td>
</tr>
<tr>
<td>math2</td>
<td>_Add5</td>
<td>.</td>
</tr>
<tr>
<td>math3</td>
<td>_Add6</td>
<td>.</td>
</tr>
<tr>
<td>writing1</td>
<td>_Add7</td>
<td>.</td>
</tr>
<tr>
<td>writing2</td>
<td>_Add8</td>
<td>.</td>
</tr>
<tr>
<td>writing3</td>
<td>_Add9</td>
<td>.</td>
</tr>
</tbody>
</table>

NOTE: Parameters with prefix '_Add' are added by PROC CALIS.

**Output 29.20.2 Optimization of the Uncorrelated Factor Model: Initial Estimates**

<table>
<thead>
<tr>
<th>Initial Estimation Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>
These initial estimates turn out to be pretty good, in the sense that only three more iterations are needed to converge to the maximum likelihood estimates and the final function value 0.784 does not change much from the initial function value 0.910, as shown in Output 29.20.3.

**Output 29.20.2 continued**

<table>
<thead>
<tr>
<th>Parameter Estimates</th>
<th>N</th>
<th>Parameter</th>
<th>Estimate</th>
<th>Gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 _Parm1</td>
<td>7</td>
<td>7.15372</td>
<td>0.00851</td>
<td></td>
</tr>
<tr>
<td>2 _Parm2</td>
<td>7</td>
<td>7.80225</td>
<td>-0.00170</td>
<td></td>
</tr>
<tr>
<td>3 _Parm3</td>
<td>8</td>
<td>8.70856</td>
<td>-0.00602</td>
<td></td>
</tr>
<tr>
<td>4 _Parm4</td>
<td>7</td>
<td>7.68637</td>
<td>0.00272</td>
<td></td>
</tr>
<tr>
<td>5 _Parm5</td>
<td>8</td>
<td>8.01765</td>
<td>-0.01096</td>
<td></td>
</tr>
<tr>
<td>6 _Parm6</td>
<td>7</td>
<td>7.05012</td>
<td>0.00932</td>
<td></td>
</tr>
<tr>
<td>7 _Parm7</td>
<td>8</td>
<td>8.76776</td>
<td>-0.009955</td>
<td></td>
</tr>
<tr>
<td>8 _Parm8</td>
<td>5</td>
<td>5.96161</td>
<td>-0.01335</td>
<td></td>
</tr>
<tr>
<td>9 _Parm9</td>
<td>7</td>
<td>7.23168</td>
<td>0.01665</td>
<td></td>
</tr>
<tr>
<td>10 _Add1</td>
<td>31</td>
<td>31.84831</td>
<td>-0.00179</td>
<td></td>
</tr>
<tr>
<td>11 _Add2</td>
<td>47</td>
<td>47.36790</td>
<td>0.0003461</td>
<td></td>
</tr>
<tr>
<td>12 _Add3</td>
<td>23</td>
<td>23.50199</td>
<td>0.00257</td>
<td></td>
</tr>
<tr>
<td>13 _Add4</td>
<td>23</td>
<td>23.13374</td>
<td>-0.0008384</td>
<td></td>
</tr>
<tr>
<td>14 _Add5</td>
<td>31</td>
<td>31.84224</td>
<td>0.00280</td>
<td></td>
</tr>
<tr>
<td>15 _Add6</td>
<td>38</td>
<td>38.92075</td>
<td>-0.00167</td>
<td></td>
</tr>
<tr>
<td>16 _Add7</td>
<td>13</td>
<td>13.86035</td>
<td>-0.00579</td>
<td></td>
</tr>
<tr>
<td>17 _Add8</td>
<td>61</td>
<td>61.00217</td>
<td>0.00115</td>
<td></td>
</tr>
<tr>
<td>18 _Add9</td>
<td>46</td>
<td>46.14784</td>
<td>-0.00300</td>
<td></td>
</tr>
</tbody>
</table>

**Value of Objective**

Function = 0.9103815918

---

**Output 29.20.3 Optimization of the Uncorrelated Factor Model: Iteration Summary**

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Restarts</th>
<th>Function Calls</th>
<th>Active Constraints</th>
<th>Objective Function</th>
<th>Objective Function Change</th>
<th>Max Abs Gradient Element</th>
<th>Lambda</th>
<th>Ratio Between Actual and Predicted Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0.78792</td>
<td>0.1225</td>
<td>0.00175</td>
<td>0</td>
<td>0.932</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0.78373</td>
<td>0.00419</td>
<td>0.00037</td>
<td>0</td>
<td>1.051</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0.78373</td>
<td>5.087E-7</td>
<td>3.715E-9</td>
<td>0</td>
<td>1.001</td>
</tr>
</tbody>
</table>

**Optimization Results**

<table>
<thead>
<tr>
<th>Iterations</th>
<th>3 Function Calls</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobian Calls</td>
<td>5 Active Constraints</td>
<td>0</td>
</tr>
<tr>
<td>Objective Function</td>
<td>0.783733415 Max Abs Gradient Element</td>
<td>3.7146571E-9</td>
</tr>
<tr>
<td>Lambda</td>
<td>0 Actual Over Pred Change</td>
<td>1.0006660673</td>
</tr>
<tr>
<td>Radius</td>
<td>0.0025042942</td>
<td></td>
</tr>
</tbody>
</table>

Convergence criterion (ABSGCONV=0.00001) satisfied.
The fit summary is shown in Output 29.20.4.

**Output 29.20.4** Fit of the Uncorrelated Factor Model

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modeling Info</strong></td>
<td>Number of Observations</td>
</tr>
<tr>
<td></td>
<td>Number of Variables</td>
</tr>
<tr>
<td></td>
<td>Number of Moments</td>
</tr>
<tr>
<td></td>
<td>Number of Parameters</td>
</tr>
<tr>
<td></td>
<td>Number of Active Constraints</td>
</tr>
<tr>
<td></td>
<td>Baseline Model Function Value</td>
</tr>
<tr>
<td></td>
<td>Baseline Model Chi-Square</td>
</tr>
<tr>
<td></td>
<td>Baseline Model Chi-Square DF</td>
</tr>
<tr>
<td></td>
<td>Pr &gt; Baseline Model Chi-Square</td>
</tr>
<tr>
<td><strong>Absolute Index</strong></td>
<td>Fit Function</td>
</tr>
<tr>
<td></td>
<td>Chi-Square</td>
</tr>
<tr>
<td></td>
<td>Chi-Square DF</td>
</tr>
<tr>
<td></td>
<td>Pr &gt; Chi-Square</td>
</tr>
<tr>
<td></td>
<td>Z-Test of Wilson &amp; Hilferty</td>
</tr>
<tr>
<td></td>
<td>Hoelter Critical N</td>
</tr>
<tr>
<td></td>
<td>Root Mean Square Residual (RMR)</td>
</tr>
<tr>
<td></td>
<td>Standardized RMR (SRMR)</td>
</tr>
<tr>
<td></td>
<td>Goodness of Fit Index (GFI)</td>
</tr>
<tr>
<td><strong>Parsimony Index</strong></td>
<td>Adjusted GFI (AGFI)</td>
</tr>
<tr>
<td></td>
<td>Parsimonious GFI</td>
</tr>
<tr>
<td></td>
<td>RMSEA Estimate</td>
</tr>
<tr>
<td></td>
<td>RMSEA Lower 90% Confidence Limit</td>
</tr>
<tr>
<td></td>
<td>RMSEA Upper 90% Confidence Limit</td>
</tr>
<tr>
<td></td>
<td>Probability of Close Fit</td>
</tr>
<tr>
<td></td>
<td>ECVI Estimate</td>
</tr>
<tr>
<td></td>
<td>ECVI Lower 90% Confidence Limit</td>
</tr>
<tr>
<td></td>
<td>ECVI Upper 90% Confidence Limit</td>
</tr>
<tr>
<td></td>
<td>Akaike Information Criterion</td>
</tr>
<tr>
<td></td>
<td>Bozdogan CAIC</td>
</tr>
<tr>
<td></td>
<td>Schwarz Bayesian Criterion</td>
</tr>
<tr>
<td></td>
<td>McDonald Centrality</td>
</tr>
<tr>
<td><strong>Incremental Index</strong></td>
<td>Bentler Comparative Fit Index</td>
</tr>
<tr>
<td></td>
<td>Bentler-Bonett NFI</td>
</tr>
<tr>
<td></td>
<td>Bentler-Bonett Non-normed Index</td>
</tr>
<tr>
<td></td>
<td>Bollen Normed Index Rho1</td>
</tr>
<tr>
<td></td>
<td>Bollen Non-normed Index Delta2</td>
</tr>
<tr>
<td></td>
<td>James et al. Parsimonious NFI</td>
</tr>
</tbody>
</table>

Using the chi-square model test criterion, the uncorrelated factor model should be rejected at $\alpha = 0.05$. The RMSEA estimate is 0.1147, which is not indicative of a good fit according to Browne and Cudeck (1993). Other indices might suggest only a marginal good fit. For example, Bentler’s comparative fit index and Bollen nonnormed index delta2 are both above 0.90. However, many other do not attain this 0.90 level. For example, adjusted GFI is only 0.759. It is thus safe to conclude that there could be some improvements on the model fit.
The MODIFICATION option in the PROC CALIS statement has been used to request for computing the LM test indices for model modifications. The results are shown in Output 29.20.5.

### Output 29.20.5 Lagrange Multiplier Tests

<table>
<thead>
<tr>
<th>Variable</th>
<th>Factor</th>
<th>LM Stat</th>
<th>Pr &gt; ChiSq</th>
<th>Parm Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>writing1</td>
<td>Read_Factor</td>
<td>9.76596</td>
<td>0.0018</td>
<td>2.95010</td>
</tr>
<tr>
<td>math3</td>
<td>Write_Factor</td>
<td>3.58077</td>
<td>0.0585</td>
<td>1.89703</td>
</tr>
<tr>
<td>math1</td>
<td>Read_Factor</td>
<td>2.15312</td>
<td>0.1423</td>
<td>1.17976</td>
</tr>
<tr>
<td>writing3</td>
<td>Math_Factor</td>
<td>1.87637</td>
<td>0.1707</td>
<td>1.41298</td>
</tr>
<tr>
<td>math3</td>
<td>Read_Factor</td>
<td>1.02954</td>
<td>0.3103</td>
<td>0.95427</td>
</tr>
<tr>
<td>reading2</td>
<td>Write_Factor</td>
<td>0.91230</td>
<td>0.3395</td>
<td>0.99933</td>
</tr>
<tr>
<td>writing2</td>
<td>Math_Factor</td>
<td>0.86221</td>
<td>0.3531</td>
<td>0.95672</td>
</tr>
<tr>
<td>reading1</td>
<td>Write_Factor</td>
<td>0.63403</td>
<td>0.4259</td>
<td>0.73916</td>
</tr>
<tr>
<td>math1</td>
<td>Write_Factor</td>
<td>0.55562</td>
<td>0.4559</td>
<td>0.63906</td>
</tr>
<tr>
<td>reading2</td>
<td>Math_Factor</td>
<td>0.55362</td>
<td>0.4568</td>
<td>0.74628</td>
</tr>
</tbody>
</table>

### Rank Order of the 3 Largest LM Stat for Covariances of Factors

<table>
<thead>
<tr>
<th>Var1</th>
<th>Var2</th>
<th>LM Stat</th>
<th>Pr &gt; ChiSq</th>
<th>Parm Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write_Factor</td>
<td>Read_Factor</td>
<td>8.95268</td>
<td>0.0028</td>
<td>0.44165</td>
</tr>
<tr>
<td>Write_Factor</td>
<td>Math_Factor</td>
<td>7.07904</td>
<td>0.0078</td>
<td>0.40132</td>
</tr>
<tr>
<td>Math_Factor</td>
<td>Read_Factor</td>
<td>4.61896</td>
<td>0.0316</td>
<td>0.30411</td>
</tr>
</tbody>
</table>

### Rank Order of the 10 Largest LM Stat for Error Variances and Covariances

<table>
<thead>
<tr>
<th>Error of</th>
<th>Error of</th>
<th>LM Stat</th>
<th>Pr &gt; ChiSq</th>
<th>Parm Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>writing1</td>
<td>math2</td>
<td>5.45986</td>
<td>0.0195</td>
<td>-13.16822</td>
</tr>
<tr>
<td>writing1</td>
<td>math1</td>
<td>5.05573</td>
<td>0.0245</td>
<td>12.32431</td>
</tr>
<tr>
<td>writing3</td>
<td>math3</td>
<td>3.93014</td>
<td>0.0474</td>
<td>13.59149</td>
</tr>
<tr>
<td>writing3</td>
<td>math1</td>
<td>2.83209</td>
<td>0.0924</td>
<td>-9.86342</td>
</tr>
<tr>
<td>writing2</td>
<td>reading1</td>
<td>2.56677</td>
<td>0.1091</td>
<td>10.15901</td>
</tr>
<tr>
<td>writing2</td>
<td>math2</td>
<td>1.94879</td>
<td>0.1627</td>
<td>8.40273</td>
</tr>
<tr>
<td>writing2</td>
<td>reading3</td>
<td>1.75181</td>
<td>0.1856</td>
<td>-7.82777</td>
</tr>
<tr>
<td>writing3</td>
<td>reading1</td>
<td>1.57978</td>
<td>0.2088</td>
<td>-7.97915</td>
</tr>
<tr>
<td>writing1</td>
<td>reading2</td>
<td>1.34894</td>
<td>0.2455</td>
<td>7.77158</td>
</tr>
<tr>
<td>writing2</td>
<td>math3</td>
<td>1.11704</td>
<td>0.2906</td>
<td>-7.23762</td>
</tr>
</tbody>
</table>
Three different tables for ranking the LM test results are shown. In the first table, the new loading parameters that would improve the model fit the most are shown first. For example, in the first row a new factor loading of writing1 on the Read_Factor is suggested to improve the model fit the most. The ‘LM Stat’ value is 9.77. This is an approximation of the chi-square drop if this parameter was included in the model. The ‘Pr > ChiSq’ value of 0.0018 indicates a significant improvement of model fit at $\alpha = 0.05$. Nine more new loading parameters are suggested in the table, with less and less statistical significance in the change of model fit chi-square. Note that these approximate chi-squares are one-at-a-time chi-square changes. That means that the overall chi-square drop is not a simple sum of individual chi-square changes when you include two or more new parameters in the modified model.

The other two tables in Output 29.20.5 shows the new parameters in factor covariances, error variances, or error covariances that would result in a better model fit. The table for the new parameters of the factor covariance matrix indicates that adding each of the covariances among factors might lead to a statistically significant improvement in model fit. The largest ‘LM Stat’ value in this table is 8.95, which is smaller than that of the largest ‘LM Stat’ for the factor loading parameters. Despite this, it is more reasonable to add the covariance parameters among factors first to determine whether that improves the model fit.

**Confirmatory Factor Model with Correlated Factors**

To fit the corresponding confirmatory factor model with correlated factors, you can remove the fixed zeros from the COV statement in the preceding specification, as shown in the following statements:

```sas
proc calis data=cognitive1 nobs=64 modification;
  factor
    Read_Factor ===> reading1-reading3 ,
    Math_Factor ===> math1-math3 ,
    Write_Factor ===> writing1-writing3 ;
  pvar
    Read_Factor Math_Factor Write_Factor = 3 * 1.;
  cov
    Read_Factor Math_Factor Write_Factor /* = 3 * 0. */;
run;
```

In the COV statement, you comment out the fixed zeros so that the covariances among the latent factors are now free parameters. An alternative way is to delete the entire COV statement so that the covariances among factors are free parameters by the FACTOR model default.

The fit summary of the correlated factor model is shown in Output 29.20.6.
The model fit chi-square value is 29.47, which is about 20 less than the model with uncorrelated factors. The $p$-value is 0.20, indicating a satisfactory model fit. The RMSEA value is 0.06, which is close to 0.05, a value recommended as an indication of good model fit by Browne and Cudeck (1993) More fit indices that do not attain the 0.9 level with the uncorrelated factor model now have values close to or above 0.9. These include the goodness-of-fit index (GFI), McDonald centrality, Bentler-Bonnet NFI, and Bentler-Bonnet nonnormed index. By all counts, the correlated factor model is a much better fit than the uncorrelated factor model.

In Output 29.20.7, the estimation results for factor loadings are shown. All these loadings are statistically significant, indicating non-chance relationships with the factors.
Output 29.20.7  Estimation of the Factor Loading Matrix

<table>
<thead>
<tr>
<th></th>
<th>Factor Loading Matrix:</th>
<th>Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read_Factor</td>
<td>Math_Factor</td>
</tr>
<tr>
<td>reading1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reading2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>reading3</td>
<td></td>
</tr>
<tr>
<td>math1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>math2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>math3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>writing1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>writing2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>writing3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In Output 29.20.8, the factor covariance matrix is shown. Because the diagonal elements are all ones, the off-diagonal elements are correlations among factors. The correlations range from 0.30–0.5. These factors are moderately correlated.
Output 29.20.8  Estimation of the Correlations of Factors

<table>
<thead>
<tr>
<th></th>
<th>Read_Factor</th>
<th>Math_Factor</th>
<th>Write_Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read_Factor</td>
<td>1.0000</td>
<td>0.3272</td>
<td>0.4810</td>
</tr>
<tr>
<td></td>
<td>0.1311</td>
<td>0.1208</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.4955</td>
<td>3.9813</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0126</td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Parm10]</td>
<td>[Parm11]</td>
<td></td>
</tr>
<tr>
<td>Math_Factor</td>
<td>0.3272</td>
<td>1.0000</td>
<td>0.3992</td>
</tr>
<tr>
<td></td>
<td>0.1311</td>
<td>0.1313</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.4955</td>
<td>3.0417</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0126</td>
<td>0.002352</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Parm10]</td>
<td>[Parm12]</td>
<td></td>
</tr>
<tr>
<td>Write_Factor</td>
<td>0.4810</td>
<td>0.3992</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>0.1208</td>
<td>0.1313</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.9813</td>
<td>3.0417</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
<td>0.002352</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Parm11]</td>
<td>[Parm12]</td>
<td></td>
</tr>
</tbody>
</table>

In Output 29.20.9, the error variances for variables are shown.

Output 29.20.9  Estimation of the Error Variances

| Variable  | Parameter | Estimate | Std Error | t Value | Pr > |t| |
|-----------|-----------|----------|-----------|---------|------|---|
| reading1  | _Add1     | 37.24939 | 8.33997   | 4.4664  | <.0001 |    |
| reading2  | _Add2     | 46.49695 | 10.69869  | 4.3460  | <.0001 |    |
| reading3  | _Add3     | 15.90447 | 9.26097   | 1.7174  | 0.0859 |    |
| math1     | _Add4     | 25.22889 | 7.72269   | 3.2669  | 0.0011 |    |
| math2     | _Add5     | 24.89032 | 8.98327   | 2.7707  | 0.0056 |    |
| math3     | _Add6     | 42.12110 | 9.20362   | 4.5766  | <.0001 |    |
| writing1  | _Add7     | 27.24965 | 10.36489  | 2.6290  | 0.0086 |    |
| writing2  | _Add8     | 49.28881 | 11.39812  | 4.3243  | <.0001 |    |
| writing3  | _Add9     | 48.10684 | 11.48868  | 4.1873  | <.0001 |    |

All $t$ values except the one for reading3 are greater than 2, a value close to a critical $t$ value at $\alpha = 0.05$. This means that the error variance for reading3 could have been zero in the population, or it could have been nonzero but the current sample just has this insignificant value by chance (that is, a Type 2 error). Further research is needed to confirm either way.

In addition to the parameter estimation results, PROC CALIS also outputs supplementary results that could be useful for interpretations. In Output 29.20.10, the squared multiple correlations and the factor scores regression coefficients are shown.
The percentages of variance for the observed variables that can be explained by the factors are shown in the ‘R-Square’ column of the table for squared multiple correlations (R-squares). These R-squares can be interpreted meaningfully because there is no reciprocal relationships among variables or correlated errors in the model. All estimates of R-squares are bounded between 0 and 1.

In the table for factor scores regression coefficients, entries are coefficients for the variables you can use to create the factor scores. The larger the coefficient, the more influence of the corresponding variable for creating the factor scores. It makes intuitive sense to see the cluster pattern of these coefficients—the reading measures are more important to create the latent variable scores of Read_Factor and so on.
Chapter 29: The CALIS Procedure

Example 29.21: Testing Equality of Two Covariance Matrices Using a Multiple-Group Analysis

You can use PROC CALIS to do multiple-group or multiple-sample analysis. The groups in the analysis must be independent. In this example, a relatively simple multiple-group analysis is carried out. The covariance matrices of two independent groups are tested for equality. Hence, individual covariance matrices are actually not structured. Rather, they are constrained to be the same under the null hypothesis. That is, you want to test the following null hypothesis:

\[ H_0 : \Sigma_1 = \Sigma_2 \]

where \( \Sigma_1 \) and \( \Sigma_2 \) represent the population covariance matrices of the two independent groups in question.

In PROC CALIS, you can use two different approaches to test the equality of covariance matrices. The first approach is to define an MSTRUCT model explicitly and to fit this model to the independent groups. The second approach is to use the COVPATTERN= option to invoke the required covariance structure model for the independent groups. Some standard covariance structures or patterns with the MSTRUCT modeling language are built into PROC CALIS internally. With appropriate keywords for the COVPATTERN= option, you can invoke the target built-in covariance patterns without defining the MSTRUCT model explicitly. This example considers these two approaches successively.

This example is concerned with a reaction time experiment that was conducted on two groups of individuals. One group \((N = 20)\) was considered to be an expert group with prior training related to the tasks of the experiment. Another group \((N = 18)\) was a control group without prior training. Three tasks of dexterity were administered to all individuals. These tasks differed by their required complexity levels of body skills. They were labeled as high, medium, and low complexities.

Apparently, the differential performance of the two groups under different task complexities was the primary research objective. In this example, however, you are interested in testing whether the groups have the same covariance matrix for the tasks. Equality of covariance matrices might be an essential assumption in some statistical tests for comparing group means. In this example, the sample covariance matrices for the two groups are stored in the data sets Expert and Novice, as shown in the following:

```plaintext
data expert(type=cov);
  input _type_ $ _name_ $ high medium low;
datalines;
COV high 5.88 . .
COV medium 2.88 7.16 .
COV low 3.12 4.44 8.14
;

data novice(type=cov);
  input _type_ $ _name_ $ high medium low;
datalines;
COV high 6.42 . .
COV medium 1.24 8.25 .
COV low 4.26 2.75 7.99
;
```

These data sets are read into the analysis through the GROUP statements in the following PROC CALIS specification:
Example 29.21: Testing Equality of Two Covariance Matrices Using a Multiple-Group Analysis

```sas
proc calis;
  group 1 / data=expert nobs=20 label="Expert";
  group 2 / data=novice nobs=18 label="Novice";
  model 1 / groups=1,2;
    mstruct
      var=high medium low;
    fitindex NoIndexType On(only)=[chisq df probchi]
      chicorrect=eqcovmat;
    ods select ModelingInfo MSTRUCTVariables MSTRUCTCovInit Fit;
run;
```

The first `GROUP` statement defines `group 1` for the expert group. The second `GROUP` statement defines `group 2` for the novice group. You use the `NOBS=` option in both statements to provide the number of observations of these groups. You use the `LABEL=` option in these statements to provide meaningful group labels.

The `MODEL` statement defines `MODEL 1`. In the analysis, this model fits to both groups 1 and 2, as indicated by the `GROUPS=` option of the statement. This is done to test the null hypothesis of equality of covariance matrices in the two groups. An `MSTRUCT` model for `MODEL 1` is defined immediately afterward. Three variables, `high`, `medium`, and `low`, are specified in the `VAR=` option of the `MSTRUCT` statement.

Without further specification about the `MSTRUCT` model, `PROC CALIS` assumes all non-redundant elements in the covariance matrix are free parameters. This is what is required under the null hypothesis of the equality of covariance matrices in the two groups—the groups have the same covariance matrix, but the covariance matrix itself is unconstrained. Your model under the null hypothesis is now well-defined and ready to run. In addition, you use `FITINDEX` and `ODS SELECT` statements to customize or fine tune the analysis.

By using the options in the `FITINDEX` statement, you can customize the fit summary table and control some analytic options. In the current example, you use the `NOINDEXTYPE` option to suppress the printing of the index types in the fit summary table. Then, you use the `ON(ONLY)=` option to specify the fit indices printed in the fit summary table. In this example, you request only the model fit chi-square statistic, degrees of freedom, and the probability value of the chi-square be printed. Finally, you use the `CHICORRECT=EQCOVMAT` option to request a chi-square correction for the test of equality of covariance matrices. This correction is due to Box (1949) and is implemented in `PROC CALIS` as a built-in chi-square correction option.

In addition, because you are not interested in all displayed output for the current hypothesized model, you use the `ODS SELECT` statement to display only those output (or `ODS` tables) of interest. In this example, you request only the modeling information, the variables involved, the initial covariance matrix specification, and the fit summary table be printed. All output in `PROC CALIS` are named as an `ODS` table. To locate a particular output in `PROC CALIS`, you must know the corresponding `ODS` table name. See the section “`ODS Table Names`” on page 1523 for a listing of `ODS` tables produced by `PROC CALIS`.

Output 29.21.1 displays some information regarding the basic model setup.

### Output 29.21.1 Modeling Information and Initial Specification

<table>
<thead>
<tr>
<th>Group</th>
<th>Label</th>
<th>Data Set</th>
<th>N Obs</th>
<th>Model</th>
<th>Type</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Expert</td>
<td>WORK.EXPERT</td>
<td>20</td>
<td>Model 1</td>
<td>MSTRUCT Covariances</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Novice</td>
<td>WORK.NOVICE</td>
<td>18</td>
<td>Model 1</td>
<td>MSTRUCT Covariances</td>
<td></td>
</tr>
</tbody>
</table>
Output 29.21.1 continued

<table>
<thead>
<tr>
<th>Model 1.</th>
<th>Variables in the Model</th>
<th>high</th>
<th>medium</th>
<th>low</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Variables = 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Model 1. Initial MSTRUCT _COV_ Matrix

<table>
<thead>
<tr>
<th>high</th>
<th>medium</th>
<th>low</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Add1]</td>
<td>[Add2]</td>
</tr>
<tr>
<td>medium</td>
<td>[Add2]</td>
<td>[Add3]</td>
</tr>
<tr>
<td>low</td>
<td>[Add4]</td>
<td>[Add5]</td>
</tr>
</tbody>
</table>

The modeling information table summarizes some basic information about the two groups. Both of them are fitted by Model 1. The next table shows the variables involved: high, medium, and low. The order of variables in this table is the same as that of the row and column variables of the covariance model matrix, which is shown next in Output 29.21.1. The parameters for the entries in the covariance matrix are shown. The names of parameters are displayed in parentheses. All these parameters are set by default and their names have the prefix _Add. No initial estimates are given as input, as indicated by the missing value '.'.

Output 29.21.2 shows the customized fit summary table, which has been much simplified for the current example due to the uses of some options in the FITINDEX statement.

Output 29.21.2 Model Fit

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square 2.4924</td>
</tr>
<tr>
<td>Chi-Square DF 6</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square 0.8693</td>
</tr>
</tbody>
</table>

As shown in Output 29.21.2, the chi-square test statistic is 2.4924. With six degrees of freedom, the test statistic is not significant at $\alpha = 0.01$. Therefore, the hypothesized model is supported, which means that the equality of the covariance matrices of the groups is supported.
Instead of using the MSTRUCT modeling language explicitly for defining the hypothesized covariance patterns (or structures), you can also invoke the same covariance patterns by using the COVPATTERN= option, as shown in the following statements:

```plaintext
proc calis covpattern=eqcovmat;
   var high medium low;
   group 1 / data=expert nob=20 label="Expert";
   group 2 / data=novice nob=18 label="Novice";
   fitindex NoIndexType On(only)=\[chisq df probchi\];
run;
```

The COVPATTERN=EQCOVMAT option in the PROC CALIS statement hypothesizes that the two population covariance matrices for the groups are the same. Next, you specify the set of variables in the covariance matrices in the VAR statement, followed by the specification of the data for the two groups. You use the FITINDEX statement to select a subset of fit indices to display in the output.

Output 29.21.3 shows the data sets and the corresponding MSTRUCT models that are generated by the COVPATTERN=EQCOVMAT option.

**Output 29.21.3** Modeling Information with the COVPATTERN=EQCOVMAT Option

<table>
<thead>
<tr>
<th>Group</th>
<th>Label</th>
<th>Data Set</th>
<th>N Obs</th>
<th>Model</th>
<th>Type</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Expert</td>
<td>WORK.EXPERT</td>
<td>20</td>
<td>Model 1</td>
<td>MSTRUCT Covariances</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Novice</td>
<td>WORK.NOVICE</td>
<td>18</td>
<td>Model 2</td>
<td>MSTRUCT Covariances</td>
<td></td>
</tr>
</tbody>
</table>

PROC CALIS generates Model 1 for the expert group and Model 2 for the novice group. Output 29.21.4 and Output 29.21.5 show the covariance matrices of these two models.

**Output 29.21.4** Initial Specification of Model 1 for the Expert Group

```
Model 1.
Variables in the Model
high medium low
Number of Variables = 3

Model 1. Initial MSTRUCT _COV_ Matrix

    high  medium  low
  [cov_1_1] [cov_2_1] [cov_3_1]
  [cov_2_1] [cov_2_2] [cov_3_2]
  [cov_3_1] [cov_3_2] [cov_3_3]
```

**Output 29.21.5** Initial Specification of Model 2 for the Novice Group

```
Model 2.
Variables in the Model
high medium low
Number of Variables = 3
```
In Output 29.21.4, the covariance matrix for the expert group has three variables: high, medium, and low. The second table of Output 29.21.4 shows the parameters for the corresponding covariance matrix. PROC CALIS generates the parameter names for the elements in this covariance matrix: _cov_1_1, _cov_2_1, ..., _cov_3_3. In Output 29.21.5, the covariance matrix for the novice group has exactly the same set of three variables: high, medium, and low. The second table of Output 29.21.5 shows the parameters for the corresponding covariance matrix. These variance and covariance parameters are exactly the same as those in Output 29.21.4, as required by the testing of equality of covariance matrices.

Output 29.21.6 shows the fit summary of the test. The test results are exactly the same as those in Output 29.21.2, as expected. The chi-square value is 2.4924. With six degrees of freedom, the test statistic is not significant at $\alpha = 0.01$. The hypothesis about the equality of the covariance matrices between the groups is supported.

### Output 29.21.6

**Model Fit with the COVPATTERN=EQCOVMAT Option**

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>2.4924</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>6</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.8693</td>
</tr>
</tbody>
</table>

One advantage of using the built-in covariance patterns such as the current COVPATTERN=EQCOVMAT option is that it is more efficient and less error-prone than if you specify the covariance patterns manually by using the MSTRUCT and MATRIX statements. With the COVPATTERN= option, PROC CALIS generates the correct model specification internally. Another advantage is that when applicable, PROC CALIS applies the appropriate chi-square correction to the chi-square test statistic. For the current example, PROC CALIS displays the following message in the output:

**NOTE:** The chi-square correction due to Box for testing equality of covariance matrices was applied. Use the CHICORRECT=0 option if this correction is not desirable.

This shows that when you use the COVPATTERN=EQCOVMAT option, an appropriate chi-square correction is applied automatically to the chi-square test statistic. To turn off this automatic chi-square, you can use the CHICORRECT=0 in the PROC CALIS statement (although this should be a rare practice with the COVPATTERN= options).

To extend the test of the equality of covariance matrices to the test of the equality of mean vectors, see Example 29.4. To extend the multiple-group analysis of covariance patterns to the multiple-group analysis of a general structural equation model, see Example 29.28.
Example 29.22: Testing Equality of Covariance and Mean Matrices between Independent Groups

To make the specification of some standard MSTRUCT models for covariance and mean patterns more efficient, PROC CALIS defines these standard models internally. You can use two options to invoke these built-in covariance and mean patterns easily. For example, with the COVPATTERN= option, you can define the compound symmetry (COMPSYM) pattern for the covariance matrix or the equality of covariance matrices between groups (EQCOVMAT). With the MEANPATTERN= option, you can define uniform means (UNIFORM) for the mean vector or the equality of mean vectors between groups (EQMEANVEC). See the COVPATTERN= and the MEANPATTERN= options for details about the supported built-in covariance and mean patterns.

In Example 29.21, you test the equality of covariance matrices between two groups. This example extends the application to the test of equality of mean vectors between three independent groups by using the COVPATTERN= and MEANPATTERN= options together. The “best” fit model for the data is explored. The following DATA steps define the covariance and mean matrices for the three independent groups, respectively:

```plaintext
data g1(type=corr);
  Input _type_ $ 1-8 _name_ $ 9-11 x1-x9;
  datalines;
   corr  x1  1. . . . . . . . .
   corr  x2  .721 1. . . . . . . .
   corr  x3  .676 .379 1. . . . . . .
   corr  x4  .149 .403 .450 1. . . . .
   corr  x5  .422 .384 .445 .411 1. . . .
   corr  x6  .343 .456 .243 .308 .531 1. . . .
   corr  x7  .115 .225 .201 .481 .373 .198 1. . .
   corr  x8  .213 .237 .434 .503 .267 .333 .355 1. .
   corr  x9  .236 .257 .159 .246 .126 .235 .601 .512 1.
  mean   .21.3 22.3 17.2 23.4 22.1 15.6 18.7 20.1 19.7
  std    . 1.2 1.4 .87 1.33 2.2 1.4 2.3 2.1 1.8
  n     . 21 21 21 21 21 21 21 21 21
;

data g2(type=corr);
  Input _type_ $ 1-8 _name_ $ 9-11 x1-x9;
  datalines;
   corr  x1  1. . . . . . . . .
   corr  x2  .733 1. . . . . . . .
   corr  x3  .576 .388 1. . . . . . .
   corr  x4  .209 .414 .425 1. . . . .
   corr  x5  .412 .286 .461 .398 1. . . .
   corr  x6  .323 .399 .212 .302 .522 1. . . .
   corr  x7  .215 .295 .188 .467 .334 .232 1. . .
   corr  x8  .204 .257 .462 .522 .298 .355 .372 1. .
   corr  x9  .245 .272 .177 .301 .156 .246 .578 .422 1.
  mean   .22.1 19.8 16.9 23.3 21.9 17.3 17.9 19.1 19.8
  std    . 1.3 1.3 .99 1.25 2.1 1.3 2.2 2.0 1.5
  n     . 22 22 22 22 22 22 22 22 22
;
```

Example 29.22: Testing Equality of Covariance and Mean Matrices between Independent Groups
Chapter 29: The CALIS Procedure

data g3(type=corr);
   Input _type_ $ 1-8 _name_ $ 9-11 x1-x9;
datalines;
  corr  x1  1.    .    .    .    .    .    .    .    .    .
  corr  x2  .699  1.    .    .    .    .    .    .    .    .
  corr  x3  .488  .328  1.    .    .    .    .    .    .    .
  corr  x4  .235  .398  .413  1.    .    .    .    .    .    .
  corr  x5  .377  .265  .471  .376  1.    .    .    .    .    .
  corr  x6  .335  .412  .265  .314  .503  1.    .    .    .    .
  corr  x7  .243  .216  .192  .423  .369  .212  1.    .    .    .
  corr  x8  .217  .292  .423  .525  .219  .317  .376  1.    .    .
  mean . 22.2  20.9  15.4  25.1  22.6  16.3  19.3  20.2  19.5
  std .  1.5  1.0  1.04  1.5  1.9  1.6  2.4  2.2  1.6
  n .  20  20  20  20  20  20  20  20  20
;

Each of these data sets contains the information about the correlations, means, standard deviations, and sample sizes. Even though these data sets contain correlations, by default PROC CALIS analyzes the covariances and means.

The first hypothesis to test is the equality of covariance matrices and mean vectors:

\[ H_0 : \Sigma_1 = \Sigma_2 = \Sigma_3 \text{ and } \mu_1 = \mu_2 = \mu_3 \]

where \( \Sigma_1, \Sigma_2, \text{ and } \Sigma_3 \) are the population covariance matrices for the three independent groups, respectively, and \( \mu_1, \mu_2, \text{ and } \mu_3 \) are the population mean vectors for the three independent groups, respectively.

The following statements specify this test:

```sas
proc calis covpattern=eqcovmat meanpattern=eqmeanvec;
   var x1-x9;
   group 1 / data=g1;
   group 2 / data=g2;
   group 3 / data=g3;
   fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
run;
```

In the PROC CALIS statement, the COVPATTERN=EQCOVMAT option specifies the same covariance matrix for the three groups and the MEANPATTERN=EQMEANVEC option specifies the same mean vector for the three groups. The VAR statement specifies that x1–9 are the variables in the hypothesis test. Next, the GROUP statements specify the data sets for the three independent groups. You use the FITINDEX statement to limit the amount of output fit statistics to the quantities specified: the chi-square test (CHISQ), the degrees of freedom (DF), the significance value of the test statistic (PROBCHI), the root mean square error of approximation (RMSEA), Akaike’s information criterion (AIC), consistent Akaike’s information criterion (CAIC), and Schwarz’s Bayesian criterion (SBC). The first three quantities are useful for the chi-square model fit test, while the rest of the fit indices are useful for comparing competing models for the data. Because there are not many quantities in this customized fit summary table, the NOINDEXTYPE option is used to suppress the printing of the fit index types.

Output 29.22.1 shows the general modeling information, including the sample sizes, the models for the groups, the model types, and the analysis types.
Output 29.22.1  Modeling Information for Testing Equality of Covariance and Mean Matrices

Output 29.22.1 shows the initial mean vector and the initial covariance matrix specifications for Model 1, which fits to Group 1. PROC CALIS generates the mean parameter names _mean_1, _mean_2, . . . , and _mean_9 for the nine elements in the mean vector. It also generates the covariance parameter names _cov_1_1, _cov_2_1, . . . , and _cov_9_9 for the 45 nonredundant elements in the covariance matrix.

Output 29.22.2  Initial Mean Vector and Covariance Matrix for Model 1

<table>
<thead>
<tr>
<th>Variable</th>
<th>Parameter</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>_mean_1</td>
<td>.</td>
</tr>
<tr>
<td>x2</td>
<td>_mean_2</td>
<td>.</td>
</tr>
<tr>
<td>x3</td>
<td>_mean_3</td>
<td>.</td>
</tr>
<tr>
<td>x4</td>
<td>_mean_4</td>
<td>.</td>
</tr>
<tr>
<td>x5</td>
<td>_mean_5</td>
<td>.</td>
</tr>
<tr>
<td>x6</td>
<td>_mean_6</td>
<td>.</td>
</tr>
<tr>
<td>x7</td>
<td>_mean_7</td>
<td>.</td>
</tr>
<tr>
<td>x8</td>
<td>_mean_8</td>
<td>.</td>
</tr>
<tr>
<td>x9</td>
<td>_mean_9</td>
<td>.</td>
</tr>
</tbody>
</table>

```
x1 [ _cov_1_1 ] [ _cov_2_1 ] [ _cov_3_1 ] [ _cov_4_1 ] [ _cov_5_1 ] [ _cov_6_1 ] [ _cov_7_1 ] [ _cov_8_1 ] [ _cov_9_1 ]
x2 [ _cov_2_1 ] [ _cov_2_2 ] [ _cov_3_2 ] [ _cov_4_2 ] [ _cov_5_2 ] [ _cov_6_2 ] [ _cov_7_2 ] [ _cov_8_2 ] [ _cov_9_2 ]
x3 [ _cov_3_1 ] [ _cov_3_2 ] [ _cov_3_3 ] [ _cov_4_3 ] [ _cov_5_3 ] [ _cov_6_3 ] [ _cov_7_3 ] [ _cov_8_3 ] [ _cov_9_3 ]
x4 [ _cov_4_1 ] [ _cov_4_2 ] [ _cov_4_3 ] [ _cov_4_4 ] [ _cov_5_4 ] [ _cov_6_4 ] [ _cov_7_4 ] [ _cov_8_4 ] [ _cov_9_4 ]
x5 [ _cov_5_1 ] [ _cov_5_2 ] [ _cov_5_3 ] [ _cov_5_4 ] [ _cov_5_5 ] [ _cov_6_5 ] [ _cov_7_5 ] [ _cov_8_5 ] [ _cov_9_5 ]
x6 [ _cov_6_1 ] [ _cov_6_2 ] [ _cov_6_3 ] [ _cov_6_4 ] [ _cov_6_5 ] [ _cov_6_6 ] [ _cov_7_6 ] [ _cov_8_6 ] [ _cov_9_6 ]
x7 [ _cov_7_1 ] [ _cov_7_2 ] [ _cov_7_3 ] [ _cov_7_4 ] [ _cov_7_5 ] [ _cov_7_6 ] [ _cov_7_7 ] [ _cov_8_7 ] [ _cov_9_7 ]
x8 [ _cov_8_1 ] [ _cov_8_2 ] [ _cov_8_3 ] [ _cov_8_4 ] [ _cov_8_5 ] [ _cov_8_6 ] [ _cov_8_7 ] [ _cov_8_8 ] [ _cov_9_8 ]
x9 [ _cov_9_1 ] [ _cov_9_2 ] [ _cov_9_3 ] [ _cov_9_4 ] [ _cov_9_5 ] [ _cov_9_6 ] [ _cov_9_7 ] [ _cov_9_8 ] [ _cov_9_9 ]
```
Although not shown here, the initial mean vector and covariance matrices for Models 2 and 3 are exactly the same as those shown in Output 29.22.2, as required by the equality of covariance and mean matrices in the null hypothesis $H_0$.

Output 29.22.3 shows the customized fit summary table. The chi-square test statistic is 203.2605. The degrees of freedom is 108 and the $p$-value is less than 0.0001. Therefore, the hypothesis $H_0$ of equality in covariance and mean matrices is rejected for the three independent groups. The RMSEA index is much greater than 0.05, which does not indicate a good model fit. Other fit indices such as AIC, CAIC, and SBC are not interpreted for the fit of the model itself, but are useful for comparing competing models in the later discussion.

**Output 29.22.3** Fit Summary for Testing $H_0$: Equality of Covariance and Mean Matrices

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>203.2605</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>108</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.2100</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>311.2605</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>480.9897</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>426.9897</td>
</tr>
</tbody>
</table>

A less restrictive hypothesis is now considered. This hypothesis states the equality of covariance matrices only:

$$H_1 : \Sigma_1 = \Sigma_2 = \Sigma_3 (\mu_1, \mu_2, \text{ and } \mu_3 \text{ unconstrained})$$

$H_1$ differs from $H_0$ in that the population means in $H_1$ are not constrained. To test this hypothesis, you need to change the MEANPATTERN= option to use the SATURATED keyword, as shown in the following statements:

```plaintext
proc calis covpattern=eqcovmat meanpattern=saturated;
  var x1-x9;
  group 1 / data=g1;
  group 2 / data=g2;
  group 3 / data=g3;
  fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
run;
```

Output 29.22.4 shows the results of the testing $H_1$.

**Output 29.22.4** Fit Summary for Testing $H_1$: Equality of Covariance Matrices but Unconstrained Means

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>26.7897</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>90</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>1.0000</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0000</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>170.7897</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>397.0954</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>325.0954</td>
</tr>
</tbody>
</table>
The chi-square test statistic is 26.7897 (df = 90, p = 1.000). You cannot reject this null hypothesis about the equality of the population covariance matrices. The RMSEA value is virtually zero, which indicates a perfect fit. Comparing the models under $H_0$ and $H_1$, it is clear that the three groups are significantly different with regard to their mean vectors. By relaxing all the equality constraints on the means in $H_0$, $H_1$ is derived and is supported by the chi-square test. In addition, the RMSEA value for the model under $H_1$ is perfect. Because lower values of AIC, CAIC, and SBC values indicate better model fit (with the model complexity taken into account), these indices in Output 29.22.3 and Output 29.22.4 support that the model under $H_1$ is better than $H_0$.

However, in getting a superior model fit, $H_1$ might have relaxed more constraints than absolutely necessary for an optimal fit. That is, it might be possible to impose equality constraints on only some (but not all, as in $H_1$) of the means to reach the same or even better model fit (by the RMSEA, AIC, CAIC, or SBC criterion) than the model under $H_1$. But how can you determine this set of constrained means?

To answer this question, you conduct an exploratory analysis of the data by using some model modification techniques. Models established from exploratory analysis should be validated by external data in the future. However, this example demonstrates the exploratory part only.

Beginning with the model under $H_0$, you can manually take away some particular constraints on the means and explore whether the revised model improves the fit. If the revised model fits better, you can repeat the process until you cannot improve more. Ultimately, you might be able to find the “best” model between the models specified under $H_0$ and $H_1$. Such an exploratory analysis is laborious, considering the vast possibilities of constraints on the nine variable means in three independent groups that you could attempt to release. Fortunately, PROC CALIS provides some model modification statistics, called the LM (Lagrange multiplier) statistics, to assist this kind of exploratory analysis.

The following statements specify the model under $H_0$, but now with the MODIFICATION option added to the PROC CALIS statement:

```plaintext
proc calis covpattern=eqcovmat meanpattern=eqmeanvec modification;
  var x1-x9;
  group 1 / data=g1;
  group 2 / data=g2;
  group 3 / data=g3;
  fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
run;
```

The MODIFICATION option requests the so-called LM (Lagrange multiplier) statistics for releasing the parameter constraints. These constraints include the cross-group or within-group constraints and the fixed values in the model. For the model under $H_0$, the covariances and the means are all constrained across groups. These are the equality constraints that you would like to release to obtain a better model fit. Output 29.22.5 shows the results of the LM statistics for releasing these equality constraints in variances, covariances, and means.
### Output 29.22.5 Lagrange Multiplier Statistics for Releasing the Equality Constraints

<table>
<thead>
<tr>
<th>Parm</th>
<th>Model Type</th>
<th>Var1</th>
<th>Var2</th>
<th>LM Stat</th>
<th>Pr &gt; ChiSq</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>_cov_1_1</td>
<td>1 COV</td>
<td>x1</td>
<td>x1</td>
<td>0.01137</td>
<td>0.9151</td>
<td>0.0178  -0.0355</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x1</td>
<td></td>
<td>1.00150</td>
<td>0.3169</td>
<td>0.1729  -0.3212</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x1</td>
<td></td>
<td>1.28632</td>
<td>0.2567</td>
<td>-0.1818 0.3923</td>
</tr>
<tr>
<td>_cov_2_1</td>
<td>1 COV</td>
<td>x2</td>
<td>x1</td>
<td>2.19353</td>
<td>0.1386</td>
<td>0.2038  -0.4076</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x2</td>
<td></td>
<td>0.77014</td>
<td>0.3802</td>
<td>-0.1253 0.2327</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x2</td>
<td></td>
<td>0.36128</td>
<td>0.5478</td>
<td>-0.0796 0.1718</td>
</tr>
<tr>
<td>_cov_2_2</td>
<td>1 COV</td>
<td>x2</td>
<td>x2</td>
<td>3.12065</td>
<td>0.0773</td>
<td>-0.4344 0.8687</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x2</td>
<td></td>
<td>0.05704</td>
<td>0.8112</td>
<td>-0.0609 0.1132</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x2</td>
<td></td>
<td>4.14151</td>
<td>0.0418</td>
<td>0.4817  -1.0395</td>
</tr>
<tr>
<td>_cov_3_1</td>
<td>1 COV</td>
<td>x3</td>
<td>x1</td>
<td>0.00672</td>
<td>0.9347</td>
<td>0.00888 -0.0178</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x3</td>
<td></td>
<td>2.23758</td>
<td>0.1347</td>
<td>-0.1681 0.3122</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x3</td>
<td></td>
<td>2.10455</td>
<td>0.1469</td>
<td>0.1512  -0.3264</td>
</tr>
<tr>
<td>_cov_3_2</td>
<td>1 COV</td>
<td>x3</td>
<td>x2</td>
<td>2.18538</td>
<td>0.1393</td>
<td>-0.1940 0.3881</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x3</td>
<td></td>
<td>3.14532</td>
<td>0.0761</td>
<td>0.2416  -0.4487</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x3</td>
<td></td>
<td>0.10264</td>
<td>0.7487</td>
<td>-0.0405 0.0874</td>
</tr>
<tr>
<td>_cov_3_3</td>
<td>1 COV</td>
<td>x3</td>
<td>x3</td>
<td>1.56813</td>
<td>0.2105</td>
<td>0.1815  -0.3630</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x3</td>
<td></td>
<td>0.66118</td>
<td>0.4161</td>
<td>0.1223  -0.2272</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x3</td>
<td></td>
<td>4.42160</td>
<td>0.0355</td>
<td>-0.2934 0.6332</td>
</tr>
<tr>
<td>_cov_4_1</td>
<td>1 COV</td>
<td>x4</td>
<td>x1</td>
<td>0.31691</td>
<td>0.5735</td>
<td>-0.0667 0.1333</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x4</td>
<td></td>
<td>0.32615</td>
<td>0.5679</td>
<td>0.0702  -0.1304</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x4</td>
<td></td>
<td>0.0002277</td>
<td>0.9880</td>
<td>-0.00172 0.00371</td>
</tr>
<tr>
<td>_cov_4_2</td>
<td>1 COV</td>
<td>x4</td>
<td>x2</td>
<td>0.73377</td>
<td>0.3917</td>
<td>0.1242  -0.2484</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x4</td>
<td></td>
<td>0.53196</td>
<td>0.4658</td>
<td>-0.1097 0.2038</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x4</td>
<td></td>
<td>0.01445</td>
<td>0.9043</td>
<td>-0.0168 0.0362</td>
</tr>
<tr>
<td>_cov_4_3</td>
<td>1 COV</td>
<td>x4</td>
<td>x3</td>
<td>0.0000258</td>
<td>0.9959</td>
<td>0.000547 -0.00109</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x4</td>
<td></td>
<td>0.24892</td>
<td>0.6178</td>
<td>-0.0558 0.1036</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x4</td>
<td></td>
<td>0.25646</td>
<td>0.6126</td>
<td>0.0525  -0.1134</td>
</tr>
<tr>
<td>_cov_4_4</td>
<td>1 COV</td>
<td>x4</td>
<td>x4</td>
<td>0.04412</td>
<td>0.8336</td>
<td>0.0361  -0.0722</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x4</td>
<td></td>
<td>0.52198</td>
<td>0.4700</td>
<td>0.1288  -0.2392</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x4</td>
<td></td>
<td>0.90948</td>
<td>0.3403</td>
<td>-0.1577 0.3403</td>
</tr>
<tr>
<td>_cov_5_1</td>
<td>1 COV</td>
<td>x5</td>
<td>x1</td>
<td>0.0008607</td>
<td>0.9766</td>
<td>-0.00477 0.00953</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x5</td>
<td></td>
<td>0.01238</td>
<td>0.9114</td>
<td>0.0188  -0.0348</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x5</td>
<td></td>
<td>0.00712</td>
<td>0.9328</td>
<td>-0.0132 0.0285</td>
</tr>
<tr>
<td>_cov_5_2</td>
<td>1 COV</td>
<td>x5</td>
<td>x2</td>
<td>0.10637</td>
<td>0.7443</td>
<td>-0.0649 0.1297</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x5</td>
<td></td>
<td>0.00631</td>
<td>0.9367</td>
<td>-0.0164 0.0304</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x5</td>
<td></td>
<td>0.16971</td>
<td>0.6804</td>
<td>0.0789  -0.1702</td>
</tr>
<tr>
<td>_cov_5_3</td>
<td>1 COV</td>
<td>x5</td>
<td>x3</td>
<td>0.06645</td>
<td>0.7966</td>
<td>-0.0385 0.0771</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x5</td>
<td></td>
<td>0.0008275</td>
<td>0.9771</td>
<td>0.00446 -0.00829</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x5</td>
<td></td>
<td>0.05370</td>
<td>0.8167</td>
<td>0.0334  -0.0720</td>
</tr>
<tr>
<td>_cov_5_4</td>
<td>1 COV</td>
<td>x5</td>
<td>x4</td>
<td>0.24212</td>
<td>0.6227</td>
<td>0.0809  -0.1617</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x5</td>
<td></td>
<td>0.04459</td>
<td>0.8328</td>
<td>-0.0360 0.0669</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x5</td>
<td></td>
<td>0.07959</td>
<td>0.7779</td>
<td>-0.0446 0.0963</td>
</tr>
<tr>
<td>_cov_5_5</td>
<td>1 COV</td>
<td>x5</td>
<td>x5</td>
<td>0.01778</td>
<td>0.8939</td>
<td>-0.0431 0.0862</td>
</tr>
<tr>
<td>2</td>
<td>COV</td>
<td>x5</td>
<td></td>
<td>0.08223</td>
<td>0.7743</td>
<td>-0.0962 0.1787</td>
</tr>
<tr>
<td>3</td>
<td>COV</td>
<td>x5</td>
<td></td>
<td>0.18417</td>
<td>0.6678</td>
<td>0.1336  -0.2883</td>
</tr>
<tr>
<td>_cov_6_1</td>
<td>1 COV</td>
<td>x6</td>
<td>x1</td>
<td>0.29558</td>
<td>0.5867</td>
<td>-0.0721 0.1442</td>
</tr>
</tbody>
</table>
### Output 29.22.5 continued

<table>
<thead>
<tr>
<th>Parm</th>
<th>Model</th>
<th>Type</th>
<th>Var1</th>
<th>Var2</th>
<th>LM Stat</th>
<th>Pr &gt; ChiSq</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Original Parm</td>
<td>Released Parm</td>
<td></td>
</tr>
<tr>
<td>2 COV</td>
<td>x6</td>
<td>x1</td>
<td></td>
<td></td>
<td>0.26589</td>
<td>0.6061</td>
<td>-0.0710</td>
</tr>
<tr>
<td>3 COV</td>
<td>x6</td>
<td>x1</td>
<td></td>
<td></td>
<td>1.16570</td>
<td>0.2803</td>
<td>0.1378</td>
</tr>
<tr>
<td>_cov_6_2</td>
<td>x6</td>
<td>x2</td>
<td></td>
<td></td>
<td>0.00228</td>
<td>0.9619</td>
<td>-0.00780</td>
</tr>
<tr>
<td>2 COV</td>
<td>x6</td>
<td>x2</td>
<td></td>
<td></td>
<td>1.00319</td>
<td>0.3165</td>
<td>0.1697</td>
</tr>
<tr>
<td>3 COV</td>
<td>x6</td>
<td>x2</td>
<td></td>
<td></td>
<td>0.95767</td>
<td>0.3278</td>
<td>-0.1538</td>
</tr>
<tr>
<td>_cov_6_3</td>
<td>x6</td>
<td>x3</td>
<td></td>
<td></td>
<td>1.39116</td>
<td>0.2382</td>
<td>0.1513</td>
</tr>
<tr>
<td>2 COV</td>
<td>x6</td>
<td>x3</td>
<td></td>
<td></td>
<td>0.08741</td>
<td>0.7675</td>
<td>-0.0394</td>
</tr>
<tr>
<td>3 COV</td>
<td>x6</td>
<td>x3</td>
<td></td>
<td></td>
<td>0.79586</td>
<td>0.3723</td>
<td>-0.1102</td>
</tr>
<tr>
<td>_cov_6_4</td>
<td>x6</td>
<td>x4</td>
<td></td>
<td></td>
<td>0.46031</td>
<td>0.4975</td>
<td>-0.0947</td>
</tr>
<tr>
<td>2 COV</td>
<td>x6</td>
<td>x4</td>
<td></td>
<td></td>
<td>0.04254</td>
<td>0.8366</td>
<td>0.0299</td>
</tr>
<tr>
<td>3 COV</td>
<td>x6</td>
<td>x4</td>
<td></td>
<td></td>
<td>0.22665</td>
<td>0.6340</td>
<td>0.0640</td>
</tr>
<tr>
<td>_cov_6_5</td>
<td>x6</td>
<td>x5</td>
<td></td>
<td></td>
<td>0.14991</td>
<td>0.6986</td>
<td>-0.0700</td>
</tr>
<tr>
<td>2 COV</td>
<td>x6</td>
<td>x5</td>
<td></td>
<td></td>
<td>0.04723</td>
<td>0.8280</td>
<td>0.0408</td>
</tr>
<tr>
<td>3 COV</td>
<td>x6</td>
<td>x5</td>
<td></td>
<td></td>
<td>0.02874</td>
<td>0.8654</td>
<td>0.0295</td>
</tr>
<tr>
<td>_cov_6_6</td>
<td>x6</td>
<td>x6</td>
<td></td>
<td></td>
<td>0.22550</td>
<td>0.6349</td>
<td>0.1079</td>
</tr>
<tr>
<td>2 COV</td>
<td>x6</td>
<td>x6</td>
<td></td>
<td></td>
<td>0.04390</td>
<td>0.8340</td>
<td>0.0494</td>
</tr>
<tr>
<td>3 COV</td>
<td>x6</td>
<td>x6</td>
<td></td>
<td></td>
<td>0.48451</td>
<td>0.4864</td>
<td>-0.1523</td>
</tr>
<tr>
<td>_cov_7_1</td>
<td>x7</td>
<td>x1</td>
<td></td>
<td></td>
<td>0.50774</td>
<td>0.4761</td>
<td>0.1203</td>
</tr>
<tr>
<td>2 COV</td>
<td>x7</td>
<td>x1</td>
<td></td>
<td></td>
<td>0.01246</td>
<td>0.9111</td>
<td>-0.0196</td>
</tr>
<tr>
<td>3 COV</td>
<td>x7</td>
<td>x1</td>
<td></td>
<td></td>
<td>0.36926</td>
<td>0.5434</td>
<td>-0.0988</td>
</tr>
<tr>
<td>_cov_7_2</td>
<td>x7</td>
<td>x2</td>
<td></td>
<td></td>
<td>0.01235</td>
<td>0.9115</td>
<td>0.0228</td>
</tr>
<tr>
<td>2 COV</td>
<td>x7</td>
<td>x2</td>
<td></td>
<td></td>
<td>0.16400</td>
<td>0.6855</td>
<td>-0.0861</td>
</tr>
<tr>
<td>3 COV</td>
<td>x7</td>
<td>x2</td>
<td></td>
<td></td>
<td>0.09159</td>
<td>0.7622</td>
<td>0.0597</td>
</tr>
<tr>
<td>_cov_7_3</td>
<td>x7</td>
<td>x3</td>
<td></td>
<td></td>
<td>0.16844</td>
<td>0.6815</td>
<td>-0.0644</td>
</tr>
<tr>
<td>2 COV</td>
<td>x7</td>
<td>x3</td>
<td></td>
<td></td>
<td>0.15095</td>
<td>0.6976</td>
<td>0.0633</td>
</tr>
<tr>
<td>3 COV</td>
<td>x7</td>
<td>x3</td>
<td></td>
<td></td>
<td>0.0003079</td>
<td>0.9860</td>
<td>0.00265</td>
</tr>
<tr>
<td>_cov_7_4</td>
<td>x7</td>
<td>x4</td>
<td></td>
<td></td>
<td>0.22542</td>
<td>0.6349</td>
<td>-0.0776</td>
</tr>
<tr>
<td>2 COV</td>
<td>x7</td>
<td>x4</td>
<td></td>
<td></td>
<td>0.00754</td>
<td>0.9308</td>
<td>0.0147</td>
</tr>
<tr>
<td>3 COV</td>
<td>x7</td>
<td>x4</td>
<td></td>
<td></td>
<td>0.15376</td>
<td>0.6950</td>
<td>0.0617</td>
</tr>
<tr>
<td>_cov_7_5</td>
<td>x7</td>
<td>x5</td>
<td></td>
<td></td>
<td>0.07831</td>
<td>0.7796</td>
<td>-0.0631</td>
</tr>
<tr>
<td>2 COV</td>
<td>x7</td>
<td>x5</td>
<td></td>
<td></td>
<td>0.07552</td>
<td>0.7835</td>
<td>0.0643</td>
</tr>
<tr>
<td>3 COV</td>
<td>x7</td>
<td>x5</td>
<td></td>
<td></td>
<td>3.293E-6</td>
<td>0.9986</td>
<td>0.000394</td>
</tr>
<tr>
<td>_cov_7_6</td>
<td>x7</td>
<td>x6</td>
<td></td>
<td></td>
<td>0.13810</td>
<td>0.7102</td>
<td>0.0726</td>
</tr>
<tr>
<td>2 COV</td>
<td>x7</td>
<td>x6</td>
<td></td>
<td></td>
<td>0.0001086</td>
<td>0.9917</td>
<td>0.00211</td>
</tr>
<tr>
<td>3 COV</td>
<td>x7</td>
<td>x6</td>
<td></td>
<td></td>
<td>0.14999</td>
<td>0.6985</td>
<td>-0.0729</td>
</tr>
<tr>
<td>_cov_7_7</td>
<td>x7</td>
<td>x7</td>
<td></td>
<td></td>
<td>0.09334</td>
<td>0.7600</td>
<td>0.1051</td>
</tr>
<tr>
<td>2 COV</td>
<td>x7</td>
<td>x7</td>
<td></td>
<td></td>
<td>0.00128</td>
<td>0.9714</td>
<td>0.0128</td>
</tr>
<tr>
<td>3 COV</td>
<td>x7</td>
<td>x7</td>
<td></td>
<td></td>
<td>0.11994</td>
<td>0.7291</td>
<td>-0.1147</td>
</tr>
<tr>
<td>_cov_8_1</td>
<td>x8</td>
<td>x1</td>
<td></td>
<td></td>
<td>0.04800</td>
<td>0.8266</td>
<td>0.0353</td>
</tr>
<tr>
<td>2 COV</td>
<td>x8</td>
<td>x1</td>
<td></td>
<td></td>
<td>0.19725</td>
<td>0.6569</td>
<td>0.0743</td>
</tr>
<tr>
<td>3 COV</td>
<td>x8</td>
<td>x1</td>
<td></td>
<td></td>
<td>0.45888</td>
<td>0.4981</td>
<td>-0.1051</td>
</tr>
<tr>
<td>_cov_8_2</td>
<td>x8</td>
<td>x2</td>
<td></td>
<td></td>
<td>0.13689</td>
<td>0.7114</td>
<td>0.0727</td>
</tr>
<tr>
<td>2 COV</td>
<td>x8</td>
<td>x2</td>
<td></td>
<td></td>
<td>0.31671</td>
<td>0.5736</td>
<td>-0.1147</td>
</tr>
<tr>
<td>3 COV</td>
<td>x8</td>
<td>x2</td>
<td></td>
<td></td>
<td>0.04084</td>
<td>0.8398</td>
<td>0.0382</td>
</tr>
<tr>
<td>_cov_8_3</td>
<td>x8</td>
<td>x3</td>
<td></td>
<td></td>
<td>0.37615</td>
<td>0.5397</td>
<td>-0.0904</td>
</tr>
<tr>
<td>2 COV</td>
<td>x8</td>
<td>x3</td>
<td></td>
<td></td>
<td>0.00452</td>
<td>0.9464</td>
<td>-0.0103</td>
</tr>
</tbody>
</table>
### Output 29.22.5 continued

<table>
<thead>
<tr>
<th>Parm</th>
<th>Model Type</th>
<th>Var1</th>
<th>Var2</th>
<th>LM Stat</th>
<th>Pr &gt; ChiSq</th>
<th>Original Parm</th>
<th>Released Parm</th>
</tr>
</thead>
<tbody>
<tr>
<td>_cov_8.4</td>
<td>COV</td>
<td>x8</td>
<td>x3</td>
<td>0.47678</td>
<td>0.4899</td>
<td>0.0980</td>
<td>-0.2114</td>
</tr>
<tr>
<td>_cov_8.5</td>
<td>COV</td>
<td>x8</td>
<td>x5</td>
<td>0.01378</td>
<td>0.9066</td>
<td>-0.0267</td>
<td>0.0533</td>
</tr>
<tr>
<td>_cov_8.6</td>
<td>COV</td>
<td>x8</td>
<td>x6</td>
<td>0.0007193</td>
<td>0.9786</td>
<td>0.00510</td>
<td>-0.0102</td>
</tr>
<tr>
<td>_cov_8.7</td>
<td>COV</td>
<td>x8</td>
<td>x7</td>
<td>0.16543</td>
<td>0.6842</td>
<td>0.0952</td>
<td>-0.1904</td>
</tr>
<tr>
<td>_cov_8.8</td>
<td>COV</td>
<td>x8</td>
<td>x8</td>
<td>0.00581</td>
<td>0.9392</td>
<td>0.0244</td>
<td>-0.0487</td>
</tr>
<tr>
<td>_cov_9.1</td>
<td>COV</td>
<td>x9</td>
<td>x1</td>
<td>0.19272</td>
<td>0.6607</td>
<td>-0.0532</td>
<td>0.1063</td>
</tr>
<tr>
<td>_cov_9.2</td>
<td>COV</td>
<td>x9</td>
<td>x2</td>
<td>0.09017</td>
<td>0.7640</td>
<td>-0.0446</td>
<td>0.0892</td>
</tr>
<tr>
<td>_cov_9.3</td>
<td>COV</td>
<td>x9</td>
<td>x3</td>
<td>0.44236</td>
<td>0.5060</td>
<td>0.0758</td>
<td>-0.1516</td>
</tr>
<tr>
<td>_cov_9.4</td>
<td>COV</td>
<td>x9</td>
<td>x3</td>
<td>0.12761</td>
<td>0.7209</td>
<td>-0.0422</td>
<td>0.0784</td>
</tr>
<tr>
<td>_cov_9.5</td>
<td>COV</td>
<td>x9</td>
<td>x5</td>
<td>0.09470</td>
<td>0.7583</td>
<td>-0.0338</td>
<td>0.0728</td>
</tr>
<tr>
<td>_cov_9.6</td>
<td>COV</td>
<td>x9</td>
<td>x6</td>
<td>0.03989</td>
<td>0.8417</td>
<td>-0.0282</td>
<td>0.0563</td>
</tr>
<tr>
<td>_cov_9.7</td>
<td>COV</td>
<td>x9</td>
<td>x6</td>
<td>0.15069</td>
<td>0.6979</td>
<td>-0.0568</td>
<td>0.1055</td>
</tr>
<tr>
<td>_cov_9.8</td>
<td>COV</td>
<td>x9</td>
<td>x7</td>
<td>0.36051</td>
<td>0.5482</td>
<td>0.0815</td>
<td>-0.1759</td>
</tr>
<tr>
<td>_cov_9.9</td>
<td>COV</td>
<td>x9</td>
<td>x7</td>
<td>0.03398</td>
<td>0.8537</td>
<td>-0.0284</td>
<td>0.0567</td>
</tr>
<tr>
<td>_mean_1</td>
<td>MEAN</td>
<td>x1</td>
<td></td>
<td>11.09173</td>
<td>0.0009</td>
<td>0.3453</td>
<td>-0.6906</td>
</tr>
<tr>
<td></td>
<td>MEAN</td>
<td>x1</td>
<td></td>
<td>1.21196</td>
<td>0.2709</td>
<td>-0.1184</td>
<td>0.2200</td>
</tr>
<tr>
<td></td>
<td>MEAN</td>
<td>x1</td>
<td></td>
<td>5.04550</td>
<td>0.0247</td>
<td>-0.2242</td>
<td>0.4838</td>
</tr>
</tbody>
</table>
To use the results of this table, you look for parameters that have large LM statistics (in the LM Stat column). Equivalently, you can look for parameters that have small p-values (in the Pr > ChiSq column). Loosely speaking, an LM statistic estimates the reduction of model fit chi-square statistic if you release the constraint on the corresponding parameter. The p-value indicates whether the improvement would be significant. Therefore, releasing those parameters with a high LM statistic and small p-value would be the key to model improvements. Bear in mind that the LM statistics are linear approximations and they might not be very accurate as estimates of the actual model improvement, which could only be accessed when you refit the model with the particular constraint released. Nonetheless, the LM statistics could still be very useful because they show which constraints could potentially improve the model the most.

Output 29.22.5 shows the results from releasing the constraints on the variances and covariances first. Each constrained element of the covariance matrix has three rows, respectively, for the three models (or groups). For example, the first parameter is _cov_1_1, which is the same variance parameter for x1 in the three models. The first row shows that if you release the variance of x1 in Model 1 from the constraint (while keeping the variances of x1 being constrained between Models 2 and 3), the LM statistic is 0.01127, and the corresponding p-value is 0.9155. This means that the model fit improvement would be very small and so you do not expect a significant model fit improvement by releasing this constraint. The columns entitled “Changes” show the estimated parameter changes in the original parameters (that is, _cov_1_1 for Models 2 and 3) and in the released parameter (that is, the new parameter for the variance of x1 in Model 1) if you
release the corresponding equality constraint. These two “Changes” columns are not very useful for the present purpose.

Looking through the results for the variance and covariance constraints, you can see that almost all the associated $p$-values are large (that is, as compared with the conventional 0.05 level for significance). Therefore, all these constraints on variances and covariances would not improve the model fit significantly. In contrast, the constraints on the means show that several of them could be released for a sizable model fit improvement. The largest LM statistic in the table is the one for \_mean\_3 in Model 3. The LM statistic is 22.27678 and its corresponding $p$-value is less than 0.0001. This means that if the mean of $x_3$ in Model 3 were not constrained with the means of $x_3$ in Models 1 and 2, you would have expected a reduction in the model fit chi-square statistic that is estimated at 22.27678. Other notable LM statistics are those for \_mean\_1 in Model 1, \_mean\_2 in Model 1 or 2, and \_mean\_6 in Model 2.

Two important points are noted about the use of the LM statistics. First, the LM statistics are not additive. You cannot expect that the total reduction in model fit chi-square for releasing a particular set of parameter constraints is the sum of the corresponding LM statistics. Second, once you release a particular constraint and refit the model, the LM statistics in the revised model might not follow the same pattern as those LM statistics in the original model. Basically, these are due to the nonlinearity of the fit function and the dependence of the parameter estimates. Therefore, in order to find the best model for the data, it would be more sensible to adopt a one-at-a-time approach to release the constraints. That is, you release one constraint at a time and refit the model to see if you can release more constraints to improve the model fit.

According to the results of LM statistics in Output 29.22.5, you first release the constraint on the \_mean\_3 parameter, which is for the mean of $x_3$ in Model 3. The following statements fit such a model:

```plaintext
proc calis modification;
   var x1-x9;
   group 1 / data=g1;
   group 2 / data=g2;
   group 3 / data=g3;
   model 1 / group = 1;   
      mstruct;
      matrix _cov_ = cov01-cov45;
      matrix _mean_ = mean1-mean9;
   model 2 / group = 2;
      refmodel 1;
   model 3 / group = 3;
      refmodel 1;
      renameparm mean3=mean3_mdl3;
   fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
run;
```

Because the revised model is no longer a supported built-in MSTRUCT model, you cannot use the MEANPATTERN= or the COVPATTERN= options any more. Instead, you now use the MSTRUCT modeling language to specify the covariance and mean patterns. Model 1, which fits to Group 1, is an MSTRUCT model with variance and covariance parameters cov01–cov45 and mean parameters mean1–mean9. Model 2, which fits to Group 2, refers to the specifications of Model 1, as indicated in a REFMODEL statement. Hence, Model 1 and Model 2 are completely constrained in variances, covariances, and means. Model 3, which fits to Group 3, also refers to the specifications of Model 1, as indicated in another REFMODEL statement. However, the RENAMEPARM statement renames the parameter mean3 in the reference model (that is, Model 1) to a new name mean3_mdl3. As a result, all variance, covariance, and mean parameters except one in Model 3 are constrained to be the same as those in Model 1. The mean of $x_3$ in Model 3 is the
only parameter that is not constrained with any other parameters. This forms the first revised model from $H_0$. The MODIFICATION option is specified again to determine whether a further model fit improvement is possible.

Output 29.22.6 shows the modeling information of the first revised model. It shows that Models 2 and 3 make references to Model 1. Therefore, parameters between models are constrained by referencing.

**Output 29.22.6** Modeling Information for The First Revised Model

<table>
<thead>
<tr>
<th>Group</th>
<th>Data Set</th>
<th>N Obs</th>
<th>Model</th>
<th>Type</th>
<th>Base Model</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WORK.G1</td>
<td>21</td>
<td>Model 1</td>
<td>MSTRUCT</td>
<td></td>
<td>Means and Covariances</td>
</tr>
<tr>
<td>2</td>
<td>WORK.G2</td>
<td>22</td>
<td>Model 2</td>
<td>MSTRUCT Model 1</td>
<td></td>
<td>Means and Covariances</td>
</tr>
<tr>
<td>3</td>
<td>WORK.G3</td>
<td>20</td>
<td>Model 3</td>
<td>MSTRUCT Model 1</td>
<td></td>
<td>Means and Covariances</td>
</tr>
</tbody>
</table>

Output 29.22.7 shows the initial specifications of the means, variances, and covariances in Model 1.

**Output 29.22.7** Initial Mean Vector and Covariance Matrix for Model 1 in the First Revised Model

```
<table>
<thead>
<tr>
<th>Variable</th>
<th>Parameter</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>mean1</td>
<td>.</td>
</tr>
<tr>
<td>x2</td>
<td>mean2</td>
<td>.</td>
</tr>
<tr>
<td>x3</td>
<td>mean3</td>
<td>.</td>
</tr>
<tr>
<td>x4</td>
<td>mean4</td>
<td>.</td>
</tr>
<tr>
<td>x5</td>
<td>mean5</td>
<td>.</td>
</tr>
<tr>
<td>x6</td>
<td>mean6</td>
<td>.</td>
</tr>
<tr>
<td>x7</td>
<td>mean7</td>
<td>.</td>
</tr>
<tr>
<td>x8</td>
<td>mean8</td>
<td>.</td>
</tr>
<tr>
<td>x9</td>
<td>mean9</td>
<td>.</td>
</tr>
</tbody>
</table>
```

```
x1 = [cov01] [cov02] [cov04] [cov07] [cov11] [cov16] [cov22] [cov29] [cov37]  
x2 = [cov02] [cov03] [cov05] [cov08] [cov12] [cov17] [cov23] [cov30] [cov38]  
x3 = [cov04] [cov05] [cov06] [cov09] [cov13] [cov18] [cov24] [cov31] [cov39]  
x4 = [cov07] [cov08] [cov09] [cov10] [cov14] [cov19] [cov25] [cov32] [cov40]  
x5 = [cov11] [cov12] [cov13] [cov14] [cov15] [cov20] [cov26] [cov33] [cov41]  
x6 = [cov16] [cov17] [cov18] [cov19] [cov20] [cov21] [cov27] [cov34] [cov42]  
x7 = [cov22] [cov23] [cov24] [cov25] [cov26] [cov27] [cov28] [cov35] [cov43]  
x8 = [cov29] [cov30] [cov31] [cov32] [cov33] [cov34] [cov35] [cov36] [cov44]  
x9 = [cov37] [cov38] [cov39] [cov40] [cov41] [cov42] [cov43] [cov44] [cov45]  
```
Output 29.22.8 shows the initial specifications of the means in Model 2. The mean parameters in Model 2 are exactly the same as those in Model 1, as shown in Output 29.22.7. The variance and covariance parameters in Model 2 are also exactly the same as those in Model 1, but are not shown here to conserve space.

**Output 29.22.8** Initial Mean Vector for Model 2 in the First Revised Model

<table>
<thead>
<tr>
<th>Variable</th>
<th>Parameter</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>mean1</td>
<td>.</td>
</tr>
<tr>
<td>x2</td>
<td>mean2</td>
<td>.</td>
</tr>
<tr>
<td>x3</td>
<td>mean3</td>
<td>.</td>
</tr>
<tr>
<td>x4</td>
<td>mean4</td>
<td>.</td>
</tr>
<tr>
<td>x5</td>
<td>mean5</td>
<td>.</td>
</tr>
<tr>
<td>x6</td>
<td>mean6</td>
<td>.</td>
</tr>
<tr>
<td>x7</td>
<td>mean7</td>
<td>.</td>
</tr>
<tr>
<td>x8</td>
<td>mean8</td>
<td>.</td>
</tr>
<tr>
<td>x9</td>
<td>mean9</td>
<td>.</td>
</tr>
</tbody>
</table>

Output 29.22.9 shows the initial specifications of the means in Model 3. All but one mean parameter in Model 3 are exactly the same as those in Models 1 and 2, as shown in Output 29.22.7 and Output 29.22.8, respectively. The mean for x3 in Model 3 is mean3_mdl3, which is now a distinct parameter, and therefore it is not constrained with any other parameters in the first or the second models for Groups 1 or 2. However, the variance and covariance parameters in Model 3 are exactly the same as those in Model 1. They are not shown here to conserve space.

**Output 29.22.9** Initial Mean Vector for Model 3 in the First Revised Model

<table>
<thead>
<tr>
<th>Variable</th>
<th>Parameter</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>mean1</td>
<td>.</td>
</tr>
<tr>
<td>x2</td>
<td>mean2</td>
<td>.</td>
</tr>
<tr>
<td>x3</td>
<td>mean3.mdl3</td>
<td>.</td>
</tr>
<tr>
<td>x4</td>
<td>mean4</td>
<td>.</td>
</tr>
<tr>
<td>x5</td>
<td>mean5</td>
<td>.</td>
</tr>
<tr>
<td>x6</td>
<td>mean6</td>
<td>.</td>
</tr>
<tr>
<td>x7</td>
<td>mean7</td>
<td>.</td>
</tr>
<tr>
<td>x8</td>
<td>mean8</td>
<td>.</td>
</tr>
<tr>
<td>x9</td>
<td>mean9</td>
<td>.</td>
</tr>
</tbody>
</table>
Output 29.22.10 shows the fit summary of the first revised model. The model fit chi-square is 148.8865, which drops quite a bit from the original model under $H_0$. The $p$-value of the model fit chi-square is 0.0046, which is statistically significant. The RMSEA value is 0.1399, which is also a sizable improvement. All the AIC, CAIC, and SBC values are reduced, indicating better model fit than the model under $H_0$.

**Output 29.22.10**  Fit Summary for the First Revised Model

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>148.8865</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>107</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.0046</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.1399</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>258.8865</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>431.7589</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>376.7589</td>
</tr>
</tbody>
</table>

Output 29.22.11 shows the LM statistics for releasing the equality constraints in the first revised model. Almost all of the results for the variance and covariance constraints are omitted because their LM statistics are not significant. However, Output 29.22.11 shows all the LM statistics for releasing the constraints in means. The mean of $x_2$ in Model 2 has the largest LM statistic at 26.25044.
### Output 29.22.11  LM Statistics for Releasing the Equality Constraints in the First Revised Model

<table>
<thead>
<tr>
<th>Parm</th>
<th>Model Type</th>
<th>Var1</th>
<th>Var2</th>
<th>LM Stat</th>
<th>Pr &gt; ChiSq</th>
<th>Original Parm</th>
<th>Released Parm</th>
</tr>
</thead>
<tbody>
<tr>
<td>cov01</td>
<td>COV</td>
<td>x1</td>
<td>x1</td>
<td>0.64999</td>
<td>0.4201</td>
<td>0.1050</td>
<td>-0.2100</td>
</tr>
<tr>
<td>2 COV</td>
<td>x1</td>
<td>x1</td>
<td>0.41758</td>
<td>0.5181</td>
<td>0.0874</td>
<td>-0.1622</td>
<td></td>
</tr>
<tr>
<td>3 COV</td>
<td>x1</td>
<td>x1</td>
<td>2.18923</td>
<td>0.1390</td>
<td>-0.1855</td>
<td>0.4004</td>
<td></td>
</tr>
<tr>
<td>mean1</td>
<td>MEAN</td>
<td>x1</td>
<td></td>
<td>9.26674</td>
<td>0.0023</td>
<td>0.2872</td>
<td>-0.5745</td>
</tr>
<tr>
<td>2 MEAN</td>
<td>x1</td>
<td></td>
<td>3.00599</td>
<td>0.0830</td>
<td>-0.1702</td>
<td>0.3160</td>
<td></td>
</tr>
<tr>
<td>3 MEAN</td>
<td>x1</td>
<td></td>
<td>2.13787</td>
<td>0.1437</td>
<td>-0.1481</td>
<td>0.3196</td>
<td></td>
</tr>
<tr>
<td>mean2</td>
<td>MEAN</td>
<td>x2</td>
<td></td>
<td>26.25115</td>
<td>&lt;.0001</td>
<td>-0.6568</td>
<td>1.3135</td>
</tr>
<tr>
<td>2 MEAN</td>
<td>x2</td>
<td></td>
<td>12.34638</td>
<td>0.0004</td>
<td>0.4674</td>
<td>-0.8680</td>
<td></td>
</tr>
<tr>
<td>3 MEAN</td>
<td>x2</td>
<td></td>
<td>2.52683</td>
<td>0.1119</td>
<td>0.1962</td>
<td>-0.4234</td>
<td></td>
</tr>
<tr>
<td>mean3</td>
<td>MEAN</td>
<td>x3</td>
<td></td>
<td>0.58891</td>
<td>0.4428</td>
<td>-0.0787</td>
<td>0.0827</td>
</tr>
<tr>
<td>2 MEAN</td>
<td>x3</td>
<td></td>
<td>0.58891</td>
<td>0.4428</td>
<td>0.0827</td>
<td>-0.0787</td>
<td></td>
</tr>
<tr>
<td>mean4</td>
<td>MEAN</td>
<td>x4</td>
<td></td>
<td>6.59009</td>
<td>0.0103</td>
<td>0.2746</td>
<td>-0.5493</td>
</tr>
<tr>
<td>2 MEAN</td>
<td>x4</td>
<td></td>
<td>0.51343</td>
<td>0.4737</td>
<td>0.0796</td>
<td>-0.1478</td>
<td></td>
</tr>
<tr>
<td>3 MEAN</td>
<td>x4</td>
<td></td>
<td>11.61610</td>
<td>0.0007</td>
<td>-0.3586</td>
<td>0.7739</td>
<td></td>
</tr>
<tr>
<td>mean5</td>
<td>MEAN</td>
<td>x5</td>
<td></td>
<td>0.52967</td>
<td>0.4667</td>
<td>-0.1042</td>
<td>0.2084</td>
</tr>
<tr>
<td>2 MEAN</td>
<td>x5</td>
<td></td>
<td>0.22294</td>
<td>0.6368</td>
<td>0.0702</td>
<td>-0.1304</td>
<td></td>
</tr>
<tr>
<td>3 MEAN</td>
<td>x5</td>
<td></td>
<td>0.06889</td>
<td>0.7930</td>
<td>0.0374</td>
<td>-0.0807</td>
<td></td>
</tr>
<tr>
<td>mean6</td>
<td>MEAN</td>
<td>x6</td>
<td></td>
<td>1.16656</td>
<td>0.2801</td>
<td>0.1270</td>
<td>-0.2540</td>
</tr>
<tr>
<td>2 MEAN</td>
<td>x6</td>
<td></td>
<td>5.29599</td>
<td>0.0214</td>
<td>-0.2810</td>
<td>0.5218</td>
<td></td>
</tr>
<tr>
<td>3 MEAN</td>
<td>x6</td>
<td></td>
<td>1.69412</td>
<td>0.1931</td>
<td>0.1518</td>
<td>-0.3275</td>
<td></td>
</tr>
<tr>
<td>mean7</td>
<td>MEAN</td>
<td>x7</td>
<td></td>
<td>0.03791</td>
<td>0.8456</td>
<td>-0.0291</td>
<td>0.0582</td>
</tr>
<tr>
<td>2 MEAN</td>
<td>x7</td>
<td></td>
<td>0.44510</td>
<td>0.5047</td>
<td>0.1036</td>
<td>-0.1923</td>
<td></td>
</tr>
<tr>
<td>3 MEAN</td>
<td>x7</td>
<td></td>
<td>0.23804</td>
<td>0.6256</td>
<td>-0.0704</td>
<td>0.1520</td>
<td></td>
</tr>
<tr>
<td>mean8</td>
<td>MEAN</td>
<td>x8</td>
<td></td>
<td>0.39420</td>
<td>0.5301</td>
<td>-0.0883</td>
<td>0.1765</td>
</tr>
<tr>
<td>2 MEAN</td>
<td>x8</td>
<td></td>
<td>0.24231</td>
<td>0.6225</td>
<td>0.0719</td>
<td>-0.1335</td>
<td></td>
</tr>
<tr>
<td>3 MEAN</td>
<td>x8</td>
<td></td>
<td>0.01951</td>
<td>0.8889</td>
<td>0.0200</td>
<td>-0.0431</td>
<td></td>
</tr>
<tr>
<td>mean9</td>
<td>MEAN</td>
<td>x9</td>
<td></td>
<td>0.00156</td>
<td>0.9685</td>
<td>0.00423</td>
<td>-0.00846</td>
</tr>
<tr>
<td>2 MEAN</td>
<td>x9</td>
<td></td>
<td>1.06866</td>
<td>0.3012</td>
<td>-0.1150</td>
<td>0.2136</td>
<td></td>
</tr>
<tr>
<td>3 MEAN</td>
<td>x9</td>
<td></td>
<td>1.05210</td>
<td>0.3050</td>
<td>0.1065</td>
<td>-0.2297</td>
<td></td>
</tr>
</tbody>
</table>
You now modify the preceding statements to specify the second revised model, as shown in the following statements:

```sas
proc calis modification;
  var x1-x9;
  group 1 / data=g1;
  group 2 / data=g2;
  group 3 / data=g3;
  model 1 / group = 1;
    mstruct;
      matrix _cov_ = cov01-cov45;
      matrix _mean_ = mean1-mean9;
    model 2 / group = 2;
      refmodel 1;
      renameparm mean2=mean2_new; /* constraint a */
    model 3 / group = 3;
      refmodel 1;
      renameparm mean2=mean2_new, /* constraint a */
      mean3=mean3_mdl3;
    fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
run;
```

This second revised model must not constrain the mean of x2 in Model 1 with any parameters. A straightforward way to do this is to rename the mean2 parameter to a unique name in Model 1. However, for the current specification it is more convenient to rename the mean2 parameter in Models 2 and 3 to another name. In the specification of the second revised model, Models 2 and 3 still make references to Model 1. However, in the respective RENAMEPARM statements, both Model 2 and 3 rename the mean2 parameter that is referenced from Model 1 to the new name mean2_new. This way the mean for x2 in Model 1 is not constrained with the means of x2 in Models 2 and 3. But the means for x2 in Models 2 and 3 are still constrained to be equal by the same parameter mean2_new. Output 29.22.12 shows the fit summary of the second revised model.

### Output 29.22.12  Fit Summary for the Second Revised Model

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>86.3927</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>106</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.9183</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0000</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>198.3927</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>374.4083</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>318.4083</td>
</tr>
</tbody>
</table>

Again, a sizable improvement over the first revised model is shown in the second revised model. The model fit chi-square statistic is no longer significant ($p = 0.9183$), and the RMSEA value is perfect at 0. Large drops in the AIC, CAIC, and SBC values are also observed.
Output 29.22.13 suggests that the mean of x6 in Model 2 (which has the largest LM statistic at 11.41243) could be released from the equality constraints to achieve the largest model improvement over the current model.

### Output 29.22.13  LM Statistics for Releasing the Equality Constraints in the Second Revised Model

<table>
<thead>
<tr>
<th>Parm</th>
<th>Model Type</th>
<th>Var1</th>
<th>Var2</th>
<th>LM Stat</th>
<th>Pr &gt; ChiSq</th>
<th>Released Parm</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>cov01</td>
<td>1 COV</td>
<td>x1</td>
<td>x1</td>
<td>2.77024</td>
<td>0.0960</td>
<td>-0.2770</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 COV</td>
<td>x1</td>
<td>x1</td>
<td>0.28728</td>
<td>0.5920</td>
<td>-0.0860</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 COV</td>
<td>x1</td>
<td>x1</td>
<td>5.00087</td>
<td>0.0253</td>
<td>-0.1791</td>
<td>0.3864</td>
</tr>
<tr>
<td>mean1</td>
<td>1 MEAN</td>
<td>x1</td>
<td></td>
<td>2.75437</td>
<td>0.0970</td>
<td>-0.3292</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x1</td>
<td></td>
<td>3.21093</td>
<td>0.0731</td>
<td>-0.2806</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x1</td>
<td></td>
<td>0.24923</td>
<td>0.6176</td>
<td>-0.0915</td>
<td></td>
</tr>
<tr>
<td>mean3</td>
<td>1 MEAN</td>
<td>x1</td>
<td></td>
<td>0.74338</td>
<td>0.3886</td>
<td>-0.0877</td>
<td>0.0934</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x1</td>
<td></td>
<td>0.74338</td>
<td>0.3886</td>
<td>-0.0877</td>
<td></td>
</tr>
<tr>
<td>mean4</td>
<td>1 MEAN</td>
<td>x1</td>
<td></td>
<td>6.17449</td>
<td>0.0130</td>
<td>0.2672</td>
<td>-0.5343</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x4</td>
<td></td>
<td>0.02087</td>
<td>0.8851</td>
<td>-0.0146</td>
<td>0.0272</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x4</td>
<td></td>
<td>4.71344</td>
<td>0.0299</td>
<td>-0.2072</td>
<td>0.4470</td>
</tr>
<tr>
<td>mean5</td>
<td>1 MEAN</td>
<td>x5</td>
<td></td>
<td>1.65517</td>
<td>0.1983</td>
<td>-0.1853</td>
<td>0.3706</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x5</td>
<td></td>
<td>1.61118</td>
<td>0.2812</td>
<td>0.1606</td>
<td>-0.2982</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x5</td>
<td></td>
<td>0.04040</td>
<td>0.8407</td>
<td>0.0287</td>
<td>-0.0618</td>
</tr>
<tr>
<td>mean6</td>
<td>1 MEAN</td>
<td>x6</td>
<td></td>
<td>5.03834</td>
<td>0.0248</td>
<td>0.2712</td>
<td>-0.5423</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x6</td>
<td></td>
<td>11.41247</td>
<td>0.0007</td>
<td>-0.4217</td>
<td>0.7831</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x6</td>
<td></td>
<td>1.51175</td>
<td>0.2189</td>
<td>0.1460</td>
<td>-0.3150</td>
</tr>
<tr>
<td>mean7</td>
<td>1 MEAN</td>
<td>x7</td>
<td></td>
<td>0.32382</td>
<td>0.5693</td>
<td>-0.0853</td>
<td>0.1706</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x7</td>
<td></td>
<td>0.82184</td>
<td>0.3646</td>
<td>0.1410</td>
<td>-0.2619</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x7</td>
<td></td>
<td>0.12513</td>
<td>0.7235</td>
<td>-0.0512</td>
<td>0.1104</td>
</tr>
<tr>
<td>mean8</td>
<td>1 MEAN</td>
<td>x8</td>
<td></td>
<td>2.39207</td>
<td>0.1220</td>
<td>-0.2210</td>
<td>0.4420</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x8</td>
<td></td>
<td>1.58292</td>
<td>0.2083</td>
<td>0.1867</td>
<td>-0.3467</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x8</td>
<td></td>
<td>0.08641</td>
<td>0.7688</td>
<td>0.0427</td>
<td>-0.0922</td>
</tr>
<tr>
<td>mean9</td>
<td>1 MEAN</td>
<td>x9</td>
<td></td>
<td>0.00682</td>
<td>0.9342</td>
<td>0.00886</td>
<td>-0.0177</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x9</td>
<td></td>
<td>1.20949</td>
<td>0.2714</td>
<td>-0.1225</td>
<td>0.2274</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x9</td>
<td></td>
<td>1.10016</td>
<td>0.2942</td>
<td>0.1089</td>
<td>-0.2349</td>
</tr>
<tr>
<td>mean2_new</td>
<td>2 MEAN</td>
<td>x2</td>
<td></td>
<td>4.47814</td>
<td>0.0343</td>
<td>0.2983</td>
<td>-0.2661</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x2</td>
<td></td>
<td>4.47814</td>
<td>0.0343</td>
<td>-0.2661</td>
<td>0.2983</td>
</tr>
</tbody>
</table>
The process of model refitting should now become familiar. You modify the previous model to release the constraint on the mean of \(x_6\) in Model 2. As a result, the third revised model is specified by the following statements:

```sas
proc calis modification;
   var x1-x9;
   group 1 / data=g1;
   group 2 / data=g2;
   group 3 / data=g3;
   model 1 / group = 1;
      mstruct;
      matrix _cov_ = cov01-cov45;
      matrix _mean_ = mean1-mean9;
   model 2 / group = 2;
      refmodel 1;
      renameparm mean2=mean2_new, /* constraint a */
               mean6=mean6_mdl2;
   model 3 / group = 3;
      refmodel 1;
      renameparm mean2=mean2_new, /* constraint a */
               mean3=mean3_mdl3;
   fitindex NoIndexType On(only)= [chisq df probchi rmsea aic caic sbc];
run;
```

The only modification from the previous specification is to rename `mean6` to `mean6_mdl2` in the `RENAMEPARM` statement of Model 2. Output 29.22.14 shows the model fit summary of the third revised model.

**Output 29.22.14** Fit Summary for the Third Revised Model

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>68.7869</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>105</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.9976</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0000</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>182.7869</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>361.9456</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>304.9456</td>
</tr>
</tbody>
</table>

The model improvement over the second revised model is still notable in the third revised model. The chi-square value drops about 20 points in the third revised model. The AIC, CAIC, and the SBC values are reduced notably, though not as impressively as with the previous improvements.
Output 29.22.15 suggests that the mean of $x_4$ in Model 1 (which has the largest LM statistic at 7.01946) could be released from the equality constraint to improve model fit further.

**Output 29.22.15**  LM Statistics for Releasing the Equality Constraints in the Third Revised Model

<table>
<thead>
<tr>
<th>Released Parameter</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parm</td>
<td>Model Type</td>
</tr>
<tr>
<td>cov01</td>
<td>1 COV</td>
</tr>
<tr>
<td></td>
<td>2 COV</td>
</tr>
<tr>
<td></td>
<td>3 COV</td>
</tr>
<tr>
<td>mean1</td>
<td>1 MEAN</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
</tr>
<tr>
<td>mean3</td>
<td>1 MEAN</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
</tr>
<tr>
<td>mean4</td>
<td>1 MEAN</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
</tr>
<tr>
<td>mean5</td>
<td>1 MEAN</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
</tr>
<tr>
<td>mean6</td>
<td>1 MEAN</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
</tr>
<tr>
<td>mean7</td>
<td>1 MEAN</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
</tr>
<tr>
<td>mean8</td>
<td>1 MEAN</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
</tr>
<tr>
<td>mean9</td>
<td>1 MEAN</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
</tr>
<tr>
<td>mean2_new</td>
<td>2 MEAN</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
</tr>
</tbody>
</table>
To make the mean parameter for \( x_4 \) in Model 1 unique, the mean parameters for \( x_4 \) in Models 2 and 3 are renamed from \( \text{mean}_4 \) to \( \text{mean}_4\_\text{new} \), as shown in the following statements:

```sas
proc calis modification;
  var x1-x9;
  group 1 / data=g1;
  group 2 / data=g2;
  group 3 / data=g3;
  model 1 / group = 1;
    mstruct;
    matrix _cov_ = cov01-cov45;
    matrix _mean_ = mean1-mean9;
  model 2 / group = 2;
    refmodel 1;
    renameparm mean2=mean2\_new, /* constraint a */
      mean4=mean4\_new, /* constraint b */
      mean6=mean6\_mdl2;
  model 3 / group = 3;
    refmodel 1;
    renameparm mean2=mean2\_new, /* constraint a */
      mean3=mean3\_mdl3,
      mean4=mean4\_new; /* constraint b */
    fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
run;
```

This forms the fourth revised model. Output 29.22.16 shows the fit summary of this revised model. Again, the chi-square, AIC, CAIC, and SBC values all show improvements, as compared with the third revised model. However, the improvements do seem to slow down. For example, the CAIC value drops from 361.95 to the current value at 358.43—a mere 3 points reduction. The SBC value drops from 304.95 to the current value at 300.43—a mere 4 points reduction. These small reductions indicate that you might soon reach the point that no more model fit improvement would be possible with additional release of parameter constraints.

**Output 29.22.16** Fit Summary for the Fourth Revised Model

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>60.1265</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>104</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.9998</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0000</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>176.1265</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>358.4283</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>300.4283</td>
</tr>
</tbody>
</table>
Output 29.22.17 suggests that the mean of $x_1$ in Model 1 (which has the largest LM statistic at 6.45785) could be released from the equality constraint to achieve the largest model improvement over the current model.

Output 29.22.17  LM Statistics for Releasing the Equality Constraints in the Fourth Revised Model

<table>
<thead>
<tr>
<th>Parm</th>
<th>Model</th>
<th>Type</th>
<th>Var1</th>
<th>Var2</th>
<th>LM Stat</th>
<th>Pr &gt; ChiSq</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>cov01</td>
<td>1 COV</td>
<td>x1</td>
<td>x1</td>
<td></td>
<td>2.60531</td>
<td>0.1065</td>
<td>-0.2751</td>
</tr>
<tr>
<td></td>
<td>2 COV</td>
<td>x1</td>
<td>x1</td>
<td></td>
<td>0.28122</td>
<td>0.5959</td>
<td>-0.0871</td>
</tr>
<tr>
<td></td>
<td>3 COV</td>
<td>x1</td>
<td>x1</td>
<td></td>
<td>4.75001</td>
<td>0.0293</td>
<td>0.3859</td>
</tr>
<tr>
<td>mean1</td>
<td>1 MEAN</td>
<td>x1</td>
<td></td>
<td></td>
<td>6.45761</td>
<td>0.0110</td>
<td>-0.5232</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x1</td>
<td></td>
<td></td>
<td>5.00991</td>
<td>0.0252</td>
<td>0.3568</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x1</td>
<td></td>
<td></td>
<td>0.05931</td>
<td>0.8076</td>
<td>-0.0452</td>
</tr>
<tr>
<td>mean3</td>
<td>1 MEAN</td>
<td>x3</td>
<td></td>
<td></td>
<td>1.53300</td>
<td>0.2157</td>
<td>0.1406</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x3</td>
<td></td>
<td></td>
<td>1.53300</td>
<td>0.2157</td>
<td>-0.1298</td>
</tr>
<tr>
<td>mean5</td>
<td>1 MEAN</td>
<td>x5</td>
<td></td>
<td></td>
<td>0.09749</td>
<td>0.7549</td>
<td>0.0913</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x5</td>
<td></td>
<td></td>
<td>0.19688</td>
<td>0.6572</td>
<td>0.1330</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x5</td>
<td></td>
<td></td>
<td>0.56583</td>
<td>0.4519</td>
<td>-0.2310</td>
</tr>
<tr>
<td>mean6</td>
<td>1 MEAN</td>
<td>x6</td>
<td></td>
<td></td>
<td>0.35800</td>
<td>0.5496</td>
<td>-0.1113</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x6</td>
<td></td>
<td></td>
<td>0.35800</td>
<td>0.5496</td>
<td>0.1141</td>
</tr>
<tr>
<td>mean7</td>
<td>1 MEAN</td>
<td>x7</td>
<td></td>
<td></td>
<td>4.53367E-6</td>
<td>0.9983</td>
<td>0.000350</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x7</td>
<td></td>
<td></td>
<td>0.96363</td>
<td>0.3263</td>
<td>-0.2920</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x7</td>
<td></td>
<td></td>
<td>1.00890</td>
<td>0.3152</td>
<td>0.3208</td>
</tr>
<tr>
<td>mean8</td>
<td>1 MEAN</td>
<td>x8</td>
<td></td>
<td></td>
<td>0.20289</td>
<td>0.6524</td>
<td>0.1351</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x8</td>
<td></td>
<td></td>
<td>0.12445</td>
<td>0.7243</td>
<td>-0.0974</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x8</td>
<td></td>
<td></td>
<td>0.00590</td>
<td>0.9388</td>
<td>-0.0237</td>
</tr>
<tr>
<td>mean9</td>
<td>1 MEAN</td>
<td>x9</td>
<td></td>
<td></td>
<td>0.05893</td>
<td>0.8082</td>
<td>0.0542</td>
</tr>
<tr>
<td></td>
<td>2 MEAN</td>
<td>x9</td>
<td></td>
<td></td>
<td>1.63723</td>
<td>0.2007</td>
<td>0.2689</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x9</td>
<td></td>
<td></td>
<td>2.44241</td>
<td>0.1181</td>
<td>-0.3565</td>
</tr>
<tr>
<td>mean2_new</td>
<td>2 MEAN</td>
<td>x2</td>
<td></td>
<td></td>
<td>3.05068</td>
<td>0.0807</td>
<td>-0.2246</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x2</td>
<td></td>
<td></td>
<td>3.05068</td>
<td>0.0807</td>
<td>0.2396</td>
</tr>
<tr>
<td>mean4_new</td>
<td>2 MEAN</td>
<td>x4</td>
<td></td>
<td></td>
<td>1.81983</td>
<td>0.1773</td>
<td>-0.2003</td>
</tr>
<tr>
<td></td>
<td>3 MEAN</td>
<td>x4</td>
<td></td>
<td></td>
<td>1.81983</td>
<td>0.1773</td>
<td>0.2306</td>
</tr>
</tbody>
</table>
Example 29.22: Testing Equality of Covariance and Mean Matrices between Independent Groups

To make the mean parameter for \( x_1 \) in Model 1 unique, the mean parameters for \( x_1 \) in Models 2 and 3 are renamed from mean1 to mean1_new, as shown in the following statements:

```bash
proc calis modification;
  var x1-x9;
  group 1 / data=g1;
  group 2 / data=g2;
  group 3 / data=g3;
  model 1 / group = 1;
    mstruct;
    matrix _cov_ = cov01-cov45;
    matrix _mean_ = mean1-mean9;
  model 2 / group = 2;
    refmodel 1;
    renameparm mean1=mean1_new, /* constraint c */
        mean2=mean2_new, /* constraint a */
        mean4=mean4_new, /* constraint b */
        mean6=mean6_mdl2;
  model 3 / group = 3;
    refmodel 1;
    renameparm mean1=mean1_new, /* constraint c */
        mean2=mean2_new, /* constraint a */
        mean3=mean3_mdl3,
        mean4=mean4_new; /* constraint b */
  fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
run;
```

This forms the fifth revised model. Output 29.22.18 shows the fit summary of the fifth revised model. Again, the chi-square, AIC, CAIC, and SBC values all show improvements, as compared with the fourth revised model. However, the improvements slow down even more. For example, the CAIC value drops from 358.43 to the current value at 356.32. The SBC value drops from 300.43 to the current value at 297.32. Because the model fit does not improve much, this is the point where you would cease to release more equality constraints for improving the model fit.

**Output 29.22.18** Fit Summary for the Fifth Revised Model

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>52.8821</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>103</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>1.0000</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0000</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>170.8821</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>356.3270</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>297.3270</td>
</tr>
</tbody>
</table>
Output 29.22.19 does not suggest the release of any equality constraints on the means, because all the p-values for the LM statistics are not significant (that is, all are greater than 0.05). Therefore, the same suggestion from examining the model fit improvements of the fifth revised model echoes here: this is the point that the “best” model for the data is found.

Output 29.22.19  LM Statistics for Releasing the Equality Constraints in the Fifth Revised Model

<table>
<thead>
<tr>
<th>Parm</th>
<th>Model Type</th>
<th>Var1</th>
<th>Var2</th>
<th>LM Stat</th>
<th>Pr &gt; ChiSq</th>
<th>Original Parm</th>
<th>Released Parm</th>
</tr>
</thead>
<tbody>
<tr>
<td>cov01</td>
<td>1 COV</td>
<td>x1</td>
<td>x1</td>
<td>4.06279</td>
<td>0.0438</td>
<td>0.1590</td>
<td>-0.3180</td>
</tr>
<tr>
<td></td>
<td>2 COV</td>
<td>x1</td>
<td>x1</td>
<td>0.48735</td>
<td>0.4851</td>
<td>0.0571</td>
<td>-0.1061</td>
</tr>
<tr>
<td></td>
<td>3 COV</td>
<td>x1</td>
<td>x1</td>
<td>7.60892</td>
<td>0.0058</td>
<td>-0.2095</td>
<td>0.4520</td>
</tr>
<tr>
<td></td>
<td>mean3</td>
<td>1 MEAN</td>
<td>x3</td>
<td>0.08363</td>
<td>0.7724</td>
<td>-0.0312</td>
<td>0.0382</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 MEAN</td>
<td>x3</td>
<td>0.08363</td>
<td>0.7724</td>
<td>0.0382</td>
<td>-0.0312</td>
</tr>
<tr>
<td></td>
<td>mean5</td>
<td>1 MEAN</td>
<td>x5</td>
<td>0.02394</td>
<td>0.8770</td>
<td>0.0229</td>
<td>-0.0458</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 MEAN</td>
<td>x5</td>
<td>0.47076</td>
<td>0.4926</td>
<td>-0.1113</td>
<td>0.2067</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 MEAN</td>
<td>x5</td>
<td>0.26015</td>
<td>0.6100</td>
<td>0.0728</td>
<td>-0.1571</td>
</tr>
<tr>
<td></td>
<td>mean6</td>
<td>1 MEAN</td>
<td>x6</td>
<td>0.97521</td>
<td>0.3234</td>
<td>0.1893</td>
<td>-0.1892</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 MEAN</td>
<td>x6</td>
<td>0.97521</td>
<td>0.3234</td>
<td>-0.1892</td>
<td>0.1893</td>
</tr>
<tr>
<td></td>
<td>mean7</td>
<td>1 MEAN</td>
<td>x7</td>
<td>0.03746</td>
<td>0.8465</td>
<td>-0.0319</td>
<td>0.0638</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 MEAN</td>
<td>x7</td>
<td>1.10428</td>
<td>0.2933</td>
<td>0.1683</td>
<td>-0.3126</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 MEAN</td>
<td>x7</td>
<td>0.79474</td>
<td>0.3727</td>
<td>-0.1321</td>
<td>0.2851</td>
</tr>
<tr>
<td></td>
<td>mean8</td>
<td>1 MEAN</td>
<td>x8</td>
<td>0.86792</td>
<td>0.3515</td>
<td>-0.1426</td>
<td>0.2852</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 MEAN</td>
<td>x8</td>
<td>0.47493</td>
<td>0.4907</td>
<td>0.1038</td>
<td>-0.1928</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 MEAN</td>
<td>x8</td>
<td>0.03722</td>
<td>0.8470</td>
<td>0.0276</td>
<td>-0.0595</td>
</tr>
<tr>
<td></td>
<td>mean9</td>
<td>1 MEAN</td>
<td>x9</td>
<td>0.12190</td>
<td>0.7270</td>
<td>0.0401</td>
<td>-0.0801</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 MEAN</td>
<td>x9</td>
<td>2.66768</td>
<td>0.1024</td>
<td>-0.1869</td>
<td>0.3472</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 MEAN</td>
<td>x9</td>
<td>1.78114</td>
<td>0.1820</td>
<td>0.1417</td>
<td>-0.3058</td>
</tr>
<tr>
<td></td>
<td>mean1_new</td>
<td>2 MEAN</td>
<td>x1</td>
<td>1.28034</td>
<td>0.2578</td>
<td>-0.1794</td>
<td>0.1359</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 MEAN</td>
<td>x1</td>
<td>1.28034</td>
<td>0.2578</td>
<td>0.1359</td>
<td>-0.1794</td>
</tr>
<tr>
<td></td>
<td>mean2_new</td>
<td>2 MEAN</td>
<td>x2</td>
<td>2.53131</td>
<td>0.1116</td>
<td>0.2117</td>
<td>-0.2112</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 MEAN</td>
<td>x2</td>
<td>2.53131</td>
<td>0.1116</td>
<td>-0.2112</td>
<td>0.2117</td>
</tr>
<tr>
<td></td>
<td>mean4_new</td>
<td>2 MEAN</td>
<td>x4</td>
<td>2.25832</td>
<td>0.1329</td>
<td>0.2558</td>
<td>-0.2253</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 MEAN</td>
<td>x4</td>
<td>2.25832</td>
<td>0.1329</td>
<td>-0.2253</td>
<td>0.2558</td>
</tr>
</tbody>
</table>
To see where the fifth revised model (equality in the covariance matrix and partial equality in the means) stands between the models under $H_0$ (equality in the covariance and mean matrices) and $H_1$ (equality in the covariance matrix only), the following table shows the fit statistics of these three models:

<table>
<thead>
<tr>
<th></th>
<th>$H_0$</th>
<th>“Fifth”</th>
<th>$H_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-square</td>
<td>203.2605</td>
<td>52.8821</td>
<td>26.7897</td>
</tr>
<tr>
<td>Chi-square DF</td>
<td>108</td>
<td>103</td>
<td>90</td>
</tr>
<tr>
<td>Pr &gt; chi-square</td>
<td>&lt;0.0001</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>RMSEA estimate</td>
<td>0.2100</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Akaike information criterion</td>
<td>311.2605</td>
<td>170.8821</td>
<td>170.7897</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>480.9898</td>
<td>356.3270</td>
<td>397.0954</td>
</tr>
<tr>
<td>Schwarz Bayesian criterion</td>
<td>426.9898</td>
<td>297.3270</td>
<td>325.0954</td>
</tr>
</tbody>
</table>

The fifth revised model is labeled “Fifth” in the table. Compared with the model under $H_0$, the fifth revised model is clearly superior. It uses only five more parameters (or five fewer degrees of freedom), but the improvement in the model fit chi-square and the RMSEA value are huge. The AIC, CAIC, and SBC are also much better.

Compared with the model under $H_1$, the fifth revised model appears to be inferior in only the chi-square model fit statistic, although both models already have the highest possible $p$-value at 1.000 and smallest possible RMSEA value at 0. However, the model under $H_1$ uses 13 more parameters (or it has 13 fewer degrees of freedom), and hence it is more complex. In fact, because the model fit chi-square value does not take model complexity into account, it is often criticized as the basis for choosing competing models for the data. In contrast, the AIC, CAIC, and SBC measures take model complexity into account, and they are more reasonable as the basis for choosing competing models. Although the AIC values for the fifth revised model and the model under $H_1$ are very close, the CAIC and SBC values clearly favor the fifth revised model. Therefore, according to the CAIC and SBC criteria, the fifth revised model, which is a model with partial equality constraints on the means, is actually better than the model with all the means being unconstrained (that is, under $H_1$) for the current data with three independent groups.
Example 29.23: Illustrating Various General Modeling Languages

In PROC CALIS, you can use many different modeling languages to specify the same model. The choice of modeling language depends on personal preferences and the purposes of the analysis. See the section “Which Modeling Language?” on page 1194 for guidance. In this example, the data and the model in Example 29.17 are used to illustrate how a particular model can be specified by various general modeling languages.

RAM Model Specification

In Example 29.17, you use the PATH modeling language to specify the model because of its close resemblance to the path diagram. In this example, you consider another modeling language of PROC CALIS that is also closely related to the path diagram representation of structural equation models. The so-called RAM model language has syntax that represents the single- and double-headed paths (or arrows) in the path diagram. However, unlike the PATH modeling language, the RAM modeling language is matrix-based. The following statements show how you can specify the same path model with the RAM model specification for the data in Example 29.17:

```plaintext
proc calis nobs=932 data=Wheaton;
   ram
   var = Anomie67     /* 1 */
        Powerless67  /* 2 */
        Anomie71     /* 3 */
        Powerless71  /* 4 */
        Education   /* 5 */
        SEI         /* 6 */
        Alien67     /* 7 */
        Alien71     /* 8 */
        SES,        /* 9 */
   A_  1  7  1.0,
   A_  2  7  0.833,
   A_  3  8  1.0,
   A_  4  8  0.833,
   A_  5  9  1.0,
   A_  6  9  lambda,
   A_  7  9  gamma1,
   A_  8  9  gamma2,
   A_  8  7  beta,
   P_  1  1  theta1,
   P_  2  2  theta2,
   P_  3  3  theta1,
   P_  4  4  theta2,
   P_  5  5  theta3,
   P_  6  6  theta4,
   P_  7  7  psi1,
   P_  8  8  psi2,
   P_  9  9  phi,
   P_  1  3  theta5,
   P_  2  4  theta5;
run;
```
In the RAM model for covariance structure analysis, you have two important matrices to specify. The first one is the _A_ matrix, which is for the specification of the single-headed paths (arrows) in the path diagram. The second one is the _P_ matrix, which is for the specification of the double-headed paths (arrows) in the path diagram. Hence, to specify the RAM model is much like mapping the path diagram arrows into the parameter of the RAM model matrices.

In the RAM statement, you can specify the variables in the model in the VAR= option. The VAR= list contains all observed and latent variables in your path diagram (without the use of error terms). Although you can specify the variables in the VAR= list in any order you like, the variable order in the list is also the order of variables in the RAM model matrices. In VAR= list of the RAM statement, you put comments to note the order of the variables.

After you specify the variable list, you can specify the model parameter locations in the RAM statement entries. In the first nine entries, you specify the single-headed paths by mapping them into the elements of the _A_ matrix of the RAM model. For example, the first entry represents the single-headed path of variable 1 (Anomie67) from variable 7 (Alien67). The corresponding path effect or coefficient is fixed at 1, which is also the value for _A_[1,7]. Another example is the ninth path entry. You specify a single-headed path of variable 8 (Alien71) from variable 7 (Alien67). The corresponding path effect or coefficient is a free parameter named beta, which is also the parameter for _A_[8,7]. Hence, you can specify all single-headed paths in the path diagram as elements in the _A_ matrix of the RAM model.

To facilitate the comparisons between the RAN and PATH modeling languages, the PATH model specification in Example 29.17 for the same data is reproduced in the following:

```plaintext
proc calis nobs=932 data=Wheaton plots=residuals;
    path
        Anomie67  Powerless67 <=== Alien67 = 1.0 0.833,
        Anomie71  Powerless71 <=== Alien71 = 1.0 0.833,
        Education SEI <=== SES = 1.0 lambda,
        Alien67  Alien71 <=== SES = gamma1 gamma2,
        Alien71 <=== Alien67 = beta;
    pvar
        Anomie67 = theta1,
        Powerless67 = theta2,
        Anomie71 = theta1,
        Powerless71 = theta2,
        Education = theta3,
        SEI = theta4,
        Alien67 = psi1,
        Alien71 = psi2,
        SES = phi;
    pcov
        Anomie67 Anomie71 = theta5,
        Powerless67 Powerless71 = theta5;
run;
```

It is clear that each of the path entries specified in the PATH statement corresponds to an matrix element entry of the _A_ matrix in the RAM statement. How about the specifications of the double-headed arrows in the path diagram? Do the RAM and PATH model specifications correspond to each other?
The answer is yes. In the PATH modeling language, you specify all double-headed arrows in the path diagram as entries either in the PVAR or PCOV statement. In the RAM modeling language, you specify the corresponding entries as matrix element entries of the _P_ matrix in the RAM statement. For example, the error variance of Anomie67 is a parameter called _Variabletheta1 in the PVAR statement of the PATH model. You specify the same parameter for the _P_[1,1] element in an entry of the RAM statement. Another example is the error covariance between Powerless67 and Powerless71. You specify this a parameter called theta5 in the last entry of the PCOV statement in the PATH model. You specify the same parameter for the _P_[2,4] element in the last entry of the RAM statement. Therefore, it is not difficult to find that the specifications in the PATH and the RAM model have some kind of one-to-one correspondence.

Output 29.23.1 shows the RAM model estimates for the Wheaton data. These RAM model estimates match the set of estimates using the PATH model specification, as shown in Output 29.17.11.

**Output 29.23.1 RAM Model Estimates**

| Matrix | Row | Column | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|--------|-----|--------|-----------|----------|----------------|---------|----------|----------------|
| _A_ (1) | Anomie67 | 1 | Alien67 | 7 | 1.00000 | | | |
| | Powerless67 | 2 | Alien67 | 7 | 0.83300 | | | |
| | Anomie71 | 3 | Alien71 | 8 | 1.00000 | | | |
| | Powerless71 | 4 | Alien71 | 8 | 0.83300 | | | |
| | Education | 5 | SES | 9 | 1.00000 | | | |
| | SEI | 6 | SES | 9 | lambda | 5.36883 | 0.43371 | 12.3788 <.0001 | |
| | Alien67 | 7 | SES | 9 | gamma1 | -0.62994 | 0.05634 | -11.1809 <.0001 | |
| | Alien71 | 8 | SES | 9 | gamma2 | -0.24086 | 0.05489 | -4.3884 <.0001 | |
| | Alien67 | 8 | Alien67 | 7 | beta | 0.59312 | 0.04678 | 12.6788 <.0001 | |
| _P_ (2) | Anomie67 | 1 | Anomie67 | 1 | theta1 | 3.60796 | 0.20092 | 17.9572 <.0001 | |
| | Powerless67 | 2 | Powerless67 | 2 | theta2 | 3.59488 | 0.16448 | 21.8556 <.0001 | |
| | Anomie71 | 3 | Anomie71 | 3 | theta1 | 3.60796 | 0.20092 | 17.9572 <.0001 | |
| | Powerless71 | 4 | Powerless71 | 4 | theta2 | 3.59488 | 0.16448 | 21.8556 <.0001 | |
| | Education | 5 | Education | 5 | theta3 | 2.99366 | 0.49861 | 6.0040 <.0001 | |
| | SEI | 6 | SEI | 6 | theta4 | 259.57639 | 18.31151 | 14.1756 <.0001 | |
| | Alien67 | 7 | Alien67 | 7 | psi1 | 5.67046 | 0.42301 | 13.4050 <.0001 | |
| | Alien71 | 8 | Alien71 | 8 | psi2 | 4.51479 | 0.33532 | 13.4639 <.0001 | |
| | SES | 9 | SES | 9 | phi | 6.61634 | 0.63914 | 10.3519 <.0001 | |
| | Anomie67 | 1 | Anomie71 | 3 | theta5 | 0.90580 | 0.12167 | 7.4447 <.0001 | |
| | Powerless67 | 2 | Powerless71 | 4 | theta5 | 0.90580 | 0.12167 | 7.4447 <.0001 | |
LINEQS Model Specification

Another way to specify the model in Example 29.17 is to use the LINEQS modeling language, which is shown in the following:

```plaintext
proc calis nobs=932 data=Wheaton;
  lineqs
    Anomie67 = 1.0 * f_Alien67 + e1,
    Powerless67 = 0.833 * f_Alien67 + e2,
    Anomie71 = 1.0 * f_Alien71 + e3,
    Powerless71 = 0.833 * f_Alien71 + e4,
    Education = 1.0 * f_SES + e5,
    SEI = lambda * f_SES + e6,
    f_Alien67 = gamma1 * f_SES + d1,
    f_Alien71 = gamma2 * f_SES + beta * f_Alien67 + d2;
  variance
    E1 = theta1,
    E2 = theta2,
    E3 = theta1,
    E4 = theta2,
    E5 = theta3,
    E6 = theta4,
    D1 = psi1,
    D2 = psi2,
    f_SES = phi;
  cov
    E1 E3 = theta5,
    E2 E4 = theta5;
run;
```

As compared with the PATH and RAM modeling languages, the most distinct feature of the LINEQS modeling language is the explicit use of error terms in equation specifications. In the LINEQS statement, you specify exactly one equation for each endogenous variable. In each equation, you list an endogenous variable on the left-hand-side of the equation and all its predictors on the right-hand-side of the equation. You must also include an error term in each equation. Because each endogenous variable in the LINEQS statement can only be specified in exactly one equation, the number of equations in the LINEQS model and the number of paths in the corresponding path diagram do not match necessarily. In this example, there are eight equations in the LINEQS statement, but there are nine paths in the corresponding path diagram.

In addition, in the LINEQS model, you need to follow a convention of naming latent variables. For latent variables that are neither errors nor disturbances, you must use either the ‘F’ or ‘f’ prefix. For error terms, you must use either the ‘E’ or ‘e’ prefix. For disturbances, you must use either the ‘D’ or ‘d’ prefix. However, in the PATH or RAM model specification, no such convention is imposed. For example, f_Alien67, f_Alien71, and f_SES are latent factors in the LINEQS model. They are not error terms, and so they must start with the ‘f’ prefix. However, this prefix is not needed in the PATH or RAM model. Furthermore, there are no explicit error terms that need to be specified in the PATH or RAM model, let alone specific prefixes for the error terms.
The **PVAR** statement in the PATH model is replaced with the **VARIANCE** statement in the LINEQS model, and the **PCOV** statement with the **COV** statement. The **PVAR** and **PCOV** statements in the PATH model are for the partial variance and partial covariance specifications. The partial variance or covariance concepts are used in the PATH or RAM model specification because error terms are not named explicitly. Specification of error variances in the PATH and RAM model is conceptualized as the specification of the partial variances of the corresponding variables. But in the LINEQS model, because errors or disturbances are named explicitly as *exogenous* variables, the partial variance or covariance concepts are no longer necessary. Instead, you specify the variances of the error terms directly, which reflects the conceptualization behind the **VARIANCE** statement of the LINEQS modeling language. Similarly, you use the **COV**, but not **PCOV**, statement in the LINEQS modeling language because you can specify the covariances among variables or error terms without using the partial covariance conceptualization.

In this example, the variances of the errors (“E”-variables) and disturbances (“D”-variables) specified in the **VARIANCE** statement of the LINEQS model correspond to the partial variances of the endogenous variables specified in the **PVAR** statement of the PATH model. Similarly, covariances of errors specified in the **COV** statement of the LINEQS model correspond to the partial covariances of endogenous variables specified in the **PCOV** statement of the PATH model. The estimation results of the LINEQS model are shown in **Output 29.23.2**. Again, they are essentially the same estimates obtained from the PATH model specified in Example 29.17, as shown in **Output 29.23.11**.

**Output 29.23.2** LINEQS Model Estimates

<table>
<thead>
<tr>
<th>Linear Equations</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67</td>
<td>1.0000</td>
<td>f_Alien67 + 1.0000 e1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Powerless67</td>
<td>0.8330</td>
<td>f_Alien67 + 1.0000 e2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anomie71</td>
<td>1.0000</td>
<td>f_Alien71 + 1.0000 e3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Powerless71</td>
<td>0.8330</td>
<td>f_Alien71 + 1.0000 e4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Education</td>
<td>1.0000</td>
<td>f_SES + 1.0000 e5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEI</td>
<td>5.3688 (***)</td>
<td>f_SES + 1.0000 e6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f_Alien67</td>
<td>-0.6299 (***)</td>
<td>f_SES + 1.0000 d1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f_Alien71</td>
<td>-0.2409 (***)</td>
<td>f_SES + 0.5931 (**) f_Alien67 + 1.0000 d2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Effects in Linear Equations**

| Variable    | Predictor | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|-------------|-----------|-----------|----------|----------------|---------|------|---|
| Anomie67    | f_Alien67 |           | 1.0000   |                |         |      |   |
| Powerless67 | f_Alien67 |           | 0.8330   |                |         |      |   |
| Anomie71    | f_Alien71 |           | 1.0000   |                |         |      |   |
| Powerless71 | f_Alien71 |           | 0.8330   |                |         |      |   |
| Education   | f_SES     |           | 1.0000   |                |         |      |   |
| SEI         | f_SES     | lambda    | 5.36883  | 0.43371        | 12.3788 | <.0001|   |
| f_Alien67   | f_SES     | gamma1    | -0.62994 | 0.05634        | -11.1809| <.0001|   |
| f_Alien71   | f_SES     | gamma2    | -0.24086 | 0.05489        | -4.3884 | <.0001|   |
| f_Alien71   | f_Alien67 | beta      | 0.59312  | 0.04678        | 12.6788 | <.0001|   |
**LISMOD Specification**

You can also specify general structural models by using the LISMOD modeling language. See the section “The LISMOD Model and Submodels” on page 1408 for details.

To use the LISMOD modeling language, you must recognize four types of variables in the model. The $\eta$-variables (eta-variables) are latent factors that are endogenous, or predicted by other latent factors. The $\xi$-variables (xi-variables) are exogenous latent variables that are not predicted by any other variables. The $y$-variables are manifest variables that are indicators of the $\eta$-variables, and the $x$-variables are manifest variables that are indicators of the $\xi$-variables. In this example, Alien67 and Alien71 are the $\eta$-variables, and SES is the $\xi$-variable in the model. Manifest indicators for Alien67 and Alien71 include Anomie67, Powerless67, Anomie71, and Powerless71, which are the $y$-variables. Manifest indicators for SES include Education and SEI, which are the $x$-variables.

After defining these four types of variables, the parameters of the model are defined as entries in the model matrices. The _LAMBDAY_, _LAMDBAX_, _GAMMA_, and _BETA_ are matrices for the path coefficients or effects. The _THETAY_, _THETAX_, _PSI_, and _PHI_ are matrices for the variances and covariances.

### Estimates for Variances of Exogenous Variables

| Variable Type | Variable Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|---------------|--------------------|----------|----------------|---------|-------|---|
| Error         | e1 theta1          | 3.60796  | 0.20092        | 17.9572 | <.0001|
|               | e2 theta2          | 3.59488  | 0.16448        | 21.8556 | <.0001|
|               | e3 theta1          | 3.60796  | 0.20092        | 17.9572 | <.0001|
|               | e4 theta2          | 3.59488  | 0.16448        | 21.8556 | <.0001|
|               | e5 theta3          | 2.99366  | 0.49861        | 6.0040  | <.0001|
|               | e6 theta4          | 259.57639| 18.31151       | 14.1756 | <.0001|
| Disturbance   | d1 psi1            | 5.67046  | 0.42301        | 13.4050 | <.0001|
|               | d2 psi2            | 4.51479  | 0.33532        | 13.4639 | <.0001|
| Latent        | f_SES phi          | 6.61634  | 0.63914        | 10.3519 | <.0001|

### Covariances Among Exogenous Variables

| Var1 | Var2 | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|------|------|-----------|----------|----------------|---------|-------|---|
| e1   | e3   | theta5    | 0.90580  | 0.12167        | 7.4447  | <.0001|
| e2   | e4   | theta5    | 0.90580  | 0.12167        | 7.4447  | <.0001|
The following is the LISMOD specification for the model in Example 29.17:

```plaintext
proc calis nobs=932 data=Wheaton;
  lismod
    yvar = Anomie67 Powerless67 Anomie71 Powerless71,
    xvar = Education SEI,
    etavar = Alien67 Alien71,
    xivar = SES;
  matrix _LAMBDAY_
    [1,1] = 1,
    [2,1] = 0.833,
    [3,2] = 1,
    [4,2] = 0.833;
  matrix _LAMBDAX_
    [1,1] = 1,
    [2,1] = lambda;
  matrix _GAMMA_
    [1,1] = gamma1,
    [2,1] = gamma2;
  matrix _BETA_
    [2,1] = beta;
  matrix _THETAY_
    [1,1] = theta1-theta2 theta1-theta2,
    [3,1] = theta5,
    [4,2] = theta5;
  matrix _THETAX_
    [1,1] = theta3-theta4;
  matrix _PSI_
    [1,1] = psi1-psi2;
  matrix _PHI_
    [1,1] = phi;
run;
```

In the LISMOD statement, you specify the four lists of variables in the model. The orders of the variables in these lists define the order of the row and column variables in the model matrices, of which the parameter locations are specified in the MATRIX statements.

The estimated model is divided into three conceptual parts. The first part is the measurement model that relates the δ-variables with the y-variables, as shown in Output 29.23.3:
Output 29.23.3 LISMOD Model Measurement Model for the $\eta$-Variables

<table>
<thead>
<tr>
<th><em>LAMBDAY</em> Matrix: Estimate/StdErr/t-value/p-value</th>
<th>Alien67</th>
<th>Alien71</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67</td>
<td>1.0000</td>
<td>0</td>
</tr>
<tr>
<td>Powerless67</td>
<td>0.8330</td>
<td>0</td>
</tr>
<tr>
<td>Anomie71</td>
<td>0</td>
<td>1.0000</td>
</tr>
<tr>
<td>Powerless71</td>
<td>0</td>
<td>0.8330</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><em>THETAY</em> Matrix: Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Anomie67</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Powerless67</td>
</tr>
<tr>
<td>0.1645</td>
</tr>
<tr>
<td>21.8556</td>
</tr>
<tr>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[theta2]</td>
</tr>
<tr>
<td>Anomie71</td>
</tr>
<tr>
<td>0.9058</td>
</tr>
<tr>
<td>0.1217</td>
</tr>
<tr>
<td>7.4447</td>
</tr>
<tr>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[theta5]</td>
</tr>
<tr>
<td>Powerless71</td>
</tr>
<tr>
<td>0.9058</td>
</tr>
<tr>
<td>0.1217</td>
</tr>
<tr>
<td>7.4447</td>
</tr>
<tr>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[theta5]</td>
</tr>
</tbody>
</table>

The _LAMBDAY_ matrix contains the coefficients or effects of the $\eta$-variables on the $y$-variables. All these estimates are fixed constants as specified. The _THETAY_ matrix contains the error variances and covariances for the $y$-variables. Three free parameters are located in this matrix: theta1, theta2, and theta5.

The second part of the estimated model is the measurement model that relates the $\xi$-variable with the $x$-variables, as shown in Output 29.23.4:
Output 29.23.4 LISMOD Model Measurement Model for the ξ-Variables

<table>
<thead>
<tr>
<th>LAMBDA Matrix:</th>
<th>Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SES</td>
<td></td>
</tr>
<tr>
<td>Education</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>THETA Matrix:</th>
<th>Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Education</td>
<td></td>
</tr>
<tr>
<td>2.9937</td>
<td>0</td>
</tr>
<tr>
<td>0.4986</td>
<td>6.0040</td>
</tr>
<tr>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td>SEI</td>
<td></td>
</tr>
<tr>
<td>5.3688</td>
<td>0.4337</td>
</tr>
<tr>
<td>12.3788</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[lambda]</td>
<td></td>
</tr>
<tr>
<td>SEI</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>259.5764</td>
</tr>
<tr>
<td>18.3115</td>
<td>14.1756</td>
</tr>
<tr>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td>[theta3]</td>
<td></td>
</tr>
</tbody>
</table>

The _LAMBDA_ matrix contains the coefficients or effects of the ξ-variable SES on the x-variables. The effect of SES on Education is fixed at one. The effect of SES on SEI is represented by the free parameter lambda, which is estimated at 5.3688. The _THETA_ matrix contains the error variances and covariances for the x-variables. Two free parameters are located in this matrix: theta3 and theta4.

The last part of the estimated model is the structural model that relates the latent variables η and ξ, as shown in Output 29.23.5:

Output 29.23.5 LISMOD Structural Model for the Latent Variables

<table>
<thead>
<tr>
<th>BETA Matrix:</th>
<th>Estimate/StdErr/t-value/p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alien67</td>
<td>Alien71</td>
</tr>
<tr>
<td>Alien67</td>
<td>0</td>
</tr>
<tr>
<td>Alien71</td>
<td>0</td>
</tr>
<tr>
<td>0.5931</td>
<td>0</td>
</tr>
<tr>
<td>0.0468</td>
<td></td>
</tr>
<tr>
<td>12.6788</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>[beta]</td>
<td></td>
</tr>
</tbody>
</table>
The _BETA_ matrix contains effects of \( \eta \)-variables on themselves. In the current example, there is only one such effect. The effect of Alien67 on Alien71 is represented by the free parameter beta. The _GAMMA_ matrix contains effects of the \( \xi \)-variable, which is SES in this example, on the \( \eta \)-variables Alien67 on Alien71. These effects are represented by the free parameters gamma1 and gamma2. The _PSI_ matrix contains the error variances and covariances in the structural model. In this example, psi1 and psi2 are two free parameters for the error variances. Finally, the _PHI_ matrix is the covariance matrix for the \( \xi \)-variables. In this example, there is only one \( \xi \)-variable so that this matrix contains only the estimated variance of SES. This variance is represented by the parameter phi.

The estimates obtained from fitting the LISMOD model are the same as those from fitting the equivalent PATH, RAM, or LINEQS model. To some researchers the LISMOD modeling language might be more familiar, while for others modeling languages such as PATH, RAM, or LINEQS are more convenient to use.
Example 29.24: Testing Competing Path Models for the Career Aspiration Data

This example uses some well-known data from Haller and Butterworth (1960). The section “A Combined Measurement-Structural Model” on page 325 in Chapter 17, “Introduction to Structural Equation Modeling with Latent Variables,” analyzes some models for these data. Inspired by the examples given in Loehlin (1987), this example shows additional applications to the same data set, but with a focus on testing nested models. By manipulating the OUTMODEL= data set, this example shows how you can specify new models in an efficient way. Various models and analyses of these data are also given by Duncan, Haller, and Portes (1968), Jöreskog and Sörbom (1988), and Loehlin (1987).

The study is concerned with the career aspirations of high school students and how these aspirations are affected by close friends. The data are collected from 442 seventeen-year-old boys in Michigan. There are 329 boys in the sample who named another boy in the sample as a best friend. The data from these 329 boys paired with the data from their best friends are analyzed.

Because of the dependency of the data, the effective sample size assumed in the example is 329, which you can specify in the NOBS= option in the PROC CALIS statements. See the section “A Combined Measurement-Structural Model” on page 325 in Chapter 17, “Introduction to Structural Equation Modeling with Latent Variables,” for the justification of the use of this effective sample size.

The correlation matrix, taken from Jöreskog and Sörbom (1988), is shown in the following DATA step:

```plaintext
title 'Peer Influences on Aspiration: Haller & Butterworth (1960)';
data aspire(type=corr);
  _type_='corr';
  input _name_ $ riq rpa rses roa rea fiq fpa fses foa fea;
  label riq='Respondent: Intelligence'
    rpa='Respondent: Parental Aspiration'
    rses='Respondent: Family SES'
    roa='Respondent: Occupational Aspiration'
    rea='Respondent: Educational Aspiration'
    fiq='Friend: Intelligence'
    fpa='Friend: Parental Aspiration'
    fses='Friend: Family SES'
    foa='Friend: Occupational Aspiration'
    fea='Friend: Educational Aspiration';
datalines;
  riq 1. . . . . . . . . .
  rpa .1839 1. . . . . . . . .
  rses .2220 .0489 1. . . . . . . .
  roa .4105 .2137 .3240 1. . . . . . . .
  rea .4043 .2742 .4047 .6247 1. . . . . . . .
  fiq .3355 .0782 .2302 .2995 .2863 1. . . . . . . .
  fpa .1021 .1147 .0931 .0760 .0702 .2087 1. . . . . . . .
  fses .1861 .0186 .2707 .2930 .2407 .2950 -.0438 1. . . . . . . .
  foa .2598 .0839 .2786 .4216 .3275 .5007 .1988 .3607 1. . . . . . . .
  fea .2903 .1124 .3054 .3269 .3669 .5191 .2784 .4105 .6404 1. . . . . . . .
;```
For illustration purposes, this correlation matrix is treated here as if it were a covariance matrix for PROC CALIS to analyze. The reason is that the chi-square tests shown in this example are valid only with covariance structure analysis. See Example 29.27 for an illustration of covariance structure analysis on correlations.

**Model 1: The Full Model**

Loehlin (1987) analyzes the following path model for the data:

![Path Diagram for Career Aspiration: Model 1](image)

In Figure 29.24, the observed variables rpa, riq, rses, fses, fiq, and fpa are measured with errors. Their true scores counterparts f_rpa, f_riq, f_rses, f_fses, f_fiq, and f_fpa are latent variables in the model. Path coefficients from these latent variables to the observed variables are fixed coefficients, indicating the square roots of the theoretical reliabilities in the model. These latent variables, rather than the observed counterparts, serve as predictors of the ambition factors R_Amb and F_Amb. The error terms for these two latent factors are correlated, as indicated by a double-headed path (arrow) that connects the two factors. Correlated errors for the occupational aspiration variables (roa and foa) and the educational aspiration variables (rea and fea) are also shown in Figure 29.24. These correlated errors are also represented by two double-headed paths (arrows) in the path diagram.

Notice that the covariances among the six exogenous latent variables (f_rpa, f_riq, f_rses, f_fses, f_fiq, and f_fpa) are not represented in the path diagram for two reasons. First, there are 15 of these covariances and hence you need 15 double-headed arrows to represent them in the path diagram. Apparently, because of the space limitations, it would be difficult to put all these double-headed arrows in the path diagram without cluttering it. Second, covariances among exogenous latent variables are free parameters by default in
PROC CALIS, and therefore omitting these double-headed arrows in the path diagram is compatible with the default model specification in PROC CALIS. Similarly, double-headed arrows for the error variances of the endogenous variables (rpa, riq, rses, fses, fiq, fpa, R_Amb, and F_Amb) in the path diagram are omitted because they are unconstrained free parameters and are set automatically by default in PROC CALIS.

The model represented by the path diagram in Figure 29.24 is considered to be the full model for the data, in the sense that it has the largest number of parameters among the competing models considered this example. The same model is analyzed in the section “A Combined Measurement-Structural Model” on page 325 in Chapter 17, “Introduction to Structural Equation Modeling with Latent Variables,” with the following specification:

```
proc calis data=aspire nobs=329;
  path
    /* measurement model for intelligence and environment */
    rpa <= f_rpa = 0.837,
    riq <= f_riq = 0.894,
    rses <= f_rses = 0.949,
    fses <= f_fses = 0.949,
    fiq <= f_fiq = 0.894,
    fpa <= f_fpa = 0.837,
    /* structural model of influences: 5 equality constraints */
    f_rpa ===> R_Amb ,
    f_riq ===> R_Amb ,
    f_rses ===> R_Amb ,
    f_fses ===> R_Amb ,
    f_fiq ===> F_Amb ,
    f_fpa ===> F_Amb ,
    F_Amb ===> R_Amb ,
    R_Amb ===> F_Amb ,
    /* measurement model for aspiration: 1 equality constraint */
    R_Amb ===> rea ,
    R_Amb ===> roa = 1.,
    F_Amb ===> foa = 1.,
    F_Amb ===> fea ;
  pvar
    f_rpa f_riq f_rses f_fpa f_fiq f_fses = 6 * 1.0;
  pcov
    R_Amb F_Amb ,
    rea fea ,
    roa foa ;
run;
```

The PATH model specification represents each arrow (single-headed and double-headed) in the path diagram. You transcribe each arrow in Figure 29.24 into an entry in the PATH model. The PATH statement specifies all the single-headed arrows in the path diagram. The PVAR statement specifies all the double-headed arrows that point to individual variables (that is, the fixed error variances of the exogenous latent variables) in the path diagram. The PCOV statement specifies all the double-headed arrows that connect paired variables (that is, the error covariances) in the path diagram.
Output 29.24.1 shows the fit summary of Model 1.

Output 29.24.1  Career Aspiration Data: Fit Summary of Model 1

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>12.0132</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>13</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.5266</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0149</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0000</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>96.0132</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>297.4476</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>255.4476</td>
</tr>
</tbody>
</table>

Since the p-value for the chi-square test is 0.5266, this model clearly cannot be rejected. Both standardized RMR and RMSEA are very small. All these point to an excellent model fit. Three information-theoretic fit indices are also shown: Akaike’s information criterion (AIC), Bozdogan’s CAIC, and Schwarz’s Bayesian Criterion (SBC). These indices are useful when you need to compare competing models for the data.

Model 2: The Model with Equality Constraints

You now consider a much more restrictive model with equality constraints in the model. The path diagram for this constrained model is shown in Figure 29.25.

Figure 29.25  Path Diagram for Career Aspiration: Model 2
The main idea about setting the equality constraints in this model is that there is some symmetry in the model components that correspond to the respondent and his friend. In particular, the corresponding coefficients or parameters should be equal. For example, the path $f_{rpa} \rightarrow R_{Amb}$ for the respondent has the same effect as that of $f_{fpa} \rightarrow F_{Amb}$. In the path diagram, they are both labeled by the same parameter $\gamma_1$. Generalizing the same idea to other pairs of paths, Output 29.25 shows nine pairs of these equality constraints, which are all represented by the same parameter names for distinct (single-headed or double-headed) paths.

However, because of the space limitation, there are six more equality constraints that are not shown in the path diagram. These six constraints concern the covariance structures of the exogenous latent factors $f_{rpa}$, $f_{riq}$, $f_{rses}$, $f_{fses}$, $f_{fiq}$, and $f_{fpa}$. The first three factors are for the respondent, and the last three are for his friend. Using the same symmetry argument, the covariance structures imposed on these exogenous latent factors are shown in the following:

\[
\begin{array}{cccccc}
  f_{rpa} & f_{riq} & f_{rses} & f_{fpa} & f_{fiq} & f_{fses} \\
  f_{rpa} & 1. \\
  f_{riq} & c1 & 1. \\
  f_{rses} & c2 & c3 & 1. \\
  f_{fpa} & c4 & c5 & c6 & 1. \\
  f_{fiq} & c5 & c7 & c8 & c1 & 1. \\
  f_{fses} & c6 & c8 & c9 & c2 & c3 & 1. \\
\end{array}
\]

In this pattern of covariance structures, the covariance matrix (upper left portion) for the latent factors of the respondent is the same as that (lower right portion) for the latent factors of his friend. The cross-covariances among the factors between the friends (lower left portion) also display a symmetry pattern. There are six pairs of equality constraints in the covariance structures. Imposing these six pairs of equality constraints and the nine pairs of equality constraints in the path diagram lead to Model 2 of Loehlin (1987).
Example 29.24: Testing Competing Path Models for the Career Aspiration Data

You can specify the current constrained model by the following PATH modeling language of PROC CALIS:

```sas
proc calis data=aspire nobs=329 outmodel=model2;
p path
   /* measurement model for intelligence and environment */
   rpa <=== f_rpa = 0.837,
   riq <=== f_riq = 0.894,
   rses <=== f_rses = 0.949,
   fses <=== f_fses = 0.949,
   fiq <=== f_fiq = 0.894,
   fpa <=== f_fpa = 0.837,

   /* structural model of influences: 5 equality constraints */
   f_rpa ===> R_Amb = gam1,
   f_riq ===> R_Amb = gam2,
   f_rses ===> R_Amb = gam3,
   f_fses ===> R_Amb = gam4,
   f_fiq ===> F_Amb = gam2,
   f_fpa ===> F_Amb = gam1,
   F_Amb ===> R_Amb = beta,
   R_Amb ===> F_Amb = beta,

   /* measurement model for aspiration: 1 equality constraint */
   R_Amb ===> rea = lambda,
   R_Amb ===> roa = 1.,
   F_Amb ===> foa = 1.,
   F_Amb ===> fea = lambda;
p vvar
   f_rpa f_riq f_rses f_fpa f_fiq f_fses = 6 * 1.0,
   R_Amb F_Amb = 2 * psi, /* 1 ec */
   rea fea = 2 * theta1, /* 1 ec */
   roa foa = 2 * theta2; /* 1 ec */
p cov
   R_Amb F_Amb = psi12,
   rea fea = covea,
   roa foa = cova,
   f_rpa f_riq f_rses = cov1-cov3, /* 3 ec */
   f_fpa f_fiq f_fses = cov1-cov3,
   f_rpa f_riq f_rses * f_fpa f_fiq f_fses = /* 3 ec */
      cov4 cov5 cov6 cov5 cov7 cov8 cov6 cov8 cov9;
run;
```
In the current PATH model specification, you specify the same set of paths as in Model 1. In addition, to set the required constraints in this path model, you use parameter names to label the related paths, variances, or covariances. Same parameter names mean equality constraints. The 15 equality constraints are labeled with comments in the specification. In the PROC CALIS statement, you use the OUTMODEL= option to output the model estimation results into the output data set model2, which is used for subsequent hypotheses tests.

Output 29.24.2 shows the fit summary of Model 2.

**Output 29.24.2** Career Aspiration Data: Fit Summary of Model 2 in Loehlin (1987)

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>19.0697</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>28</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.8960</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0276</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0000</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>73.0697</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>202.5632</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>175.5632</td>
</tr>
</tbody>
</table>

The test of Model 2 against Model 1 (Loehlin 1987) yields a chi-square of 19.0697 – 12.0132 = 7.0565 with 15 degrees of freedom, which is clearly not significant. This indicates that the restricted Model 2 fits at least as well as Model 1. Schwarz’s Bayesian criterion (SBC) is also much lower for Model 2 (175.5623) than for Model 1 (255.4476). Hence, Model 2 seems preferable on both substantive and statistical grounds.

**Model 3: No SES Paths**

A question of substantive interest is whether the friend’s socioeconomic status (SES) has a significant direct influence on a boy’s ambition. This can be addressed by omitting the paths from f_fses to R_Amb and from f_rses to F_Amb designated by the parameter name gam4, yielding Model 3 of Loehlin (1987). The corresponding path diagram is shown in Figure 29.26.
In Figure 29.26, you drop the paths \( f_{\text{rses}} \rightarrow F_{\text{Amb}} \) and \( f_{\text{fses}} \rightarrow R_{\text{Amb}} \) from the previous model. Using the path diagram in Figure 29.26, you can specify the current model the same way you do for Model 2. However, because you have the estimation results from Model 2 in the SAS data set model2, you can modify this SAS data set to reflect the current model specification and then input the modified SAS data set as an INMODEL= file for PROC CALIS to analyze.

First, you create a new SAS data set model3 by the following DATA step:

```sas
data model3(type=calismdl);
  set model2;
  if _name_='gam4' then do;
    _name_=' ';
    _estim_=0;
  end;
run;
```

Essentially, by blanking out the parameter name for the target paths, you are stating that these paths are no longer associated with the free parameter gam4 in the new model. Instead, you put a fixed zero to these paths. This way you eliminate the paths \( f_{\text{rses}} \rightarrow F_{\text{Amb}} \) and \( f_{\text{fses}} \rightarrow R_{\text{Amb}} \) for Model 3, of which the model specification is now saved in the model3 data set.
Next, you input model3 as the INMODEL= data set for PROC CALIS to analyze, as shown in the following statements:

```r
proc calis data=aspire nobs=329 inmodel=model3;
run;
```

PROC CALIS can now use the previous estimation results for fitting the required model. **Output 29.24.3** shows the fit summary of Model 3.

**Output 29.24.3** Career Aspiration Data: Fit Summary of Model 3 in Loehlin (1987)

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>23.0365</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>29</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.7749</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0304</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0000</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>75.0365</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>199.7340</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>173.7340</td>
</tr>
</tbody>
</table>

The chi-square value for testing Model 3 versus Model 2 is 23.0365 – 19.0697 = 3.9668 with one degree of freedom and a \( p \)-value of 0.0464. The chi-square test shows a marginal significance, which means that the paths might be needed in the model. However, the SBC (173.7340) indicates that Model 3 is slightly preferable to Model 2, which has an SBC value of 175.5632.

**Model 4: No Reciprocal Influence between the Ambition Factors**

Another important question is whether the reciprocal influences between the respondent’s and friend’s ambitions are needed in the model. To test whether these paths are zero, you can set the parameter beta for the paths linking R_Amb and F_Amb to zero to obtain Model 4 of Loehlin (1987).

Similar to Model 3, you can modify the model2 data set to form the new model data set model4 for PROC CALIS to analyze, as shown in the following statements:

```r
data model4(type=calismdl);
set model2;
if _name_='beta' then
do;
    _name_='';
    _estim_=0;
end;
run;

proc calis data=aspire nobs=329 inmodel=model4;
run;
```

**Output 29.24.4** shows the fit summary of Model 4.
Example 29.24: Testing Competing Path Models for the Career Aspiration Data

Output 29.24.4 Career Aspiration Data: Fit Summary of Model 4 in Loehlin (1987)

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
</tr>
</tbody>
</table>

The chi-square value for testing Model 4 versus Model 2 is $20.9981 - 19.0697 = 1.9284$ with one degree of freedom and a $p$-value of 0.1649. Hence, there is little evidence of reciprocal influence.

Model 5: No Disturbance Correlation between the Ambition Factors

Model 2 of Loehlin (1987) has the direct paths connecting the latent ambition factors $R_{Amb}$ and $F_{Amb}$ and a covariance between the disturbance or error terms (that is, a double-headed arrow connecting the two factors in the path diagram shown in Figure 29.25). The presence of this disturbance correlation serves as a “wastebasket” that enables other omitted variables to have joint influences on the respondent’s and his friend’s ambition factors. To test the hypothesis that this disturbance correlation is zero, you use the following statements to set the parameter $psi12$ to zero in the model5 data set and fit the new model by PROC CALIS:

```sas
data model5(type=calismdl);
  set model2;
  if _name_='psi12' then
    do;
      _name_=' ';
      _estim_=0;
    end;
  run;

proc calis data=aspire nobls=329 inmodel=model5;
run;
```

Output 29.24.5 displays the fit summary of Model 5.

Output 29.24.5 Career Aspiration Data: Fit Summary of Model 5 in Loehlin (1987)

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
</tr>
</tbody>
</table>
The chi-square value for testing Model 5 versus Model 2 is \(19.0745 - 19.0697 = 0.0048\) with one degree of freedom. This test statistic is insignificant. Therefore, omitting the covariance between the disturbance terms causes hardly any deterioration in the fit of the model.

**Model 7: No Reciprocal Influence and No Disturbance Correlation between the Ambition Factors**

The test in Model 4 fails to provide evidence of a direct reciprocal influence between the respondent’s and friend’s ambitions, and the test in Model 5 fails to provide evidence of a covariance or correlation between the disturbance terms for the ambition factors. Because you consider these two tests separately, you cannot establish evidence to eliminate the reciprocal influence and the disturbance correlation jointly. Instead, to make such a joint inference, it is important to test both hypotheses together by setting both \(\beta\) and \(\psi_{12}\) to zero as in Model 7 of Loehlin (1987). The following statements show how you can do that by modifying the model2 data set to form a new INMODEL= data set for PROC CALIS to analyze:

```sas
data model7(type=calismdl);
  set model2;
  if _name_='psi12'|_name_='beta' then
    do;
      _name_=' ';
      _estim_=0;
    end;
  run;

proc calis data=aspire nobs=329 inmodel=model7;
run;
```

Output 29.24.6 shows the fit summary of Model 7.

**Output 29.24.6** Career Aspiration Data: Fit Summary of Model 7 in Loehlin (1987)

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>25.3466</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>30</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square (SRMR)</td>
<td>0.7080</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0363</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0000</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>75.3466</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>195.2480</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>170.2480</td>
</tr>
</tbody>
</table>

When Model 7 is tested against Models 2, 4, and 5, the \(p\)-values are respectively 0.0433, 0.0370, and 0.0123, indicating that the combined effect of the reciprocal influence and the covariance of the disturbance terms is statistically significant. Thus, the hypothesis tests indicate that it is acceptable to omit either the reciprocal influences or the covariance of the disturbances, but not both.

**Model 6: No Error Correlations between the Friend’s Educational and Occupational Aspiration**

It is also of interest to test the covariances (\(covea\) and \(covo\)) between the error terms for educational aspiration (that is, between \(rea\) and \(fe\)) and occupational aspiration (that is, between \(roa\) and \(fo\)), because
Example 29.24: Testing Competing Path Models for the Career Aspiration Data

these terms are omitted from Jöreskog and Sörbom (1988) models. Constraining covea and covoa to zero produces Model 6 of Loehlin (1987). You can use the following statements to fit this model:

```sas
data model6(type=calismdl);
set model2;
  if _name_='covea'|_name_='covoa' then
    _name_=' ';
    _estim_=0;
  end;
run;

proc calis data=aspire nobs=329 inmodel=model6;
run;
```

Output 29.24.7 shows the fit summary of Model 6.

**Output 29.24.7** Career Aspiration Data: Model 6 of Loehlin (1987)

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
</tr>
</tbody>
</table>

The chi-square value for testing Model 6 versus Model 2 is 33.4476 – 19.0697 = 14.3779 with two degrees of freedom and a \( p \)-value of 0.0008, indicating that there is considerable evidence of correlation between the error terms.

**Summary of Competing Models**

The following table summarizes the results from the seven models described in Loehlin (1987).

<table>
<thead>
<tr>
<th>Model</th>
<th>( \chi^2 )</th>
<th>df</th>
<th>( p )-value</th>
<th>SBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Full model</td>
<td>12.0132</td>
<td>13</td>
<td>0.5266</td>
<td>255.4476</td>
</tr>
<tr>
<td>2. Equality constraints</td>
<td>19.0697</td>
<td>28</td>
<td>0.8960</td>
<td>175.5632</td>
</tr>
<tr>
<td>3. No SES path</td>
<td>23.0365</td>
<td>29</td>
<td>0.7749</td>
<td>173.7340</td>
</tr>
<tr>
<td>4. No reciprocal influence</td>
<td>20.9981</td>
<td>29</td>
<td>0.8592</td>
<td>171.6956</td>
</tr>
<tr>
<td>5. No disturbance correlation</td>
<td>19.0745</td>
<td>29</td>
<td>0.9194</td>
<td>169.7721</td>
</tr>
<tr>
<td>6. No error correlation</td>
<td>33.4475</td>
<td>30</td>
<td>0.3035</td>
<td>178.3489</td>
</tr>
<tr>
<td>7. Constraints from both 4 and 5</td>
<td>25.3466</td>
<td>30</td>
<td>0.7080</td>
<td>170.2480</td>
</tr>
</tbody>
</table>

For comparing models, you can use a DATA step to compute the differences of the chi-square statistics and \( p \)-values, as shown in the following statements:
data _null_;  
array achisq[7] _temporary_  
   (12.0132 19.0697 23.0365 20.9981 19.0745 33.4475 25.3466);  
array adf[7] _temporary_  
   (13 28 29 29 29 30 30);  
retain indent 16;  
file print;  
input ho ha @@;  
chisq = achisq[ho] - achisq[ha];  
df = adf[ho] - adf[ha];  
p = 1 - probchi( chisq, df);  
if _n_ = 1 then put  
   / +indent 'model comparison chi**2 df p-value'  
   / +indent '---------------------------------------';  
put +indent +3 ho ' versus ' ha @18 +indent chisq 8.4 df 5. p 9.4;  
datalines;  
2 1 3 2 4 2 5 2 7 2 7 4 7 5 6 2  
};  
The DATA step displays the table in Output 29.24.8.

Output 29.24.8  Career Aspiration Data: Model Comparisons

<table>
<thead>
<tr>
<th>model comparison</th>
<th>chi**2</th>
<th>df</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 versus 1</td>
<td>7.0565</td>
<td>15</td>
<td>0.9561</td>
</tr>
<tr>
<td>3 versus 2</td>
<td>3.9668</td>
<td>1</td>
<td>0.0464</td>
</tr>
<tr>
<td>4 versus 2</td>
<td>1.9284</td>
<td>1</td>
<td>0.1649</td>
</tr>
<tr>
<td>5 versus 2</td>
<td>0.0048</td>
<td>1</td>
<td>0.9448</td>
</tr>
<tr>
<td>7 versus 2</td>
<td>6.2769</td>
<td>2</td>
<td>0.0433</td>
</tr>
<tr>
<td>7 versus 4</td>
<td>4.3485</td>
<td>1</td>
<td>0.0370</td>
</tr>
<tr>
<td>7 versus 5</td>
<td>6.2721</td>
<td>1</td>
<td>0.0123</td>
</tr>
<tr>
<td>6 versus 2</td>
<td>14.3778</td>
<td>2</td>
<td>0.0008</td>
</tr>
</tbody>
</table>

Although none of the seven models can be rejected when tested against the alternative of an unrestricted covariance matrix, the model comparisons make it clear that there are important differences among the models. Schwarz’s Bayesian criterion indicates Model 5 as the model of choice. The constraints added to Model 5 in Model 7 can be rejected (p = 0.0123), while Model 5 cannot be rejected when tested against the less constrained Model 2 (p = 0.9448). Hence, among the small number of models considered, Model 5 has strong statistical support. However, as Loehlin (1987, p. 106) points out, many other models for these data could be constructed. Further analysis should consider, in addition to simple modifications of the models, the possibility that more than one friend could influence a boy’s aspirations, and that a boy’s ambition might have some effect on his choice of friends. Pursuing such theories would be statistically challenging.

Example 29.25: Fitting a Latent Growth Curve Model

Latent factors in structural equation modeling are constructed to represent important unobserved hypothetical constructs. However, with some manipulations latent factors can also represent random effects in models. In this example, a simple latent growth curve model is considered. You use latent factors to represent the random intercepts and slopes in the latent growth curve model.
Sixteen individuals were invited to a training program that was designed to boost self-confidence. During the training, the individuals' confidence levels were measured at five time points: initially and four more times separated by equal intervals. The data are stored in the following SAS data set:

```
data growth;
  input y1 y2 y3 y4 y5;
datalines;
17.6  21.4  25.6  32.1  37.7
13.2  14.3  18.9  20.3  25.4
11.6  13.5  17.4  22.1  39.6
10.7  11.1  13.2  18.2  21.4
18.7  23.7  28.6  31.5  34.0
18.3  19.2  20.5  23.2  25.9
  ... (remaining data)
17.0  22.9  26.9  31.9  35.6
10.4  13.6  18.0  25.6  29.3
17.7  19.0  22.5  28.5  30.7
14.5  19.4  21.1  28.8  31.5
20.0  21.4  28.9  30.2  35.6
14.6  19.3  21.7  28.5  32.0
11.7  15.2  19.1  23.7  28.7
;
```

First, consider a simple linear regression model for the confidence levels at time $t$ due to training. That is,

$$y_t = \alpha + \beta T_t + e_t$$

where $y_t$ represents the confidence level at time $t$ ($t = 1, 2, \ldots, 5$), $\alpha$ represents the intercept, $\beta$ represents the slope or the effect of training, $T_t$ represents the fixed time point at $t$ ($T_1 = 0$ and $T_i = T_{i-1} + 1$), and $e_t$ is the error term at time $t$.

This simple linear regression assumes that the effect of training (slope) and the intercept are constants for the individuals. However, individual differences are rules rather than exceptions. It is thus more reasonable to argue that an index $i$ for individuals should be added to the intercept and slope in the model. As a result, the following individualized regression model is derived:

$$y_{it} = \alpha_i + \beta_i T_t + e_t$$

where $i = 1, 2, \ldots, 16$. In this model, individuals are assumed to have different intercepts and slopes (regression coefficients). Note that theoretically $e_t$ could also be “individualized” as $e_{ti}$ in the model. But this is not done because such a model would be unnecessarily complicated without gaining additional insights in return.

Unfortunately, this individualized model with individual intercepts and slopes cannot be estimated directly. If you treat each $\alpha_i$ and $\beta_i$ as fixed parameters, you are going to have too many parameters for the model to be identified or estimable. A workable solution is to treat $\alpha$ and $\beta$ in the original linear regression model as random variables instead. That is, the latent growth curve model of interest is as follows:

$$y_t = \alpha + \beta T_t + e_t$$

where $(\alpha, \beta)$ is bivariate normal with unknown means, variances, and covariance. Therefore, instead of having 16 intercepts and 16 slopes to estimate in the individualized regression model, the final latent growth
curve model has to estimate only two means, two variances and one covariance in the bivariate distribution of \((\alpha, \beta)\).

To use PROC CALIS to fit this latent growth curve model, the random intercept and effect are treated as if they were covarying latent factors. To make them stand out more as latent variables, the random intercept and slope are renamed as \(f_\alpha\) and \(f_\beta\) in the following structural equation:

\[
y_t = f_\alpha + T_t f_\beta + e_t
\]

where \(f_\alpha\) and \(f_\beta\) are bivariate-normal latent variables. This model assumes that the error distribution is time dependent (with the index \(t\)). A simpler version is to make this error term invariant over time, which is then represented by the following model with constrained error variances:

\[
y_t = f_\alpha + T_t f_\beta + e
\]

This constrained model is considered first. The LINEQS modeling language is used to specify this constrained model, as shown in the following statements.

```sas
proc calis method=ml data=growth nostand noparmname;
lineqs
  y1 = 0. * Intercept + f_alpha + e1,
  y2 = 0. * Intercept + f_alpha + 1 * f_beta + e2,
  y3 = 0. * Intercept + f_alpha + 2 * f_beta + e3,
  y4 = 0. * Intercept + f_alpha + 3 * f_beta + e4,
  y5 = 0. * Intercept + f_alpha + 4 * f_beta + e5;
_variance
  f_alpha f_beta,
  e1-e5 = 5 * evar;
_mean
  f_alpha f_beta;
_cov
  f_alpha f_beta;
_fitindex on(only)=[chisq df probchi];
run;
```

In the LINEQS model specification, \(f_\alpha\) and \(f_\beta\) are treated as latent factors representing the random intercept and random slope, respectively. The \(f_\) prefix for latent factors is required as a convention in the LINEQS modeling language. See the sections “Naming Variables in the LINEQS Model” on page 1403 and “Naming Variables and Parameters” on page 1433 for details.

Notice that you need to set the ordinary (non-random) intercepts for endogenous variables to zero by the 0.*Intercept specification because non-random intercepts for observed endogenous variables are default parameters in the LINEQS model. Because you have already used \(f_\alpha\) as the random intercept, you must turn off the default non-random intercept term for the observed endogenous variables \(y1\)–\(y5\). Otherwise, your latent growth curve model might be over-parameterized.

At \(T_1 = 0\), \(y_1\) represents the initial confidence measurement so that it is not subject to the random effect \(f_\beta\). The next four measurements \(y_2\), \(y_3\), \(y_4\), and \(y_5\) are measured at time points \(T_2\), \(T_3\), \(T_4\), and \(T_5\), respectively. These are fixed time points with constant values 1, 2, 3, and 4, respectively, in the equations of the LINEQS statement.

The means, variances and covariances of \(f_\alpha\) and \(f_\beta\) are parameters in the model. The variances of these two latent variables are specified in the VARIANCE statement, while their covariance is specified
Example 29.25: Fitting a Latent Growth Curve Model

in the COV statement. The means of f_alpha and f_beta are specified in the MEAN statement. Unlike the specification for the variances of e1–e5. All these parameters for the latent factors are unnamed because you do not need to constrain them by references.

The error variances for e1–e5 are also specified in the VARIANCE statement. Using the shorthand notation 5 * evar, the parameter name evar is repeated five times for the five error variances. This constrains the error variances for e1–e5 to be equal.

You also use some special printing options in this example. In the PROC CALIS statement, the NOSTAND option is specified because standardized solution is not of interest. The reason is that y1–y5 were already measured on comparable scales, making standardization unnecessary for interpretations. Another printing option specified is the NOPARMNAME option in the PROC CALIS statement. This option suppresses the printing of parameter names in the output for estimation. This makes the output look more concise when you do not need to make references to the parameter names. Still another printing option used is the ON(ONLY)= option of the FITINDEX statement. This option trims down the display of fit indices to include only those listed in the option. See the FITINDEX statement on page 1268 for details.

Output 29.25.1 shows the fit summary table.

**Output 29.25.1** Random Intercepts and Effects with Constrained Error Variances: Model Fit

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
</tbody>
</table>

In Output 29.25.1, the chi-square value in the fit summary table is 31.431 (df = 14, p < 0.01), which is a statistically significant result that might indicate a poor model fit. Despite that, it is illustrative to continue to look at the main estimation results, which are shown in the following table.

**Output 29.25.2** Estimation of Random Intercepts and Effects with Constrained Error Variances

<table>
<thead>
<tr>
<th>Estimates for Variances of Exogenous Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Type</td>
</tr>
<tr>
<td>Latent</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Covariances Among Exogenous Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Var1</td>
</tr>
<tr>
<td>f_alpha</td>
</tr>
</tbody>
</table>
Chapter 29: The CALIS Procedure

Output 29.25.2 continued

| Variable Type | Variable | Estimate | Standard Error | t Value | Pr > |t| |
|---------------|----------|----------|----------------|---------|-------|---|
| Latent        | f_alpha  | 14.15875 | 1.02906        | 13.7589 | < .0001 | |
|               | f_beta   | 4.04813  | 0.27563        | 14.6867 | < .0001 | |

In Output 29.25.2, the estimated variance of the random intercept \( \alpha \), which is represented by the variance estimate of the latent factor \( f_{\alpha} \), is 13.891 \( (t = 2.389) \). In the next row of the same table, the variance estimate of the random effect \( \beta \), which is represented by the variance estimate of the latent factor \( f_{\beta} \), is 0.807 \( (t = 1.913) \).

The covariance of the random intercept and the random effect is shown in the next table for “Covariances Among Exogenous Variables.” A negative estimate of –0.353 is shown. This means that the initial self-confidence level and the boosting effect of training are negatively correlated. The higher the initial self-confidence level, the smaller the training effect.

In the last table for the “Mean Parameters,” the estimated mean of the random intercept is 14.159, which is an estimate of the averaged initial self-confidence level. The estimated mean of random effect is 4.048, which is an estimate of the averaged training effect. They are both significantly different from zero.

Given that the model does not fit that well, perhaps you should not take the interpretations of these estimates so seriously. Knowing that the distribution of the errors might have been time-dependent, you now try to improve the fit of the model by relaxing the constraint about common error variances. You can use the following specifications:

```plaintext
proc calis method=ml data=growth nostand noparmname;
  lineqs
    y1 = 0. * Intercept + f_alpha + e1,
    y2 = 0. * Intercept + f_alpha + 1 * f_beta + e2,
    y3 = 0. * Intercept + f_alpha + 2 * f_beta + e3,
    y4 = 0. * Intercept + f_alpha + 3 * f_beta + e4,
    y5 = 0. * Intercept + f_alpha + 4 * f_beta + e5;
  variance
    f_alpha f_beta,
    e1-e5;
  mean
    f_alpha f_beta;
  cov
    f_alpha f_beta;
  fitindex on(only)=[chisq df probchi];
run;
```

In this new specification, there is only one change in the VARIANCE statement from the previous specification. That is, you now specify only the error variables without putting parameter names for them. This makes the variances of \( e1-e5 \) free (unconstrained) parameters in the model.

Output 29.25.3 shows the model fit summary.
Example 29.25: Fitting a Latent Growth Curve Model

Output 29.25.3 Random Intercepts and Effects with Unconstrained Error Variances: Model Fit

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
</tbody>
</table>

The chi-square for the unconstrained model is 11.625 ($df = 10, p > .10$). This indicates an acceptable model fit. The chi-square difference test can also be conducted for testing the previous constrained model against this new model. The chi-square difference is $19.81 = 31.431 – 11.625$. With $df = 4$, this chi-square difference value is statistically significant at $\alpha=0.01$, indicating a significant improvement of model fit by using the unconstrained model.

Output 29.25.4 shows the estimation results.

Output 29.25.4 Estimation of Random Intercepts and Effects with Unconstrained Error Variances

<table>
<thead>
<tr>
<th>Estimates for Variances of Exogenous Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Type</td>
</tr>
<tr>
<td>Latent</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Covariances Among Exogenous Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Var1</td>
</tr>
<tr>
<td>f_alpha</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Type</td>
</tr>
<tr>
<td>Latent</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

The estimation results for the unconstrained model present a slightly different picture than the constrained model. While the estimates for the means and variances of the random intercept and the random training effect look similar in both models, estimates of the covariance between the random intercept and the random training effect are quite different in the two models. The covariance estimate is negative (–0.353) in the constrained model, but it is positive (0.353) in the unconstrained model. However, because the covariance estimates are not statistically significant in both models ($t = –0.310$ and 0.391, respectively), you wonder whether the current data are showing strong evidence that supports one way or another. To get a clearer picture, perhaps you need to collect more data and fit the models again to examine the significance of the covariance between the random intercept and slope.
Example 29.26: Higher-Order and Hierarchical Factor Models

In this example, confirmatory higher-order and hierarchical factor models are fitted by PROC CALIS.

In higher-order factor models, factors are at different levels. The higher-order factors explain the relationships among factors at the next lower level, in the same way that the first-order factors explain the relationships among manifest variables. For example, in a two-level higher order factor model you have nine manifest variables \( V_1 \)–\( V_9 \) with three first-order factors \( F_1 \)–\( F_3 \). The first-order factor pattern of the model might appear like the following:

\[
\begin{array}{ccc}
F_1 & F_2 & F_3 \\
V_1 & x & \\
V_2 & x & \\
V_3 & x & \\
V_4 & x & \\
V_5 & x & \\
V_6 & x & \\
V_7 & x & \\
V_8 & x & \\
V_9 & x & \\
\end{array}
\]

where each “x” marks a nonzero factor loading and all other unmarked entries are fixed zeros in the model. To explain the correlations among the first-order factors, a second-order factor \( F_4 \) is hypothesized with the following second-order factor pattern:

\[
\begin{array}{ccc}
F_4 \\
F_1 & x & \\
F_2 & x & \\
F_3 & x & \\
\end{array}
\]

If substantiated by your theory, you might have higher-order factor models with more than two levels.

In hierarchical factor models, all factors are at the same (first-order) level but are different in their clusters of manifest variables related. Using the terminology of Yung, Thissen, and McLeod (1999), factors in hierarchical factor models are classified into “layers.” The factors in the first layer partition the manifest variables into clusters so that each factor has a distinct cluster of related manifest variables. This part of the factor pattern of the hierarchical factor model is similar to that of the first-order factor model for manifest variables. The next layer of factors in the hierarchical factor model again partitions the manifest variables into clusters. However, this time each cluster contains at least two clusters of manifest variables that are formed in the previous layer. For example, the following is a factor pattern of a confirmatory hierarchical factor model with two layers:
Example 29.26: Higher-Order and Hierarchical Factor Models

F1–F3 are first-layer factors and F4 is the only second-layer factor. This special kind of two-layer hierarchical pattern is also known as the bifactor solution. In a bifactor solution, there are two classes of factors—group factors and a general factor. For example, in the preceding hierarchical factor pattern F1–F3 are group factors for different abilities and F4 is a general factor such as “intelligence” (see, for example, Holzinger and Swineford 1937). See Mulaik and Quartetti (1997) for more examples and distinctions among various types of hierarchical factor models. Certainly, if substantiated by your theory, hierarchical factor models with more than two layers are possible.

In this example, you use PROC CALIS to fit these two types of confirmatory factor models. First, you fit a second-order factor model to a real data set. Then you fit a bifactor model to the same data set. In the final section of this example, an informal account of the relationship between the higher-order and hierarchical factor models is attempted. Techniques for constraining parameters using PROC CALIS are also shown. This final section might be too technical in the first reading. Interested readers are referred to articles by Mulaik and Quartetti (1997), Schmid and Leiman (1957), and Yung, Thissen, and McLeod (1999) for more details.

A Second-Order Factor Analysis Model

In this section, a second-order confirmatory factor analysis model is applied to a correlation matrix of Thurstone reported by McDonald (1985). The correlation matrix is read into a SAS data set in the following statements:

data Thurst(type=corr);
title "Example of THURSTONE resp. McDONALD (1985, p.57, p.105)";
   _type_ = 'corr'; input _name_ $ V1-V9;
   label V1='Sentences' V2='Vocabulary' V3='Sentence Completion'
   V4='First Letters' V5='Four-letter Words' V6='Suffices'
   V7='Letter series' V8='Pedigrees' V9='Letter Grouping';
datalines;
V1  1. . . . . . . . . . .
V2 .828 1. . . . . . . . .
V3 .776 .779 1. . . . . . .
V4 .439 .493 .460 1. . . . . .
V5 .432 .464 .425 .674 1. . . . .
V6 .447 .489 .443 .590 .541 1. . . .
V7 .447 .432 .401 .381 .402 .288 1. . . .
V8 .541 .537 .534 .350 .367 .320 .555 1. . . .
V9 .380 .358 .359 .424 .446 .325 .598 .452 1. .
;
Variables in this data set are measures of cognitive abilities. Three factors are assumed for these nine variable V1–V9. These three factors are the first-order factors in the analysis. A second-order factor is also assumed to explain the correlations among the three first-order factors.

The following statements define a second-order factor model by using the LINEQS modeling language.

```sas
proc calis corr data=Thurst method=max nobs=213 nose nostand;
   lineqs
      V1 = X11 * Factor1 + E1,
      V2 = X21 * Factor1 + E2,
      V3 = X31 * Factor1 + E3,
      V4 = X42 * Factor2 + E4,
      V5 = X52 * Factor2 + E5,
      V6 = X62 * Factor2 + E6,
      V7 = X73 * Factor3 + E7,
      V8 = X83 * Factor3 + E8,
      V9 = X93 * Factor3 + E9,
      Factor1 = L1g * FactorG + E10,
      Factor2 = L2g * FactorG + E11,
      Factor3 = L3g * FactorG + E12;
   variance
      FactorG = 1.,
      E1-E12 = U1-U9 W1-W3;
   bounds
      0. <= U1-U9;
   fitindex ON(ONLY)=[chisq df probchi];
/* SAS Programming Statements: Dependent parameter definitions */
   W1 = 1. - L1g * L1g;
   W2 = 1. - L2g * L2g;
   W3 = 1. - L3g * L3g;
run;
```

In the first nine equations of the LINEQS statement, variables V1–V3 are manifest indicators of latent factor Factor1, variables V4–V6 are manifest indicators of latent factor Factor2, and variables V7–V9 are manifest indicators of latent factor Factor3. In the last three equations of the LINEQS statement, the three first-order factors Factor1–Factor3 are explained by a common source: FactorG. Hence, Factor1–Factor3 are correlated due to the common source FactorG in the model.

An error term is added to each equation in the LINEQS statement. These error terms E1–E12 are needed because the factors are not assumed to be perfect predictors of the corresponding outcome variables.

In the VARIANCE statement, you specify variance parameters for all independent or exogenous variables in the model: FactorG, and E1–E12. The variance of FactorG is fixed at one for identification. Variances for E1–E9 are given parameter names U1–U9, respectively. Variances for E10–E12 are given parameter names W1–W3, respectively. Note that for model identification purposes, W1–W3 are defined as dependent parameters in the SAS programming statements. That is,

\[ W_i = 1. - L_{ig}^2 \quad (i = 1, 2, 3) \]

These dependent parameter definitions ensure that the variances for Factor1–Factor3 are fixed at ones for identification.

In the BOUNDS statement, you specify that variance parameters U1–U9 must be positive in the solution.

In addition to the statements for model specification, you specify some output control options in the PROC CALIS statement. You use the NOSE and NOSTAND options suppress the display of standard errors and
standardized results. In the `FITINDEX` statement, the `ON(ONLY)=` option requests only the model fit chi-square and its associated degrees of freedom and $p$-value be shown in the fit summary table. Using printing options in PROC CALIS to reduce the amount the of printout is a good practice. It makes your output more focused, as you output only what you need in a particular situation.

In Output 29.26.1, parameters and their initial values, gradients, and bounds are shown.

**Output 29.26.1** Parameters in the Model

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Gradient</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>X11</td>
<td>1.00000</td>
<td>0.13476</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>X21</td>
<td>1.01408</td>
<td>0.17327</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>X31</td>
<td>0.95518</td>
<td>0.12174</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>X42</td>
<td>1.00000</td>
<td>0.22548</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>X52</td>
<td>0.96603</td>
<td>0.21304</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>X62</td>
<td>0.88305</td>
<td>0.19782</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>X73</td>
<td>1.00000</td>
<td>0.21041</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>X83</td>
<td>1.03403</td>
<td>0.39324</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>X93</td>
<td>0.91752</td>
<td>0.19880</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>L1g</td>
<td>0.75060</td>
<td>-0.57492</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>L2g</td>
<td>0.64268</td>
<td>-0.50975</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>L3g</td>
<td>0.60919</td>
<td>-0.56538</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>U1</td>
<td>0.18879</td>
<td>0.14837</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td>U2</td>
<td>0.16579</td>
<td>0.08989</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td>U3</td>
<td>0.25988</td>
<td>-0.03231</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td>U4</td>
<td>0.33068</td>
<td>0.20120</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td>U5</td>
<td>0.37538</td>
<td>0.09124</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td>U6</td>
<td>0.47808</td>
<td>-0.03595</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td>U7</td>
<td>0.44813</td>
<td>0.20918</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td>U8</td>
<td>0.40994</td>
<td>-0.12469</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td>U9</td>
<td>0.53541</td>
<td>0.05959</td>
<td>0</td>
<td>.</td>
</tr>
</tbody>
</table>

Value of Objective Function = 0.5693888709

The first table contains all the independent parameters. There are twenty-one in total. Parameters W1–W3, which are defined in the SAS programming statements, are shown in the next table for dependent parameters. Their initial values are computed as functions of the independent parameters.

**Output 29.26.2** shows the information about optimization—iteration history and the convergence status.
First, there are 21 independent parameters in the optimization by using 45 “Functions (Observations).” The so-called functions refer to the moments in the model that are structured with parameters. Nine lower bounds, which are specified for the error variance parameters, are specified in the optimization. The next table for iteration history shows that the optimization stops in 11 iterations. The notes at the bottom of table show that the solution converges without problems.

Output 29.26.3 shows the fit summary table. The chi-square model fit value is 38.196, with $df = 24$, and $p = 0.033$. This indicates a satisfactory model fit.

Output 29.26.4 shows the fitted equations with final estimates. Interpretations of these loadings are not done here. The last table in this output shows various variance estimates. These estimates are classified by whether they are for the latent variables, error variables, or disturbance variables.
Example 29.26: Higher-Order and Hierarchical Factor Models

Output 29.26.4 Estimation Results

<table>
<thead>
<tr>
<th>Linear Equations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>V1 = 0.9047 Factor1 + 1.0000 E1</td>
<td></td>
</tr>
<tr>
<td>V2 = 0.9138 Factor1 + 1.0000 E2</td>
<td></td>
</tr>
<tr>
<td>V3 = 0.8561 Factor1 + 1.0000 E3</td>
<td></td>
</tr>
<tr>
<td>V4 = 0.8358 Factor2 + 1.0000 E4</td>
<td></td>
</tr>
<tr>
<td>V5 = 0.7972 Factor2 + 1.0000 E5</td>
<td></td>
</tr>
<tr>
<td>V6 = 0.7026 Factor2 + 1.0000 E6</td>
<td></td>
</tr>
<tr>
<td>V7 = 0.7808 Factor3 + 1.0000 E7</td>
<td></td>
</tr>
<tr>
<td>V8 = 0.7202 Factor3 + 1.0000 E8</td>
<td></td>
</tr>
<tr>
<td>V9 = 0.7035 Factor3 + 1.0000 E9</td>
<td></td>
</tr>
<tr>
<td>Factor1 = 0.8221 FactorG + 1.0000 E10</td>
<td></td>
</tr>
<tr>
<td>Factor2 = 0.7818 FactorG + 1.0000 E11</td>
<td></td>
</tr>
<tr>
<td>Factor3 = 0.8150 FactorG + 1.0000 E12</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Estimates for Variances of Exogenous Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Type</td>
</tr>
<tr>
<td>Latent FactorG</td>
</tr>
<tr>
<td>Error E1</td>
</tr>
<tr>
<td>E2</td>
</tr>
<tr>
<td>E3</td>
</tr>
<tr>
<td>E4</td>
</tr>
<tr>
<td>E5</td>
</tr>
<tr>
<td>E6</td>
</tr>
<tr>
<td>E7</td>
</tr>
<tr>
<td>E8</td>
</tr>
<tr>
<td>E9</td>
</tr>
<tr>
<td>Disturbance E10</td>
</tr>
<tr>
<td>E11</td>
</tr>
<tr>
<td>E12</td>
</tr>
</tbody>
</table>

For illustration purposes, you might check whether the model constraints put on the variances of Factor1–Factor3 are honored (although this should have been taken care of in the optimization). For example, the variance of Factor1 should be:

\[ 1 = (\text{Loading on FactorG})^2 + \text{Variance of E10} \]

Extracting the estimates from the output, you indeed verify the required equality, as shown in the following:

\[ 1.0000 = (0.8221)^2 + 0.32420 \]

A Bifactor Model

A bifactor model (or a hierarchical factor model with two layers) for the same data set is now considered. In this model, the same set of factors as in the preceding higher-order factor model are used. The most notable difference is that the second-order factor FactorG in the higher-order factor model is no longer a factor of the first-order factors Factor1–Factor3. Instead, FactorG, like Factor1–Factor3, now also serves as a factor of the observed variable V1–V9. Unlike Factor1–Factor3, FactorG is considered to be a general factor in the sense
that all observed variables have direct functional relationships with it. In contrast, Factor1–Factor3 are group factors in the sense that each of them has a direct functional relationship with only one group of observed variables. Because of the coexistence of a general factor and group factors at the same factor level, such a hierarchical model is also called a bifactor model.

The bifactor model is specified in the following statements:

```
proc calis corr data=Thurst method=max nobs=213 nose nostand;
  lineqs
    V1 = X1 * Factor1 + X1g * FactorG + E1,
    V2 = X2 * Factor1 + X2g * FactorG + E2,
    V3 = X3 * Factor1 + X3g * FactorG + E3,
    V4 = X4 * Factor2 + X4g * FactorG + E4,
    V5 = X5 * Factor2 + X5g * FactorG + E5,
    V6 = X6 * Factor2 + X6g * FactorG + E6,
    V7 = X7 * Factor3 + X7g * FactorG + E7,
    V8 = X8 * Factor3 + X8g * FactorG + E8,
    V9 = X9 * Factor3 + X9g * FactorG + E9;
  variance
    Factor1-Factor3 = 3 * 1.,
    FactorG = 1. ,
    E1-E9 = U1-U9;
  cov
    Factor1-Factor3 FactorG = 6 * 0.;
  bounds
    0. <= U1-U9;
  fitindex ON(ONLY)=[chisq df probchi];
run;
```

In the LINEQS statement, there are only nine equations for the manifest variables in the model. Unlike the second-order factor model fitted previously, Factor1–Factor3 are no longer functionally related to FactorG and therefore there are no equations with Factor1–Factor3 as outcome variables.

The factor variances are all fixed at 1 in the VARIANCE statement. The variance parameters for E1–E9 are named U1–U9, respectively. The BOUNDS statement, again, is specified so that only positive estimates are accepted for error variance estimates.

All factors in the bifactor model are uncorrelated. In the COV statement, you specify that the six covariances among Factor1–Factor3 and FactorG are all zero. This specification is necessary because by default exogenous variables (excluding error terms) in the LINEQS model are correlated.

Like the previous PROC CALIS run, options are specified in the PROC CALIS and the FITINDEX statements to reduce the amount of default output.

There are more parameters in this model than in the preceding higher-order factor model, as shown in Output 29.26.5, which shows the optimization information.

```
Output 29.26.5  Optimization

<table>
<thead>
<tr>
<th>Optimization Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Constraints</td>
</tr>
<tr>
<td>Objective Function</td>
</tr>
<tr>
<td>Max Abs Gradient Element</td>
</tr>
<tr>
<td>Radius</td>
</tr>
</tbody>
</table>
```
Example 29.26: Higher-Order and Hierarchical Factor Models

There are 27 parameters in the bifactor model: nine for the loadings on the group factors Factor1–Factor3, nine for the loadings on the general factor FactorG, and nine for the variances of errors E1–E9. The optimization converges in 14 iterations without problems.

A fit summary table is shown in Output 29.26.6

Output 29.26.6  Fit Summary

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
</tbody>
</table>

The fit of this model is quite good. The chi-square value is 24.216, with $df = 18$ and $p = 0.148$. This is expected because the bifactor model has more parameters than the second-order factor model, which already has a good fit with fewer parameters.

Estimation results are shown in Output 29.26.7. Estimates are left uninterpreted because they are not the main interest of this example.
Output 29.26.7 Estimation Results

Linear Equations

\[
\begin{align*}
V1 &= -0.4879 \text{ Factor1} + 0.7679 \text{ FactorG} + 1.0000 \text{ E1} \\
V2 &= -0.4523 \text{ Factor1} + 0.7909 \text{ FactorG} + 1.0000 \text{ E2} \\
V3 &= -0.4045 \text{ Factor1} + 0.7536 \text{ FactorG} + 1.0000 \text{ E3} \\
V4 &= 0.6140 \text{ Factor2} + 0.6084 \text{ FactorG} + 1.0000 \text{ E4} \\
V5 &= 0.5058 \text{ Factor2} + 0.5973 \text{ FactorG} + 1.0000 \text{ E5} \\
V6 &= 0.3943 \text{ Factor2} + 0.5718 \text{ FactorG} + 1.0000 \text{ E6} \\
V7 &= -0.7273 \text{ Factor3} + 0.5669 \text{ FactorG} + 1.0000 \text{ E7} \\
V8 &= -0.2468 \text{ Factor3} + 0.6623 \text{ FactorG} + 1.0000 \text{ E8} \\
V9 &= -0.4091 \text{ Factor3} + 0.5300 \text{ FactorG} + 1.0000 \text{ E9}
\end{align*}
\]

Estimates for Variances of Exogenous Variables

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Variable</th>
<th>Parameter</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latent</td>
<td>Factor1</td>
<td></td>
<td>1.00000</td>
</tr>
<tr>
<td></td>
<td>Factor2</td>
<td></td>
<td>1.00000</td>
</tr>
<tr>
<td></td>
<td>Factor3</td>
<td></td>
<td>1.00000</td>
</tr>
<tr>
<td></td>
<td>FactorG</td>
<td></td>
<td>1.00000</td>
</tr>
<tr>
<td>Error</td>
<td>E1</td>
<td>U1</td>
<td>0.17236</td>
</tr>
<tr>
<td></td>
<td>E2</td>
<td>U2</td>
<td>0.16984</td>
</tr>
<tr>
<td></td>
<td>E3</td>
<td>U3</td>
<td>0.26848</td>
</tr>
<tr>
<td></td>
<td>E4</td>
<td>U4</td>
<td>0.25281</td>
</tr>
<tr>
<td></td>
<td>E5</td>
<td>U5</td>
<td>0.38735</td>
</tr>
<tr>
<td></td>
<td>E6</td>
<td>U6</td>
<td>0.51757</td>
</tr>
<tr>
<td></td>
<td>E7</td>
<td>U7</td>
<td>0.14966</td>
</tr>
<tr>
<td></td>
<td>E8</td>
<td>U8</td>
<td>0.50039</td>
</tr>
<tr>
<td></td>
<td>E9</td>
<td>U9</td>
<td>0.55175</td>
</tr>
</tbody>
</table>

One might ask whether this bifactor (hierarchical) model provides a significantly better fit than the previous second-order model. Can one use a chi-square difference test for nested models to answer this question? The answer is yes.

Although it is not obvious that the previous second-order factor model is nested within the current bifactor model, a general nested relationship between the higher-order factor and the hierarchical factor model is formally proved by Yung, Thissen, and McLeod (1999). Therefore, a chi-square difference test can be conducted using the following DATA step:

```sas
data _null_;  
  df0 = 24; chi0 = 38.1963;  
  df1 = 18; chi1 = 24.2163;  
  diff = chi0-chi1;  
  p = 1.-probchi(chi0-chi1,df0-df1);  
  put 'Chi-square difference = ' diff;  
  put 'p-value = ' p;  
run;
```
The results are shown in the following:

**Output 29.26.8 Chi-square Difference Test**

Chi-square difference = 13.98  
p-value = 0.0298603746

The chi-square difference is 13.98, with $df = 6$ and $p = 0.02986$. If $\alpha$-level is set at 0.05, the bifactor model indicates a significantly better fit. But if $\alpha$-level is set at 0.01, statistically the two models fit equally well to the data.

In the next section, it is demonstrated that the second-order factor model is indeed nested within the bifactor model, and hence the chi-square test conducted in the previous section is justified. In addition, through the demonstration of the nested relationship between the two classes of models, you can see how some parameter constraints in structural equation model can be set up in PROC CALIS.

For some practical researchers, the technical details involved in the next section might not be of interest and therefore could be skipped.

**A Constrained Bifactor Model and Its Equivalence to the Second-Order Factor Model**

To demonstrate that the second-order factor model is indeed nested within the bifactor model, a constrained bifactor model is fitted in this section. This constrained bifactor model is essentially the same as the preceding bifactor model, but with additional constraints on the factor loadings. Hence, the constrained bifactor model is nested within the unconstrained bifactor model.

Furthermore, if it can be shown that the constrained bifactor model is equivalent to the previous second-order factor, then the second-order factor model must also be nested within the unconstrained bifactor model. As a result, it justifies the chi-square difference test conducted in the previous section.

The construction of such a constrained bifactor model is based on Yung, Thissen, and McLeod (1999). In the following statements, a constrained bifactor model is specified.

```plaintext
proc calis corr data=Thurst method=max nobs=213 nose nostand;
lineqs
V1 = X11 * Factor1 + X1g * FactorG + E1,
V2 = X21 * Factor1 + X2g * FactorG + E2,
V3 = X31 * Factor1 + X3g * FactorG + E3,
V4 = X42 * Factor2 + X4g * FactorG + E4,
V5 = X52 * Factor2 + X5g * FactorG + E5,
V6 = X62 * Factor2 + X6g * FactorG + E6,
V7 = X73 * Factor3 + X7g * FactorG + E7,
V8 = X83 * Factor3 + X8g * FactorG + E8,
V9 = X93 * Factor3 + X9g * FactorG + E9;
variance
Factor1-Factor3 = 3 * 1.,
FactorG = 1. ,
E1-E9 = U1-U9;
cov
Factor1-Factor3 FactorG = 6 * 0.;
bounds
0. <= U1-U9;
fitmap ON(ONLY)=[chisq df probchi];
```
In this constrained model, you add a `PARAMETERS` statement and nine SAS programming statements to the previous bifactor model. In the `PARAMETERS` statement, three new independent parameters are added: `p1`, `p2`, and `p3`. These parameters represent the proportions that constrain the factor loadings of the observed variables on the group factors `Factor1`–`Factor3` and the general factor `FactorG`. They are all free parameters and have initial values at 0.5. The next nine SAS programming statements represent the proportionality constraints imposed. For example, `X1g`–`X3g` are now dependent parameters expressed as functions of `p1`, `X11`, `X21`, and `X31`. Adding three new parameters (in the `PARAMETERS` statement) and redefining nine original parameters as dependent (in the SAS programming statements) is equivalent to adding six (=9-3) constraints to the original bifactor model. Mathematically, the additional statements in specifying the constrained bifactor model realizes the following six constraints:

\[
\begin{align*}
X1g &= X2g = X3g \\
X11 &= X21 = X31 \\
X4g &= X5g = X6g \\
X42 &= X52 = X62 \\
X7g &= X8g = X9g \\
X73 &= X83 = X93
\end{align*}
\]

Obviously, with these six constraints the current constrained bifactor model is nested within the unconstrained version. What remains to be shown is that this constrained bifactor model is indeed equivalent to the previous second-order factor model. If so, the second-order factor model is also nested within the unconstrained bifactor model. One evidence for the equivalence of the current constrained bifactor model and the second-order factor model is from the fit summary table shown in Output 29.26.10. But first, it is also useful to consider the optimization information of the constrained bifactor model, which is shown in Output 29.26.9.
As shown Output 29.26.9, there are 21 independent parameters in the constrained bifactor model for the 45 “Functions (Observations).” These numbers match those of the second-order factor model exactly. The optimization shows some problems in initial iterations. The iteration numbers with asterisks indicate that the Hessian matrix is not positive definite in those iterations. But as long as the final converged iteration is not marked with an asterisk, the problems exhibited in early iterations do not raise any concern, as in the current case. Next, the fit summary is shown in Output 29.26.10.
In Output 29.26.10, the chi-square value in the fit summary table is 38.196, with \( df = 24 \), and \( p = 0.033 \). Again, these numbers match those of the second-order factor model exactly. These matches (same model fit with the same number of parameters) are necessary (but not sufficient) to show that the constrained bifactor model is equivalent to the second-order model. Stronger evidence is now presented.

In Output 29.26.11, estimation results of the constrained bifactor model are shown.

**Output 29.26.11 Estimation Results**

**Linear Equations**

\[
\begin{align*}
V1 &= -0.5151 \text{ Factor1} + 0.7437 \text{ FactorG} + 1.0000 \text{ E1} \\
V2 &= -0.5203 \text{ Factor1} + 0.7512 \text{ FactorG} + 1.0000 \text{ E2} \\
V3 &= -0.4874 \text{ Factor1} + 0.7038 \text{ FactorG} + 1.0000 \text{ E3} \\
V4 &= 0.5211 \text{ Factor2} + 0.6534 \text{ FactorG} + 1.0000 \text{ E4} \\
V5 &= 0.4971 \text{ Factor2} + 0.6232 \text{ FactorG} + 1.0000 \text{ E5} \\
V6 &= 0.4381 \text{ Factor2} + 0.5493 \text{ FactorG} + 1.0000 \text{ E6} \\
V7 &= 0.4524 \text{ Factor3} + 0.6364 \text{ FactorG} + 1.0000 \text{ E7} \\
V8 &= 0.4173 \text{ Factor3} + 0.5869 \text{ FactorG} + 1.0000 \text{ E8} \\
V9 &= 0.4076 \text{ Factor3} + 0.5734 \text{ FactorG} + 1.0000 \text{ E9}
\end{align*}
\]

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Variable</th>
<th>Parameter</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latent</td>
<td>Factor1</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Factor2</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Factor3</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FactorG</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>E1</td>
<td>U1</td>
<td>0.18150</td>
</tr>
<tr>
<td></td>
<td>E2</td>
<td>U2</td>
<td>0.16493</td>
</tr>
<tr>
<td></td>
<td>E3</td>
<td>U3</td>
<td>0.26713</td>
</tr>
<tr>
<td></td>
<td>E4</td>
<td>U4</td>
<td>0.30150</td>
</tr>
<tr>
<td></td>
<td>E5</td>
<td>U5</td>
<td>0.36450</td>
</tr>
<tr>
<td></td>
<td>E6</td>
<td>U6</td>
<td>0.50641</td>
</tr>
<tr>
<td></td>
<td>E7</td>
<td>U7</td>
<td>0.39034</td>
</tr>
<tr>
<td></td>
<td>E8</td>
<td>U8</td>
<td>0.48136</td>
</tr>
<tr>
<td></td>
<td>E9</td>
<td>U9</td>
<td>0.50511</td>
</tr>
</tbody>
</table>

According to Yung, Thissen, and McLeod (1999), two models are equivalent if there is a one-to-one correspondence of the parameters in the models. This fact is illustrated for the constrained bifactor model and the second-order factor model.

First, the error variances for E1–E9 in the second-order factor model are transformed directly (using an identity map) to that of the bifactor models. The nine error variances in Output 29.26.4 for the second-order factor model match those of the constrained bifactor model exactly in Output 29.26.11. In addition, the variances of factors are fixed at one in both models. The error variances and the factor loadings at both factor levels in Output 29.26.4 for the second-order factor model are now transformed to yield the loading estimates in the constrained bifactor model.
Denote $P_1$ as the first-order factor loading matrix, $P_2$ as the second-order factor loading matrix, and $U_1^2$ be the matrix of variances for disturbances. That is, for the second-order factor model,

$$
P_1 = \begin{pmatrix}
0.9047 & 0 & 0 \\
0.9138 & 0 & 0 \\
0.8561 & 0 & 0 \\
0 & 0.8358 & 0 \\
0 & 0.7972 & 0 \\
0 & 0.7026 & 0 \\
0 & 0 & 0.7808 \\
0 & 0 & 0.7202 \\
0 & 0 & 0.7035 \\
\end{pmatrix}
$$

$$
P_2 = \begin{pmatrix}
0.8221 \\
0.7818 \\
0.8150 \\
\end{pmatrix}
$$

$$
U_1^2 = \begin{pmatrix}
0.3242 & 0 & 0 \\
0 & 0.3888 & 0 \\
0 & 0 & 0.3358 \\
\end{pmatrix}
$$

According to Yung, Thissen, and McLeod (1999), the transformation to obtain the estimates in the equivalent constrained bifactor model is:

$$
L_1 = P_1 U_1 \\
L_2 = P_1 P_2
$$

where $L_1$ is the matrix of the first-layer factor loadings (that is, loadings on group factors Factor1–Factor3), and $L_2$ is the matrix of the second-layer factor loadings (that is, loadings on FactorG) in the constrained bifactor model. Carrying out the matrix calculations for $L_1$ and $L_2$ shows that:

$$
L_1 = \begin{pmatrix}
0.5151 & 0 & 0 \\
0.5203 & 0 & 0 \\
0.4875 & 0 & 0 \\
0 & 0.5212 & 0 \\
0 & 0.4971 & 0 \\
0 & 0.4381 & 0 \\
0 & 0 & 0.4525 \\
0 & 0 & 0.4173 \\
0 & 0 & 0.4077 \\
\end{pmatrix}
$$

$$
L_2 = \begin{pmatrix}
0.7438 \\
0.7512 \\
0.7038 \\
0.6534 \\
0.6232 \\
0.5493 \\
0.6364 \\
0.5870 \\
0.5734 \\
\end{pmatrix}
$$
With very minor numerical differences and ignorable sign changes, these transformation results match the estimated loadings observed in Output 29.26.11 for the constrained bifactor model. Therefore, the second-order factor model is shown to be equivalent to the constrained bifactor model, and hence is nested within the unconstrained bifactor model.

**Example 29.27: Linear Relations among Factor Loadings**

In this example, you use the FACTOR modeling language of PROC CALIS to specify a confirmatory factor analysis model with linear constraints on loadings. You use SAS programming statements to set the constraints. This example also discusses the differences between fitting covariance structures and correlation structures in the current modeling context.

The correlation matrix of six variables from Kinzer and Kinzer (N=326) is used by Guttman (1957) as an example that yields an approximate simplex. McDonald (1980) uses this data set as an example of factor analysis where he assumes that the loadings on the second factor are linear functions of the loadings on the first factor. Let \( B \) be the factor loading matrix containing the two factors and six variables so that:

\[
B = \begin{pmatrix}
  b_{11} & b_{12} \\
  b_{21} & b_{22} \\
  b_{31} & b_{32} \\
  b_{41} & b_{42} \\
  b_{51} & b_{52} \\
  b_{61} & b_{62}
\end{pmatrix}
\]

and

\[
b_{j2} = \alpha + \beta b_{j1}, \quad j = 1, \ldots, 6
\]

The correlation structures are represented by:

\[
P = BB' + \Psi
\]

where \( \Psi = \text{diag}(\psi_{11}, \psi_{22}, \psi_{33}, \psi_{44}, \psi_{55}, \psi_{66}) \) represents the diagonal matrix of unique variances for the variables.

With parameters \( \alpha \) and \( \beta \) being unconstrained, McDonald (1980) has fitted an underidentified model with seven degrees of freedom. Browne (1982) imposes the following identification condition:

\[
\beta = -1
\]

In this example, Browne’s identification condition is imposed. The following is the specification of the confirmatory factor model using the FACTOR modeling language.
Example 29.27: Linear Relations among Factor Loadings

```sas
data kinzer(type=corr);
title "Data Matrix of Kinzer & Kinzer, see GUTTMAN (1957)";
  _type_ = 'corr';
  input _name_ $ var1-var6;
datalines;
var1 1.00 . . . . .
var2 .51 1.00 . . . .
var3 .46 .51 1.00 . . .
var4 .46 .47 .54 1.00 . .
var5 .40 .39 .49 .57 1.00 .
var6 .33 .39 .47 .45 .56 1.00;

proc calis data=kinzer nobs=326 nose;
  factor
    factor1 ===> var1-var6 = b11 b21 b31 b41 b51 b61 (6 *.6),
    factor2 ===> var1-var6 = b12 b22 b32 b42 b52 b62;
  pvar
    factor1-factor2 = 2 * 1.,
    var1-var6 = psi1-psi6 (6 *.3);
  cov
    factor1 factor2 = 0.;
  parameters alpha (1.);
  /* SAS Programming Statements to define dependent parameters */
  b12 = alpha - b11;
  b22 = alpha - b21;
  b32 = alpha - b31;
  b42 = alpha - b41;
  b52 = alpha - b51;
  b62 = alpha - b61;
  fitindex on(only)=[chisq df probchi];
run;
```

In the FACTOR statement, you specify two factors, named factor1 and factor2, for the variables. In this model, all manifest variables have nonzero loadings on the two factors. These loading parameters are specified after the equal signs and are named with the prefix ‘b.’ You specify the initial estimates in the parentheses for the parameters in the first entry of the FACTOR statement. The loadings in the first entry are all free parameters with initial estimates of .6. In the second entry of the FACTOR statement, you specify the Loadings of var1–var6 on factor2. However, these parameters are dependent, as shown in the SAS programming statements. Initial values for these dependent parameters are thus unnecessary.

In the PVAR statement, the factor variances are fixed at ones, while the error variances of the variables are free parameters named psi1–psi6. Again, you provide initial estimates for these error variance parameters. All have the initial value of 0.3.

An additional parameter alpha is specified in the PARAMETERS statement with an initial value of 1. Then, you use six SAS programming statements to define the loadings on the second factor as functions of the loadings on the first factor. Lastly, the FITINDEX statement is used to trim the results in the fit summary table.

In the specification, there are twelve loadings in the FACTOR statement and six error variances in the PVAR statement. Adding the parameter alpha in the list, there are 19 parameters in total. However, the loading parameters are not all independent of each other. As defined in the SAS programming statements, six loadings
are dependent. This reduces the number of free parameters to 13. Hence the degrees of freedom for the model is $8 = 21 - 13$. Notice that the factor variances are fixed at 1, as specified in the PVAR statement, and covariance among the two factors is fixed at zero, as specified in the COV statement.

Output 29.27.1 shows a concise fit summary table. The chi-square test statistic of model fit is 10.337 with $df = 8$ ($p = 0.242$). This indicates a good model fit.

**Output 29.27.1** Fit of the Correlation Structures

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>10.337</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>8</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.242</td>
</tr>
</tbody>
</table>

The estimated factor loading matrix is presented in **Output 29.27.2**, and the estimated error variances and the estimate for $\alpha$ are presented in **Output 29.27.3**.

**Output 29.27.2** Loading Estimates

<table>
<thead>
<tr>
<th>Factor Loading Matrix</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>factor1</td>
<td>factor2</td>
<td></td>
</tr>
<tr>
<td>var1</td>
<td>0.3609 [b11]</td>
<td>0.6174 [b12]</td>
</tr>
<tr>
<td>var2</td>
<td>0.3212 [b21]</td>
<td>0.6571 [b22]</td>
</tr>
<tr>
<td>var3</td>
<td>0.4859 [b31]</td>
<td>0.4923 [b32]</td>
</tr>
<tr>
<td>var4</td>
<td>0.5745 [b41]</td>
<td>0.4038 [b42]</td>
</tr>
<tr>
<td>var5</td>
<td>0.7985 [b51]</td>
<td>0.1797 [b52]</td>
</tr>
<tr>
<td>var6</td>
<td>0.6736 [b61]</td>
<td>0.3046 [b62]</td>
</tr>
</tbody>
</table>

**Output 29.27.3** Unique Variances and the Additional Parameter

<table>
<thead>
<tr>
<th>Error Variances</th>
<th>Variable Parameter</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>var1</td>
<td>psi1</td>
<td>0.53036</td>
</tr>
<tr>
<td>var2</td>
<td>psi2</td>
<td>0.44986</td>
</tr>
<tr>
<td>var3</td>
<td>psi3</td>
<td>0.48756</td>
</tr>
<tr>
<td>var4</td>
<td>psi4</td>
<td>0.47278</td>
</tr>
<tr>
<td>var5</td>
<td>psi5</td>
<td>0.31125</td>
</tr>
<tr>
<td>var6</td>
<td>psi6</td>
<td>0.53815</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional Parameters</th>
<th>Type</th>
<th>Parameter</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Independent</td>
<td>alpha</td>
<td>0.97825</td>
</tr>
</tbody>
</table>
All these estimates are essentially the same as those reported in Browne (1982). Notice that there are no standard error estimates in the output, as requested by the NOSE option in the PROC CALIS statement. Standard error estimates are not of interest in this example.

In fitting the preceding factor model, wrong covariance structures rather than the intended correlation structures have been specified. As pointed out by Browne (1982), fitting such covariance structures directly is not entirely appropriate for analyzing correlations. For example, when fitting the correlation structures, the diagonal elements of $P$ must always be fixed ones. This fact has never been enforced in the preceding specification. A simple check of the estimates will illustrate the problem. In Output 29.27.2, the loading estimates of VAR1 on the two factors are 0.3609 and 0.6174, respectively. In Output 29.27.3, the error variance estimate for VAR1 is 0.53036. The fitted variance of VAR1 can therefore be computed by the following equation:

$$\text{fitted variance} = 0.3609^2 + 0.6174^2 + 0.53036 = 1.0418$$

This fitted value is quite a bit off from 1.00, as required for the standardized variance of VAR1.

Fortunately, even though the wrong covariance structure model has been analyzed, the preceding analysis is not completely useless. For the current confirmatory factor model, according to Browne (1982) the estimates obtained from fitting the wrong covariance structure model are still consistent (as if they were estimating the population parameters in the correlation structures). However, the chi-square test statistic as reported previously is not correct.

Note that using the CORR option in the PROC CALIS statement will not solve the problem. By specifying the CORR option you merely request PROC CALIS to use the correlation matrix directly as a covariance matrix in the objective function for model fitting. It still would not constrain the fitting of the diagonal elements to 1 during estimation.

In the next section, a solution to the correlation analysis problem is suggested. It is not claimed that this is the only solution or the best solution. Alternative treatments of the problem are possible.

### Fitting the Correct Correlation Structures

This main idea of this solution is to embed the intended correlation structures (with correct constraints on the diagonal elements of the correlation matrix) into a covariance structure model so that the estimation methods of PROC CALIS can be applied legitimately to the specially constructed covariance structures.

First, the issue of the fixed ones on the diagonal of the correlation structure model is addressed. That is, the diagonal elements of the correlation structures represented by $(BB' + \Psi)$ must be fitted by ones. This can be accomplished by constraining the error variances as dependent parameters of the loadings, as shown in the following:

$$\Psi_{jj} = 1 - b_{j1}^2 - b_{j2}^2, \quad j = 1, \ldots, 6$$

Other constraints might also serve the purpose, but the proposed constraints here are the most convenient and intuitive.

Now, due to the fact that discrepancy functions used in PROC CALIS are derived for covariance matrices rather than correlation matrices, PROC CALIS is essentially set up for analyzing covariance structures (with or without mean structures), but not correlation structures. Hence, the statistical theory behind PROC CALIS applies to covariance structure analysis, but it might not generalize to correlation structure analysis in all situations. Despite that, with some manipulations PROC CALIS can fit the correct correlation structures to
the current data without compromising the statistical theory. These manipulations are now discussed. Recall that the correlation structures are represented by:

\[ P = BB' + \Psi \]

As before, in the \( B \) matrix, there are six linear constraints on the factor loadings. In addition, the diagonal elements of \((BB' + \Psi)\) are constrained to ones, as done by defining the error variances as dependent parameters of the loadings in the preceding equation. To analyze the correlation structures by using PROC CALIS, a covariance structure model with such correlation structures embedded is now specified. That is, the covariance structure to be fitted by PROC CALIS is as follows:

\[ \Sigma = DPD' = D(BB' + \Psi)D' \]

where \( D \) is a 6 x 6 diagonal matrix containing the population standard deviations for the manifest variables. Theoretically, it is legitimate that you analyze this covariance structure model for studying the embedded correlation structures. In addition, it does not matter whether your input matrix is a correlation or covariance matrix, or any rescaled covariance matrix (by multiplying any variables by any positive constants). You would get correct results if you could somehow specify these covariance structures correctly in PROC CALIS. However, there seems to be nowhere in PROC CALIS that you can specify the diagonal matrix \( D \) for the population standard deviations. So what can one do with this formulation? The answer is to rewrite the covariance structure model in a form similar to the usual confirmatory factor model, as presented in the following.

Let \( T = DB \) and \( K = D\Psi D' \). The covariance structure model of interest can now be rewritten as:

\[ \Sigma = TT' + K \]

This form of covariance structures implies a confirmatory factor model with factor loading matrix \( T \) and error covariance matrix \( K \). This confirmatory factor model can certainly be specified using the FACTOR modeling language, in much the same way you specify a confirmatory factor model in the preceding section. However, because you are actually more interested in estimating the basic set of parameters in matrices \( B \) and \( \Psi \) of the embedded correlation structures, you would define the model parameters as functions of this basic set of parameters of interest. This can be accomplished by using the PARAMETERS and the SAS programming statements.

All in all, you can use the following statements to set up such a confirmatory factor model with the desired correlation structures embedded.
First, you notice that specifications in the FACTOR and the PVAR statements are essentially unchanged from the previous specification, except that the parameters are named differently here to reflect different model matrices. In the current specification, the factor loading parameters in matrix $T$ are named with prefix ‘$t$,’ and the error variance parameters in matrix $K$ are named with prefix ‘$k$.’ Specification of these parameters reflects
the covariance structures. As you see in the last block of the SAS programming statements statements, all these parameters are functions of the correlation structure parameters in B, Ψ, and D.

Next, in the PARAMETERS statement, all correlation structure parameters are defined with initial values provided. These are the parameters of interest: alpha is used to define dependencies among loadings, d’s are the population standard deviations, b’s are the loading parameters, and psi’s are the error variance parameters. There are 25 parameters specified in this statement, but not all of them are free or independent.

In the first block of SAS programming statements, parameter dependencies or constraints on the correlation structures are specified. The first six statements realize the required linear relations among the factor loadings:

\[ b_{j2} = \alpha - b_{j1}, \quad j = 1, \ldots, 6 \]

The next six statements constrain the error variances so as to ensure that an embedded correlation structure model is being fitted. That is, each error variance is dependent on the corresponding loadings, as prescribed by the following equation:

\[ \Psi_{jj} = 1 - b_{j1}^2 - b_{j2}^2, \quad j = 1, \ldots, 6 \]

These twelve constraints reduce the number of independent parameters to 13, as expected.

The next block of SAS programming statements are essentially for relating the correlation structure parameters to the covariance structures that are specified in the FACTOR and the PVAR statements. These SAS programming statements realize the required relations: \( T = DB \) and \( K = D\Psi D' \), but in non-matrix forms:

\[ t_{ji} = d_j b_{ji}, \quad (j = 1, \ldots, 6; \quad i = 1, 2) \]
\[ k_{jj} = d_j d_j \Psi_{jj}, \quad (j = 1, \ldots, 6) \]

where \( d_j \) denotes the \( j \)th diagonal element of \( D \).

The fit summary is presented in Output 29.27.4. The chi-square test statistic is 14.63 with \( df = 8 \) (\( p = 0.067 \)). This shows that the previous chi-square test based on fitting a wrong covariance structure model is indeed questionable.

**Output 29.27.4** Model Fit of the Correlation Structures

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>14.6269</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>8</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.0668</td>
</tr>
</tbody>
</table>

 Estimates of the loadings and error variances are presented in Output 29.27.5. These estimates are for the covariance structure model with loading matrix \( T \) and error covariance matrix \( K \). They are rescaled versions of the correlation structure parameters and are not of primary interest themselves.
### Output 29.27.5 Estimates of Loadings and Error Variances

<table>
<thead>
<tr>
<th>Factor Loading Matrix</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>factor1</td>
<td>factor2</td>
<td></td>
</tr>
<tr>
<td>var1</td>
<td>0.3448</td>
<td>0.6367</td>
</tr>
<tr>
<td></td>
<td>[11]</td>
<td>[12]</td>
</tr>
<tr>
<td>var2</td>
<td>0.3200</td>
<td>0.6512</td>
</tr>
<tr>
<td></td>
<td>[21]</td>
<td>[22]</td>
</tr>
<tr>
<td>var3</td>
<td>0.4873</td>
<td>0.4778</td>
</tr>
<tr>
<td></td>
<td>[31]</td>
<td>[32]</td>
</tr>
<tr>
<td>var4</td>
<td>0.5703</td>
<td>0.3948</td>
</tr>
<tr>
<td></td>
<td>[41]</td>
<td>[42]</td>
</tr>
<tr>
<td>var5</td>
<td>0.7741</td>
<td>0.1964</td>
</tr>
<tr>
<td></td>
<td>[51]</td>
<td>[52]</td>
</tr>
<tr>
<td>var6</td>
<td>0.6778</td>
<td>0.3126</td>
</tr>
<tr>
<td></td>
<td>[61]</td>
<td>[62]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Factor Covariance Matrix</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>factor1</td>
<td>factor2</td>
<td></td>
</tr>
<tr>
<td>factor1</td>
<td>1.0000</td>
<td>0</td>
</tr>
<tr>
<td>factor2</td>
<td>0</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Error Variances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>var1</td>
</tr>
<tr>
<td>var2</td>
</tr>
<tr>
<td>var3</td>
</tr>
<tr>
<td>var4</td>
</tr>
<tr>
<td>var5</td>
</tr>
<tr>
<td>var6</td>
</tr>
</tbody>
</table>

The parameter estimates of the embedded correlation structures are shown in Output 29.27.6 as “additional” parameters.
Output 29.27.6  Estimates of Correlation Structure Parameters

<table>
<thead>
<tr>
<th>Additional Parameters</th>
<th>Type</th>
<th>Parameter</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Independent</td>
<td>alpha</td>
<td>0.97400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d1</td>
<td>1.00771</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d2</td>
<td>0.99712</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d3</td>
<td>0.99078</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d4</td>
<td>0.99085</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d5</td>
<td>0.99640</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d6</td>
<td>1.01687</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b11</td>
<td>0.34217</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b21</td>
<td>0.32095</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b31</td>
<td>0.49179</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b41</td>
<td>0.57553</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b51</td>
<td>0.77686</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b61</td>
<td>0.66659</td>
</tr>
<tr>
<td></td>
<td>Dependent</td>
<td>b12</td>
<td>0.63183</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b22</td>
<td>0.65305</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b32</td>
<td>0.48222</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b42</td>
<td>0.39848</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b52</td>
<td>0.19714</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b62</td>
<td>0.30742</td>
</tr>
<tr>
<td></td>
<td></td>
<td>psi1</td>
<td>0.48371</td>
</tr>
<tr>
<td></td>
<td></td>
<td>psi2</td>
<td>0.47051</td>
</tr>
<tr>
<td></td>
<td></td>
<td>psi3</td>
<td>0.52561</td>
</tr>
<tr>
<td></td>
<td></td>
<td>psi4</td>
<td>0.50998</td>
</tr>
<tr>
<td></td>
<td></td>
<td>psi5</td>
<td>0.35762</td>
</tr>
<tr>
<td></td>
<td></td>
<td>psi6</td>
<td>0.46116</td>
</tr>
</tbody>
</table>

Except for the population standard deviation parameter \( \delta \)'s, all other parameters estimated in the current model can be compared with those from the previous fitting of an incorrect covariance structure model. Although estimates in the current model do not differ very much from those in the previous specification, it is at least reassuring that they are obtained from fitting a correctly specified covariance structure model with the intended correlation structures embedded.
Example 29.28: Multiple-Group Model for Purchasing Behavior

In this example, data were collected from customers who made purchases from a retail company during years 2002 and 2003. A two-group structural equation model is fitted to the data.

The variables are:

- **Spend02**: total purchase amount in 2002
- **Spend03**: total purchase amount in 2003
- **Courtesy**: rating of the courtesy of the customer service
- **Responsive**: rating of the responsiveness of the customer service
- **Helpful**: rating of the helpfulness of the customer service
- **Delivery**: rating of the timeliness of the delivery
- **Pricing**: rating of the product pricing
- **Availability**: rating of the product availability
- **Quality**: rating of the product quality

For the ratings scales, nine-point scales were used. Customers could respond 1 to 9 on these scales, with 1 representing “extremely unsatisfied” and 9 representing “extremely satisfied.” Data were collected from two different regions, which are labeled as Region 1 \((N = 378)\) and Region 2 \((N = 423)\), respectively. The ratings were collected at the end of year 2002 so that they represent customers’ purchasing experience in year 2002.

The central questions of the study are:

- How does the overall customer service affect the current purchases and predict future purchases?
- How does the overall product quality affect the current purchases and predict future purchases?
- Do current purchases predict future purchases?
- Do the two regions have different structural models for predicting the purchases?

In stating these research questions, you use several constructs that might or might not correspond to objective measurements. Current and future purchases are certainly measurable directly by the spending of the customers. That is, because customer service and product satisfaction and quality were surveyed between 2002 and 2003, Spend02 represents current purchases and Spend03 represents future purchases in the study. Both variables Spend02 and Spend03 are objective measurements without measurement errors. All you need to do is to extract the information from the transaction records. But how about hypothetical constructs such as customer service quality and product quality? How would you measure them in the model?
In measuring these hypothetical constructs, you might ask customers’ perception about the service or product quality directly in a single question. A simple survey with two questions about the customer service and product qualities could then be what you need. These questions are called indicators (or indicator variables) of the underlying constructs. However, using just one indicator (question) for each of these hypothetical constructs would be quite unreliable—that is, measurement errors might dominate in the data collection process. Therefore, multiple indicators are usually recommended for measuring such hypothetical constructs.

There are two main advantages of using multiple indicators for hypothetical constructs. The first one is conceptual and the other is statistical and mathematical.

First, hypothetical constructs might conceptually be multifaceted themselves. Measuring a hypothetical construct by a single indicator does not capture the full meaning of the construct. For example, the product quality might refer to the durability of the product, the outlook of the product, the pricing of the product, and the availability of product, among others. The customer service quality might refer to the politeness of the customer service, the timeliness of the delivery, and the responsiveness of customer services, among others. Therefore, multiple indicators for a single hypothetical construct might be necessary if you want to cover the multifaceted aspects of a given hypothetical construct.

Second, from a statistical point of view, the reliability would be higher if you combine correlated indicators for a construct than if you use a single indicator only. Therefore, combining correlated indicators would lead to more accurate and reliable results.

One way to combine correlated indicators is to use a simple sum of them to represent the underlying hypothetical construct. However, a better way is to use the structural equation modeling technique that represents each indicator (variable) as a function of the underlying hypothetical construct plus an error term. In structural equation modeling, hypothetical constructs are constructed as latent factors, which are unobserved systematic (that is, non-error) variables. Theoretically, latent factors are free from measurement errors, and so the estimation through the structural equation modeling technique is more accurate than if you just use simple sums of indicators to represent hypothetical constructs. Therefore, a structural equation modeling approach is the method of the choice in the current analysis.

In practice, you must also make sure that there are enough indicators for the identification of the underlying latent factor, and hence the identification of the entire model. Necessary and sufficient rules for identification are very complicated to describe and are out of the scope of the current discussion (however, see Bollen 1989b for discussions of identification rules for various classes of structural equation models). Some simple rules of thumb might be useful as a guidance. For example, for unconstrained situations, you should at least have three indicators (variables) measured for a latent factor. Unfortunately, these rules of thumb do not guarantee identification in every case.

In this example, Service and Product are latent factors in the structural equation model which represent service and product qualities, respectively. In the model, these two latent factors are reflected by the ratings of the customers. Ratings on the Courtesy, Responsive, Helpful, and Delivery scales are indicators of Service. Ratings on the Pricing, Availability, and Quality scales are indicators of Product (that is, product quality).
A Path Diagram

A path diagram shown in Figure 29.27 represents the structural equation model for the purchase behavior. Observed or manifest variables are represented by rectangles, and latent variables are represented by ovals. As mentioned, two latent variables (factors), Service and Product, are created as overall measures of customer service and product qualities, respectively.

The left part of the diagram represents the measurement model of the latent variables. The Service factor has four indicators: Courtesy, Responsive, Helpful, and Delivery. The path coefficients to these observed variables from the Service factor are \( b_1, b_2, b_3, \) and \( b_4 \), respectively. Similarly, the Product variable has three indicators: Pricing, Availability, and Quality, with path coefficients \( b_5, b_6, \) and \( b_7 \), respectively.

The two latent factors are predictors of the purchase amounts Spend02 and Spend03. In addition, Spend02 also serves as a predictor of Spend03. Path coefficients (effects) for this part of functional relationships are represented by \( a_1 \)–\( a_5 \) in the diagram.

Each variable in the path diagram has a variance parameter. For endogenous or dependent variables, which serve as outcome variables in the model, the variance parameters are the error variances that are not accounted for by the predictors. For example, in the current model all observed variables are endogenous variables. The double-headed arrows that are attached to these variables represent error variances. In the diagram, \( \theta_1 \) to \( \theta_9 \) are the names of these error variance parameters. For exogenous or independent variables, which never serve as outcome variables in the model, the variance parameters are the (total) variances of these variables. For example, in the diagram the double-headed arrows that are attached to Service and Product represent the variances of these two latent variables. In the current model, both of these variances are fixed at one.
When the double-headed arrows point to two variables, they represent covariances in the path diagram. For example, in Figure 29.27 the covariance between Service and Product is represented by the parameter $\phi$.

**The Basic Path Model Specification**

For the moment, it is hypothesized that both ‘Region 1’ and ‘Region 2’ data are fitted by the same model as shown in Figure 29.27. Once the path diagram is drawn, it is readily translated into the PATH modeling language. See the PATH statement on page 1322 for details about how to use the PATH modeling language to specify structural equation models.

To represent all the features in the path diagram in the PATH model language, you can use the following specification:

```
path
  Service ===> Spend02  = a1,
  Service ===> Spend03  = a1,
  Product ===> Spend02  = a3,
  Product ===> Spend03  = a4,
  Spend02 ===> Spend03  = a5,
  Service ===> Courtesy  = b1,
  Service ===> Responsive = b2,
  Service ===> Helpful   = b3,
  Service ===> Delivery  = b4,
  Product ===> Pricing   = b5,
  Product ===> Availability = b6,
  Product ===> Quality   = b7;
```

```
pvar
  Courtesy  Responsive Helpful
  Delivery   Pricing
  Availability Quality = theta01-theta07,
  Spend02 = theta08,
  Spend03 = theta09,
  Service Product = 2 * 1.;
```

```
pcov
  Service Product = phi;
```

The PATH statement captures all the path coefficient specifications and the direction of the paths (single-headed arrows) in the path diagram. The first five paths define how $\text{Spend02}$ and $\text{Spend03}$ are predicted from the latent variables $\text{Service}$, $\text{Product}$, and $\text{Spend02}$. The next seven paths define the measurement model, which shows how the latent variables in the model relate to the observed indicator variables.

The PVAR statement captures the specification of the error variances and the variances of exogenous variables (that is, the double-headed arrows in the path diagram). The PCOV statement captures the specification of covariance between the two latent variables in the model (which is represented by the double-headed arrow that connects $\text{Service}$ and $\text{Product}$ in the path diagram).
You can also use the following simpler version of the PATH model specification for the path diagram:

```plaintext
path
  Service ===> Spend02  Spend03 ,
  Product ===> Spend02  Spend03 ,
  Spend02 ===> Spend03 ,
  Service ===> Courtesy Responsive Helpful Delivery ,
  Product ===> Pricing Availability Quality ;
pvar
 Courtesy Responsive Helpful Delivery Pricing Availability Quality Spend02 Spend03,
  Service Product = 2 * 1. ;
pcov
  Service Product;
```

There are two simplifications in this PATH model specification. First, you do not need to specify the parameter names if they are unconstrained in the model. For example, parameter $a_1$ in the model is unique to the path effect from Service to Spend02. You do not need to name this effect because it is not constrained to be the same as any other parameter in the model. Similar, all the path coefficients (effects), error variances, and covariances in the path diagram are not constrained. Therefore, you can omit all the corresponding parameter name specifications in the PATH model specification. The only exceptions are the variances of Service and Product. Both are fixed constants 1 in the path diagram, and so you must specify them explicitly in the PVAR statement.

Second, you use a condensed way to specify the paths. In the first three path entries of the PATH statement, you specify how Spend02 and Spend03 are predicted from the latent variables Service, Product, and Spend02. Notice that in each path entry, you can define more than one path (single-headed arrow relationship). For example, in the first path entry, you specify two paths: one is $Service ===> Spend02$ and the other is $Service ===> Spend03$. In the last two path entries of the PATH statement, you define the relationships between the two latent constructs Spend and Service and their measured indicators. Each of these path entries specifies multiple paths (single-headed arrow relationships).

You use this simplified PATH specifications in the subsequent analysis.

**A Restrictive Model with Invariant Mean and Covariance Structures**

In this section, you fit a mean and covariance structure model to the data from two regions, as shows in the following DATA steps:
To include the analysis of the mean structures, you need to introduce the mean and intercept parameters in the model. Although various researchers propose some representation schemes that include the mean parameters in the path diagram, the mean parameters are not depicted in Figure 29.27. The reason is that representing the mean and intercept parameters in the path diagram would usually obscure the “causal” paths, which are of primary interest. In addition, it is a simple matter to specify the mean and intercept parameters in the MEAN statement without the help of a path diagram when you follow these principles:

- Each variable in the path diagram has a mean parameter that can be specified in the MEAN statement. For an exogenous variable, the mean parameter refers to the variable mean. For an endogenous variable, the mean parameter refers to the intercept of the variable.

- The means of exogenous observed variables are free parameters by default. The means of exogenous latent variables are fixed zeros by default.
• The intercepts of endogenous observed variables are free parameters by default. The intercepts of endogenous latent variables are fixed zeros by default.

• The total number of mean parameters should not exceed the number of observed variables.

Because all nine observed variables are endogenous (each has at least one single-headed arrow pointing to it) in the path diagram, you can specify these nine intercepts in the MEAN statement, as shown in the following specification:

```
mean
   Courtesy Responsive Helpful Delivery Pricing
   Availability Quality Spend02 Spend03;
```

However, the intercepts of endogenous observed variables are already free parameters by default and this MEAN statement specification is not necessary for the current situation. For the means of the latent variables Service and Product, you do not have any theoretical reasons to set them other than the default fixed zero. Hence, you do not need to set these mean parameters explicitly either. Consequently, to include the analysis of the mean structures with these default mean parameters, all you need to specify the MEANSTR option in the PROC CALIS statement, as shown in the following specification of the fitting of a constrained two-group model for the purchase data:

```
proc calis meanstr;
   group 1 / data=region1 label="Region 1" nobs=378;
   group 2 / data=region2 label="Region 2" nobs=423;
   model 1 / group=1,2;
   path
      Service ===> Spend02 Spend03 ,
      Product ===> Spend02 Spend03 ,
      Spend02 ===> Spend03 ,
      Service ===>Courtesy Responsive
      Helpful Delivery ,
      Product ===> Pricing Availability
             Quality ;
   pvar
      Courtesy Responsive Helpful Delivery Pricing
      Availability Quality Spend02 Spend03,
      Service Product = 2 * 1.;
   pcov
      Service Product;
run;
```
You use the `GROUP` statements to specify the data for the two regions. Using the `DATA=` options in the `GROUP` statements, you assign the ‘Region 1’ data to Group 1 and the ‘Region 2’ data to Group 2. You label the two groups by the `LABEL=` options. Because the number of observations is not defined in the data sets, you use the `NOBS=` options in the `GROUP` statements to provide this information.

In the `MODEL` statement, you specify in the `GROUP=` option that both Groups 1 and 2 are fitted by the same model—model 1. Next, the path model is specified. As discussed before, you do not need to specify the default mean parameters by using the `MEAN` statement because the `MEANSTR` option in the `PROC CALIS` statement already indicates the analysis of mean structures.

Output 29.28.1 presents a summary of modeling information. Each group is listed with its associated data set, number of observations, and its corresponding model and the model type. In the current analysis, the same model is fitted to both groups. Next, a table for the types of variables is presented. As intended, all nine observed (manifest) variables are endogenous, and all latent variables are exogenous in the model.

**Output 29.28.1 Modeling Information and Variables**

<table>
<thead>
<tr>
<th>Modeling Information</th>
<th>Maximum Likelihood Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>Label</td>
</tr>
<tr>
<td>1</td>
<td>Region 1</td>
</tr>
<tr>
<td>2</td>
<td>Region 2</td>
</tr>
</tbody>
</table>

**Model 1. Variables in the Model**

<table>
<thead>
<tr>
<th>Endogenous</th>
<th>Manifest</th>
<th>Availability</th>
<th>Courtesy</th>
<th>Delivery</th>
<th>Helpful</th>
<th>Pricing</th>
<th>Quality</th>
<th>Responsive</th>
<th>Spend02</th>
<th>Spend03</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latent</td>
<td>Latent</td>
<td>Product</td>
<td>Service</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Number of Endogenous Variables = 9**

**Number of Exogenous Variables = 2**
The optimization converges. The fit summary table is presented in Output 29.28.2.

### Output 29.28.2 Fit Summary

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling Info</td>
<td></td>
</tr>
<tr>
<td>Number of Observations</td>
<td>801</td>
</tr>
<tr>
<td>Number of Variables</td>
<td>9</td>
</tr>
<tr>
<td>Number of Moments</td>
<td>108</td>
</tr>
<tr>
<td>Number of Parameters</td>
<td>31</td>
</tr>
<tr>
<td>Number of Active Constraints</td>
<td>0</td>
</tr>
<tr>
<td>Baseline Model Function Value</td>
<td>0.5003</td>
</tr>
<tr>
<td>Baseline Model Chi-Square</td>
<td>399.7468</td>
</tr>
<tr>
<td>Baseline Model Chi-Square DF</td>
<td>72</td>
</tr>
<tr>
<td>Pr &gt; Baseline Model Chi-Square</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>Absolute Index</td>
<td></td>
</tr>
<tr>
<td>Fit Function</td>
<td>3.5297</td>
</tr>
<tr>
<td>Chi-Square</td>
<td>2820.2504</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>77</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>Z-Test of Wilson &amp; Hilferty</td>
<td>43.2575</td>
</tr>
<tr>
<td>Hoelter Critical N</td>
<td>29</td>
</tr>
<tr>
<td>Root Mean Square Residual (RMR)</td>
<td>28.2208</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>2.1367</td>
</tr>
<tr>
<td>Goodness of Fit Index (GFI)</td>
<td>0.9996</td>
</tr>
<tr>
<td>Parsimony Index</td>
<td></td>
</tr>
<tr>
<td>Adjusted GFI (AGFI)</td>
<td>0.9995</td>
</tr>
<tr>
<td>Parsimonious GFI</td>
<td>1.0690</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.2986</td>
</tr>
<tr>
<td>RMSEA Lower 90% Confidence Limit</td>
<td>0.2892</td>
</tr>
<tr>
<td>RMSEA Upper 90% Confidence Limit</td>
<td>0.3081</td>
</tr>
<tr>
<td>Probability of Close Fit</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>2882.2504</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>3058.5121</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>3027.5121</td>
</tr>
<tr>
<td>McDonald Centrality</td>
<td>0.1804</td>
</tr>
<tr>
<td>Incremental Index</td>
<td></td>
</tr>
<tr>
<td>Bentler Comparative Fit Index</td>
<td>0.0000</td>
</tr>
<tr>
<td>Bentler-Bonett NFI</td>
<td>-6.0551</td>
</tr>
<tr>
<td>Bentler-Bonett Non-normed Index</td>
<td>-6.8265</td>
</tr>
<tr>
<td>Bollen Normed Index Rho1</td>
<td>-5.5970</td>
</tr>
<tr>
<td>Bollen Non-normed Index Delta2</td>
<td>-7.4997</td>
</tr>
<tr>
<td>James et al. Parsimonious NFI</td>
<td>-6.4756</td>
</tr>
</tbody>
</table>
The model chi-square statistic is 2820.25. With df = 77 and p < .0001, the null hypothesis for the mean and covariance structures is rejected. All incremental fit indices are negative. These negative indices indicate a bad model fit, as compared with the independence model. The same fact can be deduced by comparing the chi-square values of the baseline model and the fitted model. The baseline model has five degrees of freedom less (five parameters more) than the structural model but the chi-square value is only 399.747, much less than the model fit chi-square value of 2820.25. Because variables in social and behavioral sciences are almost always expected to correlate with each other, a structural model that explains relationships even worse than the baseline model is deemed inappropriate for the data. The RMSEA for the structural model is 0.2986, which also indicates a bad model fit. However, the GFI, AGFI, and parsimonious GFI indicate good model fit, which is a little surprising given the fact that all other indices indicate the opposite and the overall model is pretty restrictive in the first place.

There are some warnings in the output:

**WARNING: Model 1.** The estimated error variance for variable Spend02 is negative.

**WARNING: Model 1.** Although all predicted variances for the observed and latent variables are positive, the corresponding predicted covariance matrix is not positive definite. It has one negative eigenvalue.

PROC CALIS routinely checks the properties of the estimated variances and the predicted covariance matrix. It issues warnings when there are problems. In this case, the error variance estimate of Spend02 is negative, and the predicted covariance matrix for the observed and latent variables is not positive definite and has one negative eigenvalue. You can inspect **Output 29.28.3**, which shows the variance parameter estimates of the variables.

**Output 29.28.3**  Variance Estimates

| Variance Type | Variable | Parameter | Estimate | Standard Error | t Value | Pr > |t| |
|---------------|----------|-----------|----------|----------------|---------|-------|
| Error         | Courtesy | _Parm13   | 2.59181  | 0.13600        | 19.0574 | <.0001|
|               | Responsive| _Parm14   | 2.92423  | 0.15325        | 19.0821 | <.0001|
|               | Helpful   | _Parm15   | 2.44625  | 0.12320        | 19.8566 | <.0001|
|               | Delivery  | _Parm16   | 3.53408  | 0.18169        | 19.4510 | <.0001|
|               | Pricing   | _Parm17   | 2.52948  | 0.12784        | 19.7869 | <.0001|
|               | Availability | _Parm18 | 1.57410  | 0.16884        | 9.3230  | <.0001|
|               | Quality   | _Parm19   | 2.41658  | 0.13230        | 18.2661 | <.0001|
|               | Spend02   | _Parm20   | -14.40124 | -0.8507       | 16.92863 | 0.3949|
|               | Spend03   | _Parm21   | 22.79309 | 5.75120        | 3.9632  | <.0001|

The error variance estimate for Spend02 is −14.40, which is negative and might have led to the negative eigenvalue problem in the predicted covariance matrix for the observed and latent variables.
A Model with Unconstrained Parameters for the Two Regions

With all the bad model fit indications and the problematic predicted covariance matrix for the latent variables, you might conclude that an overly restricted model has been fit. Region 1 and Region 2 might not share exactly the same set of parameters. How about fitting a model at the other extreme with all parameters unconstrained for the two groups (regions)? Such a model can be easily specified, as shown in the following statements:

```plaintext
proc calis meanstr;
    group 1 / data=region1 label="Region 1" nobs=378;
    group 2 / data=region2 label="Region 2" nobs=423;
    model 1 / group=1;
    path
        Service ===> Spend02 Spend03 ,
        Product ===> Spend02 Spend03 ,
        Spend02 ===> Spend03 ,
        Service ===> Courtesy Responsive Helpful Delivery ,
        Product ===> Pricing Availability Quality ;
    pvar
        Courtesy Responsive Helpful Delivery Pricing
        Availability Quality Spend02 Spend03,
        Service Product = 2 * 1.;
    pcov
        Service Product;
    model 2 / group=2;
    refmodel 1/ allnewparms;
run;
```

Unlike the previous specification, in the current specification Group 2 is now fitted by a new model labeled as Model 2. This model is based on Model 1, as specified in `REFMODEL` statement. The `ALLNEWPARMS` option in the `REFMODEL` statement request that all parameters specified in Model 1 be renamed so that they become new parameters in Model 2. As a result, this specification gives different sets of estimates for Model 1 and Model 2, although both models have the same path structures and a comparable set of parameters.

The optimization converges without problems. The fit summary table is displayed in Output 29.28.4. The chi-square statistic is 29.613 ($df = 46, p = .97$). The theoretical model is not rejected. Many other measures of fit also indicate very good model fit. For example, the GFI, AGFI, Bentler CFI, Bentler-Bonett NFI, and Bollen nonnormed index delta2 are all close to one, and the RMSEA is close to zero.
### Output 29.28.4 Fit Summary

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fit Summary</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Modeling Info</strong></td>
<td></td>
</tr>
<tr>
<td>Number of Observations</td>
<td>801</td>
</tr>
<tr>
<td>Number of Variables</td>
<td>9</td>
</tr>
<tr>
<td>Number of Moments</td>
<td>108</td>
</tr>
<tr>
<td>Number of Parameters</td>
<td>62</td>
</tr>
<tr>
<td>Number of Active Constraints</td>
<td>0</td>
</tr>
<tr>
<td>Baseline Model Function Value</td>
<td>0.5003</td>
</tr>
<tr>
<td>Baseline Model Chi-Square</td>
<td>399.7468</td>
</tr>
<tr>
<td>Baseline Model Chi-Square DF</td>
<td>72</td>
</tr>
<tr>
<td>Pr &gt; Baseline Model Chi-Square</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td><strong>Absolute Index</strong></td>
<td></td>
</tr>
<tr>
<td>Fit Function</td>
<td>0.0371</td>
</tr>
<tr>
<td>Chi-Square</td>
<td>29.6131</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>46</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.9710</td>
</tr>
<tr>
<td>Z-Test of Wilson &amp; Hilferty</td>
<td>-1.8950</td>
</tr>
<tr>
<td>Hoelter Critical N</td>
<td>1697</td>
</tr>
<tr>
<td>Root Mean Square Residual (RMR)</td>
<td>0.0670</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0220</td>
</tr>
<tr>
<td>Goodness of Fit Index (GFI)</td>
<td>1.0000</td>
</tr>
<tr>
<td><strong>Parsimony Index</strong></td>
<td></td>
</tr>
<tr>
<td>Adjusted GFI (AGFI)</td>
<td>1.0000</td>
</tr>
<tr>
<td>Parsimonious GFI</td>
<td>0.6389</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
<td>0.0000</td>
</tr>
<tr>
<td>RMSEA Lower 90% Confidence Limit</td>
<td>0.0000</td>
</tr>
<tr>
<td>RMSEA Upper 90% Confidence Limit</td>
<td>0.0000</td>
</tr>
<tr>
<td>Probability of Close Fit</td>
<td>1.0000</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
<td>153.6131</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
<td>506.1365</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
<td>444.1365</td>
</tr>
<tr>
<td>McDonald Centrality</td>
<td>1.0103</td>
</tr>
<tr>
<td><strong>Incremental Index</strong></td>
<td></td>
</tr>
<tr>
<td>Bentler Comparative Fit Index</td>
<td>1.0000</td>
</tr>
<tr>
<td>Bentler-Bonett NFI</td>
<td>0.9259</td>
</tr>
<tr>
<td>Bentler-Bonett Non-normed Index</td>
<td>1.0783</td>
</tr>
<tr>
<td>Bollen Normed Index Rho1</td>
<td>0.8840</td>
</tr>
<tr>
<td>Bollen Non-normed Index Delta2</td>
<td>1.0463</td>
</tr>
<tr>
<td>James et al. Parsimonious NFI</td>
<td>0.5916</td>
</tr>
</tbody>
</table>
Notice that because there are no constraints between the two models for the groups, you might have fit the two sets of data by the respective models separately and gotten exactly the same results as in the current analysis. For example, you get two model fit chi-square values from separate analyses. Adding up these two chi-squares gives you the same overall chi-square as in Output 29.28.4.

PROC CALIS also provides a table for comparing the relative model fit of the groups. In Output 29.28.5, basic modeling information and some measures of fit for the two groups are shown along with the corresponding overall measures.

**Output 29.28.5**  Fit Comparison among Groups

<table>
<thead>
<tr>
<th>Fit Index</th>
<th>Overall</th>
<th>Region 1</th>
<th>Region 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling Info</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Observations</td>
<td>801</td>
<td>378</td>
<td>423</td>
</tr>
<tr>
<td>Number of Variables</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Number of Moments</td>
<td>108</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>Number of Parameters</td>
<td>62</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>Number of Active Constraints</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Baseline Model Function Value</td>
<td>0.5003</td>
<td>0.4601</td>
<td>0.5363</td>
</tr>
<tr>
<td>Baseline Model Chi-Square</td>
<td>399.7468</td>
<td>173.4482</td>
<td>226.2986</td>
</tr>
<tr>
<td>Baseline Model Chi-Square DF</td>
<td>72</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>Fit Index</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit Function</td>
<td>0.0371</td>
<td>0.0023</td>
<td>0.0681</td>
</tr>
<tr>
<td>Percent Contribution to Chi-Square</td>
<td>100</td>
<td>3</td>
<td>97</td>
</tr>
<tr>
<td>Root Mean Square Residual (RMR)</td>
<td>0.0670</td>
<td>0.0172</td>
<td>0.0907</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0220</td>
<td>0.0057</td>
<td>0.0298</td>
</tr>
<tr>
<td>Goodness of Fit Index (GFI)</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>Bentler-Bonett NFI</td>
<td>0.9259</td>
<td>0.9950</td>
<td>0.8730</td>
</tr>
</tbody>
</table>
When you examine the results of this table, the first thing you have to realize is that in general the group statistics are not independent. For example, although the overall chi-square statistic can be written as the weighted sum of fit functions of the groups, in general it does not imply that the individual terms are statistically independent. In the current two-group analysis, the overall chi-square is written as

\[ T = (N_1 - 1) f_1 + (N_2 - 1) f_2 \]

where \( N_1 \) and \( N_2 \) are sample sizes for the groups and \( f_1 \) and \( f_2 \) are the discrepancy functions for the groups. Even though \( T \) is chi-square distributed under the null hypothesis, in general the individual terms \((N_1 - 1) f_1\) and \((N_2 - 1) f_2\) are not chi-square distributed under the same null hypothesis. So when you compare the group fits by using the statistics in Output 29.28.5, you should treat those as descriptive measures only.

The current model is a special case where \( f_1 \) and \( f_2 \) are actually independent of each other. The reason is that there are no constrained parameters for the models fitted to the two groups. This would imply that the individual terms \((N_1 - 1) f_1\) and \((N_2 - 1) f_2\) are chi-square distributed under the null hypothesis. Nonetheless, this fact is not important to the group comparison of the descriptive statistics in Output 29.28.5. The values of \( f_1 \) and \( f_2 \) are shown in the row labeled “Fit Function.” Group 1 (Region 1) is fitted better by its model \((f_1 = 0.0023)\) than is Group 2 (Region 2) by its model \((f_2 = 0.0681)\). Next, the percentage contributions to the overall chi-square statistic for the two groups are shown. Group 1 contributes only 3% \((= (N_1 - 1) f_1 / T \times 100\%)\) while Group 2 contributes 97%. Other measures like RMR, SRMR, and Bentler-Bonett NFI show that Group 1 data are fitted better. The GFI’s show equal fits for the two groups, however.

Despite a very good fit, the current model is not intended to be the final model. It was fitted mainly for illustration purposes. The next section considers a partially constrained model for the two groups of data.

**A Model with Partially Constrained Parameters for the Two Regions**

For multiple-group analysis, cross-group constraints are of primary interest and should be explored whenever appropriate. The first fitting with all model parameters constrained for the groups has been shown to be too restrictive, while the current model with no cross-group constraints fits very well—so well that it might have overfit unnecessarily. A multiple-group model between these extremes is now explored. The following statements specify such a partially constrained model:
Example 29.28: Multiple-Group Model for Purchasing Behavior

proc calis meanstr modification;
group 1 / data=region1 label="Region 1" nobs=378;
group 2 / data=region2 label="Region 2" nobs=423;
model 3 / label="Model for References Only";
  path
    Service ===> Spend02 Spend03 ,
    Product ===> Spend02 Spend03 ,
    Spend02 ===> Spend03 ,
    Service ===> Courtesy Responsive Helpful Delivery ,
    Product ===> Pricing Availability Quality ;
pvar
    Courtesy Responsive Helpful Delivery Pricing Availability Quality Spend02 Spend03,
    Service Product = 2 * 1.;
pcov
    Service Product;
model 1 / groups=1;
  refmodel 3;
  mean
    Spend02 Spend03 = G1_InterSpend02 G1_InterSpend03,
    Courtesy Responsive Helpful Delivery Pricing Availability Quality = G1_intercept01-G1_intercept07;
model 2 / groups=2;
  refmodel 3;
  mean
    Spend02 Spend03 = G2_InterSpend02 G2_InterSpend03,
    Courtesy Responsive Helpful Delivery Pricing Availability Quality = G2_intercept01-G2_intercept07;
simtests
    SpendDiff = (Spend02Diff Spend03Diff)
    MeasurementDiff = (CourtesyDiff ResponsiveDiff HelpfulDiff DeliveryDiff PricingDiff AvailabilityDiff QualityDiff);
    Spend02Diff = G2_InterSpend02 - G1_InterSpend02;
    Spend03Diff = G2_InterSpend03 - G1_InterSpend03;
    CourtesyDiff = G2_intercept01 - G1_intercept01;
    ResponsiveDiff = G2_intercept02 - G1_intercept02;
    HelpfulDiff = G2_intercept03 - G1_intercept03;
    DeliveryDiff = G2_intercept04 - G1_intercept04;
    PricingDiff = G2_intercept05 - G1_intercept05;
    AvailabilityDiff = G2_intercept06 - G1_intercept06;
    QualityDiff = G2_intercept07 - G1_intercept07;
run;
In this specification, you use a special model definition. Model 3 serves as a reference model. You are not going to fit this model directly to any data set, but the specifications of other two models makes reference to it. Model 3 is no different from the basic path model specification used in preceding examples. The PATH model specification reflects the path diagram in Figure 29.27.

Region 1 is fitted by Model 1, which makes reference to Model 3 by using the REFMODEL statement. In addition, you add the MEAN statement specification. You now specify the intercept parameters explicitly by using the parameter names G1_intercept01–G1_intercept07, G1_InterSpend02, and G1_InterSpend03. In previous examples, these intercept parameters are set by default by PROC CALIS. This explicit parameter naming serves the purpose of distinguishing these parameters from those for Model 2.

Region 2 is fitted by Model 2, which also refers to Model 3 by using the REFMODEL statement. You also specify a MEAN statement for this model with explicit specifications of the intercept parameters. You name these intercepts G2_intercept01–G2_intercept07, G2_InterSpend02, and G2_InterSpend03. The G2 prefix distinguishes these parameters from the corresponding intercept parameters in the parent model. All in all, this means that both Models 1 and 2 refers to Model 3, except that Model 2 uses a different set of intercept parameters. In other words, in this multiple-group model the covariance structures for the two regions are constrained to be the same, while the means structures are allowed to be unconstrained.

You request additional statistics or tests in the current PROC CALIS analysis. The MODIFICATION option in the PROC CALIS statement requests that the Lagrange multiplier tests and Wald tests be conducted. The Lagrange multiplier tests provide information about which constrained or fixed parameters could be freed or added so as to improve the overall model fit. The Wald tests provide information about which existing parameters could be fixed at zeros (eliminated) without significantly affecting the overall model fit. These tests are discussed in more detail when the results are presented.

In the SIMTESTS statement, two simultaneous tests are requested. The first simultaneous test is named SpendDiff, which includes two parametric functions Spend02Diff and Spend03Diff. The second simultaneous test is named MeasurementDiff, which includes seven parametric functions: CourtesyDiff, ResponsiveDiff, HelpfulDiff, DeliveryDiff, PricingDiff, AvailabilityDiff, and QualityDiff. The null hypothesis of these simultaneous tests is of the form

\[ H_0 : t_i = 0 \quad (i = 1 \ldots k) \]

where \( k \) is the number of parametric functions within the simultaneous test. In the current analysis, the component parametric functions are defined in the SAS programming statements, which are shown in the last block of the specification. Essentially, all these parametric functions represent the differences of the mean or intercept parameters between the two models for groups. The first simultaneous test is intended to test whether the mean or intercept parameters in the structural models are the same, while the second simultaneous test is intended to test whether the mean parameters in the measurement models are the same.

The fit summary table is shown in Output 29.28.6.
Output 29.28.6  Fit Summary

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling Info</td>
</tr>
<tr>
<td>Number of Observations</td>
</tr>
<tr>
<td>Number of Variables</td>
</tr>
<tr>
<td>Number of Moments</td>
</tr>
<tr>
<td>Number of Parameters</td>
</tr>
<tr>
<td>Number of Active Constraints</td>
</tr>
<tr>
<td>Baseline Model Function Value</td>
</tr>
<tr>
<td>Baseline Model Chi-Square</td>
</tr>
<tr>
<td>Baseline Model Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Baseline Model Chi-Square</td>
</tr>
<tr>
<td>Absolute Index</td>
</tr>
<tr>
<td>Fit Function</td>
</tr>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
<tr>
<td>Z-Test of Wilson &amp; Hilferty</td>
</tr>
<tr>
<td>Hoelter Critical N</td>
</tr>
<tr>
<td>Root Mean Square Residual (RMR)</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
</tr>
<tr>
<td>Goodness of Fit Index (GFI)</td>
</tr>
<tr>
<td>Parsimony Index</td>
</tr>
<tr>
<td>Adjusted GFI (AGFI)</td>
</tr>
<tr>
<td>Parsimonious GFI</td>
</tr>
<tr>
<td>RMSEA Estimate</td>
</tr>
<tr>
<td>RMSEA Lower 90% Confidence Limit</td>
</tr>
<tr>
<td>RMSEA Upper 90% Confidence Limit</td>
</tr>
<tr>
<td>Probability of Close Fit</td>
</tr>
<tr>
<td>Akaike Information Criterion</td>
</tr>
<tr>
<td>Bozdogan CAIC</td>
</tr>
<tr>
<td>Schwarz Bayesian Criterion</td>
</tr>
<tr>
<td>McDonald Centrality</td>
</tr>
<tr>
<td>Incremental Index</td>
</tr>
<tr>
<td>Bentler Comparative Fit Index</td>
</tr>
<tr>
<td>Bentler-Bonett NFI</td>
</tr>
<tr>
<td>Bentler-Bonett Non-normed Index</td>
</tr>
<tr>
<td>Bollen Normed Index Rho1</td>
</tr>
<tr>
<td>Bollen Non-normed Index Delta2</td>
</tr>
<tr>
<td>James et al. Parsimonious NFI</td>
</tr>
</tbody>
</table>

The chi-square value is 107.55 (df = 68, p = 0.0016), which is statistically significant. The null hypothesis of the mean and covariance structures is rejected if an α-level at 0.01 or larger is chosen. However, in practical structural equation modeling, the chi-square test is not the only criterion, or even an important criterion, for evaluating model fit. The RMSEA estimate for the current model is 0.0382, which indicates a good fit. The probability level of close fit is 0.9275, indicating that a good population fit hypothesis (that is, population RMSEA < 0.05) cannot be rejected. The GFI, AGFI, and parsimonious GFI all indicate good fit. However, the incremental indices show only a respectable model fit.

Comparison of the model fit to the groups is shown in Output 29.28.7.
Output 29.28.7  Fit Comparison among Groups

<table>
<thead>
<tr>
<th></th>
<th>Overall</th>
<th>Region 1</th>
<th>Region 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations</td>
<td>801</td>
<td>378</td>
<td>423</td>
</tr>
<tr>
<td>Number of Variables</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Number of Moments</td>
<td>108</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>Number of Parameters</td>
<td>40</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>Number of Active Constraints</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Baseline Model Function Value</td>
<td>0.5003</td>
<td>0.4601</td>
<td>0.5363</td>
</tr>
<tr>
<td>Baseline Model Chi-Square</td>
<td>399.7468</td>
<td>173.4482</td>
<td>226.2986</td>
</tr>
<tr>
<td>Baseline Model Chi-Square DF</td>
<td>72</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>Fit Function</td>
<td>0.1346</td>
<td>0.1261</td>
<td>0.1422</td>
</tr>
<tr>
<td>Percent Contribution to Chi-Square</td>
<td>100</td>
<td>44</td>
<td>56</td>
</tr>
<tr>
<td>Root Mean Square Residual (RMR)</td>
<td>0.1577</td>
<td>0.1552</td>
<td>0.1599</td>
</tr>
<tr>
<td>Standardized RMR (SRMR)</td>
<td>0.0678</td>
<td>0.0792</td>
<td>0.0557</td>
</tr>
<tr>
<td>Goodness of Fit Index (GFI)</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>Bentler-Bonett NFI</td>
<td>0.7310</td>
<td>0.7260</td>
<td>0.7348</td>
</tr>
</tbody>
</table>

Looking at the percentage contribution to the chi-square, the Region 2 fitting shows a worse fit. However, this might be due to the larger sample size in Region 2. When comparing the fit of the two regions by using RMR, which does not take the sample size into account, the fitting of two groups are about the same. The standardized RMR even shows that Region 2 is fitted better. So, it seems to be safe to conclude that the models fit almost equally well (or badly) for the two regions.

The constrained parameter estimates for the two regions are shown in Output 29.28.8.
Example 29.28: Multiple-Group Model for Purchasing Behavior

Output 29.28.8 Estimates of Path Coefficients and Other Covariance Parameters

### Model 1. PATH List

| Path     | Parameter | Standard Error | t Value | Pr > |t| |
|----------|-----------|----------------|---------|------|---|
| Service  | Spend02   | _Parm01        | 0.37475 | 0.21318 | 1.7579 | 0.0788 |
| Service  | Spend03   | _Parm02        | 0.53851 | 0.20840 | 2.5840 | 0.0098 |
| Product  | Spend02   | _Parm03        | 0.80372 | 0.21939 | 3.6635 | 0.0002 |
| Product  | Spend03   | _Parm04        | 0.59879 | 0.22144 | 2.7041 | 0.0068 |
| Spend02  | Spend03   | _Parm05        | 0.08952 | 0.03694 | 2.4233 | 0.0154 |
| Service  | Courtesy  | _Parm06        | 0.72418 | 0.07989 | 9.0648 | <.0001 |
| Service  | Responsive| _Parm07        | 0.90452 | 0.08886 | 10.1797 | <.0001 |
| Service  | Helpful   | _Parm08        | 0.64969 | 0.07683 | 8.4557 | <.0001 |
| Service  | Delivery  | _Parm09        | 0.64473 | 0.09021 | 7.1468 | <.0001 |
| Product  | Pricing   | _Parm10        | 0.63452 | 0.07916 | 8.0160 | <.0001 |
| Product  | Availability | _Parm11 | 0.76737 | 0.08265 | 9.2852 | <.0001 |
| Product  | Quality   | _Parm12        | 0.79716 | 0.08922 | 8.9347 | <.0001 |

### Model 1. Variance Parameters

| Variance Type | Variable   | Parameter | Standard Error | t Value | Pr > |t| |
|---------------|------------|-----------|----------------|---------|------|---|
| Error         | Courtesy   | _Parm13   | 1.98374        | 0.13169 | 15.0638 | <.0001 |
|               | Responsive | _Parm14   | 2.02152        | 0.16159 | 12.5101 | <.0001 |
|               | Helpful    | _Parm15   | 1.96535        | 0.12263 | 16.0273 | <.0001 |
|               | Delivery   | _Parm16   | 2.97542        | 0.17049 | 17.4518 | <.0001 |
|               | Pricing    | _Parm17   | 1.93952        | 0.12326 | 15.7358 | <.0001 |
|               | Availability | _Parm18 | 1.63156        | 0.13067 | 12.4865 | <.0001 |
|               | Quality    | _Parm19   | 2.08849        | 0.15329 | 13.6246 | <.0001 |
|               | Spend02    | _Parm20   | 13.47066       | 0.71842 | 18.7505 | <.0001 |
|               | Spend03    | _Parm21   | 13.02883       | 0.68682 | 18.9697 | <.0001 |

### Model 1. Covariances Among Exogenous Variables

| Var1  | Var2  | Parameter | Standard Error | t Value | Pr > |t| |
|-------|-------|-----------|----------------|---------|------|---|
| Service | Product | _Parm22 | 0.33725 | 0.07061 | 4.7760 | <.0001 |

All parameter estimates but one are statistically significant at $\alpha = 0.05$. The parameter _Parm01, which represents the path coefficient from Service to Spend02, has a $t$ value of 1.76. This is only marginally significant. Although all these results bear the title of Model 1, these estimates are the same for Model 2, of which the corresponding results are not shown here.

The mean and intercept parameters for the two models (regions) are shown in Output 29.28.9.
All the mean and intercept estimates are statistically significant at $\alpha = 0.01$. Except for the fixed zero means for Service and Product, a quick glimpse of these mean and intercepts estimates shows a quite different pattern for the two models. Do these estimates truly differ beyond chance? The simultaneous tests of these parameter estimates shown in Output 29.28.10 can confirm this.

Output 29.28.10 shows two simultaneous tests, as requested in the original statements.
Output 29.28.10  Simultaneous Tests

<table>
<thead>
<tr>
<th>Simultaneous Test</th>
<th>Parametric Function</th>
<th>Function Value</th>
<th>DF</th>
<th>Chi-Square</th>
<th>p Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpendDiff</td>
<td></td>
<td>2</td>
<td></td>
<td>10458</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Spend02Diff</td>
<td>-27.25000</td>
<td>1</td>
<td></td>
<td>10225</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Spend03Diff</td>
<td>14.18831</td>
<td>1</td>
<td></td>
<td>185.86725</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>MeasurementDiff</td>
<td></td>
<td>7</td>
<td></td>
<td>1610</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>CourtesyDiff</td>
<td>-1.90000</td>
<td>1</td>
<td></td>
<td>286.58605</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>ResponsiveDiff</td>
<td>-1.99700</td>
<td>1</td>
<td></td>
<td>279.63659</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>HelpfulDiff</td>
<td>1.30300</td>
<td>1</td>
<td></td>
<td>141.59942</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>DeliveryDiff</td>
<td>2.01900</td>
<td>1</td>
<td></td>
<td>239.35318</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>PricingDiff</td>
<td>1.00300</td>
<td>1</td>
<td></td>
<td>85.52567</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>AvailabilityDiff</td>
<td>-1.76100</td>
<td>1</td>
<td></td>
<td>278.09360</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>QualityDiff</td>
<td>-0.85200</td>
<td>1</td>
<td></td>
<td>53.06240</td>
<td>&lt;.0001</td>
</tr>
</tbody>
</table>

The first one is SpendDiff, which tests simultaneously the following hypotheses:

\[ H_0 : G2\textunderscore InterSpend02 = G1\textunderscore InterSpend02 \]

\[ H_0 : G2\textunderscore InterSpend03 = G1\textunderscore InterSpend03 \]

The exceedingly large chi-square value 10,460 suggests the composite null hypothesis is false. Individual tests for these hypotheses suggest that each of these hypotheses should be rejected. The chi-square values for individual tests are 10,227 and 185.84, respectively.

Similarly, the simultaneous and individual tests of the intercepts in the measurement model suggest that the two models (groups) differ significantly in the means of the measured variables. Region 2 has significantly higher means in variables Helpful, Delivery, and Pricing, but significantly lower means in variables Courtesy, Responsive, Availability, and Quality.

Now you are ready to answer the main research questions. The overall customer service (Service) does affect the future purchase (Spend03), but not the current purchase (Spend02), because the corresponding path coefficient (\textunderscore Parm01) is only marginally significant. Perhaps this is an artifact because the rating was done after the purchases in 2002. That is, purchases in 2002 had been done before the impression about customer service was fully formed. However, this argument cannot explain why overall customer service (Service) also shows a strong and significant relationship with purchases in 2002 (Spend02). Nonetheless, customer service and product quality do affect the future purchases (Spend03) in an expected way, even after partialling out the effect of the previous purchase amount (Spend02). Apart from the mean differences of the variables, the common measurement and prediction (or structural) models fit the two regions very well.

Because the current model fits well and most parts of fitting meet your expectations, you might accept this model without looking for further improvement. Nonetheless, for illustration purposes, it would be useful to consider the LM test results. In Output 29.28.11, ranked LM statistics for the path coefficients in Model 1 and Model 2 are shown.
Output 29.28.11  LM Tests for Path Coefficients

<table>
<thead>
<tr>
<th>To</th>
<th>From</th>
<th>LM Stat</th>
<th>Pr &gt; ChiSq</th>
<th>Parm Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>Courtesy</td>
<td>11.15249</td>
<td>0.0008</td>
<td>-0.17145</td>
</tr>
<tr>
<td>Service</td>
<td>Helpful</td>
<td>3.09038</td>
<td>0.0788</td>
<td>0.09431</td>
</tr>
<tr>
<td>Service</td>
<td>Delivery</td>
<td>2.59511</td>
<td>0.1072</td>
<td>0.07504</td>
</tr>
<tr>
<td>Courtesy</td>
<td>Responsive</td>
<td>1.75943</td>
<td>0.1847</td>
<td>-0.07730</td>
</tr>
<tr>
<td>Delivery</td>
<td>Courtesy</td>
<td>1.66721</td>
<td>0.1966</td>
<td>0.08669</td>
</tr>
<tr>
<td>Helpful</td>
<td>Courtesy</td>
<td>1.62005</td>
<td>0.2031</td>
<td>0.07277</td>
</tr>
<tr>
<td>Courtesy</td>
<td>Product</td>
<td>1.48928</td>
<td>0.2223</td>
<td>-0.14815</td>
</tr>
<tr>
<td>Service</td>
<td>Product</td>
<td>0.83498</td>
<td>0.3608</td>
<td>-0.12327</td>
</tr>
<tr>
<td>Responsive</td>
<td>Helpful</td>
<td>0.76664</td>
<td>0.3813</td>
<td>-0.05625</td>
</tr>
<tr>
<td>Product</td>
<td>Helpful</td>
<td>0.53020</td>
<td>0.4665</td>
<td>-0.03831</td>
</tr>
</tbody>
</table>

Path coefficients that lead to better improvement (larger chi-square decrease) are shown first in the tables. For example, the first path coefficient that is suggested to be freed in Model 1 is the Service <=== Courtesy path. The associated $p$-value is 0.0008 and the estimated change of parameter value is $-0.171$. The second path coefficient is for the Service <=== Helpful path, but it is not significant at the 0.05 level. So, is it good to add the Service <=== Courtesy path to Model 1, based on the LM test results? The answer is that it depends on your application and the theoretical and practical implications. For example, the Service ====> Courtesy path, which is a part of the measurement model, is already specified in Model 1. Even though the LM test statistic shows a significant decrease of model fit chi-square, adding the Service <=== Courtesy path might destroy the measurement model and lead to problematic interpretations. In this case, it is wise not to add the Service <=== Courtesy path, which is suggested by the LM test results.

LM tests for the path coefficients in Model 2 are shown at the bottom of Output 29.28.11. Quite a few of these tests suggest significant improvements in model fit. Again, you are cautioned against adding these paths blindly.

LM tests for the error variances and covariances are shown in Output 29.28.12.
Using $\alpha = 0.05$, you might consider adding two pairs of correlated errors in Model 2. The first pair is for Delivery and Courtesy, which has a $p$-value less than 0.0001. The second pair is Pricing and Responsive, which has a $p$-value of 0.027. Again, adding correlated errors (in the `PCOV` statement) should not be a pure statistical consideration. You should also consider theoretical and practical implications.

LM tests for other subsets of parameters are also conducted. Some subsets do not have parameters that can be freed, and so they are not shown here. Other subsets are not shown here simply for conserving space.

PROC CALIS ranks and outputs the LM test results for some default subsets of parameters. You have seen the subsets for path coefficients and correlated errors in the two previous outputs. Some other LM test results are not shown. With this kind of default LM output, there could be a huge amount of modification indices to look at. Fortunately, you can limit the LM test results to any subsets of potential parameters that you might be interested in. With your substantive knowledge, you can define such meaningful subsets of potential parameters by using the `LMTESTS` statement. The LM test indices and rankings are then done for each predefined subset of potential parameters. With these customized LM results, you can limit your attention to consider only those meaningful parameters to be added. See the `LMTESTS` statement on page 1285 for details.

The next group of LM tests is for releasing implicit equality constraints in your model, as shown in Output 29.28.13.
## Output 29.28.13  LM Tests for Equality Constraints

<table>
<thead>
<tr>
<th>Parm</th>
<th>Model</th>
<th>Var1</th>
<th>Var2</th>
<th>LM Stat</th>
<th>Pr &gt; ChiSq</th>
<th>Original Parm</th>
<th>Released Parm</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Parm01</td>
<td>DV_IV</td>
<td>Spend02</td>
<td>Service</td>
<td>0.01554</td>
<td>0.9008</td>
<td>-0.0213</td>
<td>0.0238</td>
</tr>
<tr>
<td></td>
<td>DV_IV</td>
<td>Spend02</td>
<td>Service</td>
<td>0.01554</td>
<td>0.9008</td>
<td>0.0238</td>
<td>-0.0213</td>
</tr>
<tr>
<td>_Parm02</td>
<td>DV_IV</td>
<td>Spend03</td>
<td>Service</td>
<td>0.01763</td>
<td>0.8944</td>
<td>-0.0222</td>
<td>0.0248</td>
</tr>
<tr>
<td></td>
<td>DV_IV</td>
<td>Spend03</td>
<td>Service</td>
<td>0.01763</td>
<td>0.8944</td>
<td>0.0248</td>
<td>-0.0222</td>
</tr>
<tr>
<td>_Parm03</td>
<td>DV_IV</td>
<td>Spend02</td>
<td>Product</td>
<td>0.0003403</td>
<td>0.9853</td>
<td>-0.00321</td>
<td>0.00355</td>
</tr>
<tr>
<td></td>
<td>DV_IV</td>
<td>Spend02</td>
<td>Product</td>
<td>0.0003403</td>
<td>0.9853</td>
<td>0.00355</td>
<td>-0.00321</td>
</tr>
<tr>
<td>_Parm04</td>
<td>DV_IV</td>
<td>Spend03</td>
<td>Product</td>
<td>0.00176</td>
<td>0.9665</td>
<td>0.00714</td>
<td>-0.00802</td>
</tr>
<tr>
<td></td>
<td>DV_IV</td>
<td>Spend03</td>
<td>Product</td>
<td>0.00176</td>
<td>0.9665</td>
<td>-0.00802</td>
<td>0.00714</td>
</tr>
<tr>
<td>_Parm05</td>
<td>DV_IV</td>
<td>Spend03</td>
<td>Spend02</td>
<td>0.009698</td>
<td>0.9752</td>
<td>-0.00100</td>
<td>0.00112</td>
</tr>
<tr>
<td></td>
<td>DV_IV</td>
<td>Spend03</td>
<td>Spend02</td>
<td>0.009698</td>
<td>0.9752</td>
<td>0.00112</td>
<td>-0.00100</td>
</tr>
<tr>
<td>_Parm06</td>
<td>DV_IV</td>
<td>Courtesy</td>
<td>Service</td>
<td>19.17225</td>
<td>&lt;.0001</td>
<td>0.2851</td>
<td>-0.3191</td>
</tr>
<tr>
<td></td>
<td>DV_IV</td>
<td>Courtesy</td>
<td>Service</td>
<td>19.17225</td>
<td>&lt;.0001</td>
<td>-0.3191</td>
<td>0.2851</td>
</tr>
<tr>
<td>_Parm07</td>
<td>DV_IV</td>
<td>Responsive</td>
<td>Service</td>
<td>0.21266</td>
<td>0.6447</td>
<td>-0.0304</td>
<td>0.0341</td>
</tr>
<tr>
<td></td>
<td>DV_IV</td>
<td>Responsive</td>
<td>Service</td>
<td>0.21266</td>
<td>0.6447</td>
<td>0.0341</td>
<td>-0.0304</td>
</tr>
<tr>
<td>_Parm08</td>
<td>DV_IV</td>
<td>Helpful</td>
<td>Service</td>
<td>4.60629</td>
<td>0.0319</td>
<td>-0.1389</td>
<td>0.1555</td>
</tr>
<tr>
<td></td>
<td>DV_IV</td>
<td>Helpful</td>
<td>Service</td>
<td>4.60629</td>
<td>0.0319</td>
<td>0.1555</td>
<td>-0.1389</td>
</tr>
<tr>
<td>_Parm09</td>
<td>DV_IV</td>
<td>Delivery</td>
<td>Service</td>
<td>3.59763</td>
<td>0.0579</td>
<td>-0.1508</td>
<td>0.1687</td>
</tr>
<tr>
<td></td>
<td>DV_IV</td>
<td>Delivery</td>
<td>Service</td>
<td>3.59763</td>
<td>0.0579</td>
<td>0.1687</td>
<td>-0.1508</td>
</tr>
<tr>
<td>_Parm10</td>
<td>DV_IV</td>
<td>Pricing</td>
<td>Product</td>
<td>0.50974</td>
<td>0.4753</td>
<td>0.0524</td>
<td>-0.0524</td>
</tr>
<tr>
<td></td>
<td>DV_IV</td>
<td>Pricing</td>
<td>Product</td>
<td>0.50974</td>
<td>0.4753</td>
<td>-0.0524</td>
<td>0.0468</td>
</tr>
<tr>
<td>_Parm11</td>
<td>DV_IV</td>
<td>Availability</td>
<td>Product</td>
<td>0.57701</td>
<td>0.4475</td>
<td>-0.0457</td>
<td>0.0512</td>
</tr>
<tr>
<td></td>
<td>DV_IV</td>
<td>Availability</td>
<td>Product</td>
<td>0.57701</td>
<td>0.4475</td>
<td>0.0512</td>
<td>-0.0457</td>
</tr>
<tr>
<td>_Parm12</td>
<td>DV_IV</td>
<td>Quality</td>
<td>Product</td>
<td>0.00566</td>
<td>0.9400</td>
<td>-0.00511</td>
<td>0.00574</td>
</tr>
<tr>
<td></td>
<td>DV_IV</td>
<td>Quality</td>
<td>Product</td>
<td>0.00566</td>
<td>0.9400</td>
<td>0.005574</td>
<td>-0.00511</td>
</tr>
<tr>
<td>_Parm13</td>
<td>COVERR</td>
<td>Courtesy</td>
<td>Courtesy</td>
<td>45.24725</td>
<td>&lt;.0001</td>
<td>0.7204</td>
<td>-0.8064</td>
</tr>
<tr>
<td></td>
<td>COVERR</td>
<td>Courtesy</td>
<td>Courtesy</td>
<td>45.24725</td>
<td>&lt;.0001</td>
<td>-0.8064</td>
<td>0.7204</td>
</tr>
<tr>
<td>_Parm14</td>
<td>COVERR</td>
<td>Responsive</td>
<td>Responsive</td>
<td>1.73499</td>
<td>0.1878</td>
<td>-0.1555</td>
<td>0.1740</td>
</tr>
<tr>
<td></td>
<td>COVERR</td>
<td>Responsive</td>
<td>Responsive</td>
<td>1.73499</td>
<td>0.1878</td>
<td>0.1740</td>
<td>-0.1555</td>
</tr>
<tr>
<td>_Parm15</td>
<td>COVERR</td>
<td>Helpful</td>
<td>Helpful</td>
<td>11.13266</td>
<td>0.0008</td>
<td>-0.3448</td>
<td>0.3860</td>
</tr>
<tr>
<td></td>
<td>COVERR</td>
<td>Helpful</td>
<td>Helpful</td>
<td>11.13266</td>
<td>0.0008</td>
<td>0.3860</td>
<td>-0.3448</td>
</tr>
<tr>
<td>_Parm16</td>
<td>COVERR</td>
<td>Delivery</td>
<td>Delivery</td>
<td>4.99097</td>
<td>0.0255</td>
<td>-0.3364</td>
<td>0.3766</td>
</tr>
<tr>
<td></td>
<td>COVERR</td>
<td>Delivery</td>
<td>Delivery</td>
<td>4.99097</td>
<td>0.0255</td>
<td>0.3766</td>
<td>-0.3364</td>
</tr>
<tr>
<td>_Parm17</td>
<td>COVERR</td>
<td>Pricing</td>
<td>Pricing</td>
<td>2.86428</td>
<td>0.0906</td>
<td>0.1729</td>
<td>-0.1936</td>
</tr>
<tr>
<td></td>
<td>COVERR</td>
<td>Pricing</td>
<td>Pricing</td>
<td>2.86428</td>
<td>0.0906</td>
<td>-0.1936</td>
<td>0.1729</td>
</tr>
<tr>
<td>_Parm18</td>
<td>COVERR</td>
<td>Availability</td>
<td>Availability</td>
<td>2.53147</td>
<td>0.1116</td>
<td>-0.1494</td>
<td>0.1672</td>
</tr>
<tr>
<td></td>
<td>COVERR</td>
<td>Availability</td>
<td>Availability</td>
<td>2.53147</td>
<td>0.1116</td>
<td>-0.1672</td>
<td>0.1494</td>
</tr>
<tr>
<td>_Parm19</td>
<td>COVERR</td>
<td>Quality</td>
<td>Quality</td>
<td>0.07328</td>
<td>0.7866</td>
<td>-0.0315</td>
<td>0.0352</td>
</tr>
<tr>
<td></td>
<td>COVERR</td>
<td>Quality</td>
<td>Quality</td>
<td>0.07328</td>
<td>0.7866</td>
<td>0.0352</td>
<td>-0.0315</td>
</tr>
<tr>
<td>_Parm20</td>
<td>COVERR</td>
<td>Spend02</td>
<td>Spend02</td>
<td>0.00214</td>
<td>0.9631</td>
<td>-0.0304</td>
<td>0.0340</td>
</tr>
<tr>
<td></td>
<td>COVERR</td>
<td>Spend02</td>
<td>Spend02</td>
<td>0.00214</td>
<td>0.9631</td>
<td>0.0340</td>
<td>-0.0304</td>
</tr>
<tr>
<td>_Parm21</td>
<td>COVERR</td>
<td>Spend03</td>
<td>Spend03</td>
<td>0.0001773</td>
<td>0.9894</td>
<td>-0.00842</td>
<td>0.00946</td>
</tr>
<tr>
<td></td>
<td>COVERR</td>
<td>Spend03</td>
<td>Spend03</td>
<td>0.0001773</td>
<td>0.9894</td>
<td>0.00946</td>
<td>-0.00842</td>
</tr>
<tr>
<td>_Parm22</td>
<td>COVEXOG</td>
<td>Service</td>
<td>Product</td>
<td>0.87147</td>
<td>0.3505</td>
<td>0.0605</td>
<td>-0.0678</td>
</tr>
<tr>
<td></td>
<td>COVEXOG</td>
<td>Service</td>
<td>Product</td>
<td>0.87147</td>
<td>0.3505</td>
<td>-0.0678</td>
<td>0.0605</td>
</tr>
</tbody>
</table>
Recall that the measurement and the prediction models for the two regions are constrained to be the same by model referencing (that is, the REFMODEL statement). Output 29.28.13 shows you which parameter can be unconstrained so that your overall model fit might improve. For example, if you unconstrain the first parameter _Parm01, which is for the path effect of Spend02 ===> Service, for the two models, the expected chi-square decrease (LM Stat) is about 0.0158, which is not significant ($p = .9001$). The associated parameter changes are small too. However, if you consider unconstraining parameter _Parm06, which is for the path effect of Courtesy ===> Service, the expected decrease of chi-square is 19.22 ($p < .0001$). There are two rows for this parameter. Each row represents a parameter location to be released from the equality constraint. Consider the first row first. If you rename the coefficient for the Courtesy ===> Service path in Model 1 to a new parameter, say “new” (while keeping _Parm06 as the parameter for the Courtesy ===> Service path in Model 2) and fit the model again, the new estimate of _Parm06 is 0.2852 greater than the previous _Parm06 estimate. The estimate of “new” is 0.3196 less than the previous _Parm06 estimate. The second row for the _Parm06 parameter shows similar but reflected results. It is for renaming the parameter location in Model 2. For this example each equality constraint has exactly two locations, one for Model 1 and one for Model 2. That is the reason why you always observe reflected results for freeing the locations successively. Reflected results are not the case if you have equality constraints with more than two parameter locations.

Another example of a large expected improvement of model fit is the result of freeing the constrained variances of Courtesy among the two models. The corresponding row to look at is the row with parameter _Parm13, where the parameter type is labeled “COVERR” and the values for Var1 and Var2 are both “Courtesy.” The LM statistic is 45.255, which is a significant chi-square decrease if you free either parameter location. If you choose to rename the error variance for Courtesy in Model 1, the new _Parm13 estimate is 0.8052 smaller than the original _Parm13 estimate. The new estimate of the error variance for Courtesy in Model 2 is 0.7211 greater than the previous _Parm13 estimate. Finally, the constrained parameter _Parm15, which is the error variance parameter for Helpful in both models, is also a potential constraint that can be released with a significant model fit improvement.

In addition to the LM statistics for suggesting ways to improve model fit, PROC CALIS also computes the Wald tests to show which parameters can be constrained to zero without jeopardizing the model fit significantly. The Wald test results are shown in Output 29.28.14.

<table>
<thead>
<tr>
<th>Parm</th>
<th>Chi-Square</th>
<th>DF</th>
<th>Pr &gt; ChiSq</th>
<th>Chi-Square</th>
<th>Pr &gt; ChiSq</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Parm01</td>
<td>3.09039</td>
<td>1</td>
<td>0.0788</td>
<td>3.09039</td>
<td>0.0788</td>
</tr>
</tbody>
</table>

In Output 29.28.14, you see that _Parm01, which is for the path effect of Spend02 ===> Service, is suggested to be a fixed zero parameter (eliminated from the model) by the Wald test. Fixing this parameter to zero (or dropping the Spend02 ===> Service path from the model) is expected to increase the model fit chi-square by 3.085 ($p = .079$), which is only marginally significant at $\alpha = 0.05$.

As is the case for the LM test statistics, you should not automatically adhere to the suggestions by the Wald statistics. Substantive and theoretical considerations should always be considered when determining whether a parameter should be added or dropped.
Chapter 29: The CALIS Procedure

Example 29.29: Fitting the RAM and EQS Models by the COSAN Modeling Language

The COSAN modeling language in PROC CALIS enables you to specify the direct or implied mean and covariance structures for the data in terms of matrix formulas. It is a very general modeling language, and all other modeling languages in PROC CALIS are special cases of the COSAN modeling language. This example shows how you can apply the COSAN modeling language to situations where you might usually use the “easier” modeling languages. Therefore, the purpose of this example is not to recommend the use of the COSAN modeling specification to the specific application. Rather, through its connections with other more well-known model types, this example intends to help you understand the basics of the COSAN modeling language.

In Example 29.17, you fit a path model to the Wheaton data (Wheaton et al. 1977) by using the PATH modeling language. The mathematical basis of the PATH modeling language is the RAM model. In Example 29.23, you use the RAM and LINEQS statements to specify the same path model. In all these different types of specifications, you specify the functional relationships of the variables and the variance and covariance parameters in the model. PROC CALIS then generates the implied covariance structures for analysis internally. The COSAN modeling language is quite different. In the COSAN statement, you specify the covariance structures directly as a matrix formula. This example shows how you can do that in two different ways. One specification emulates the RAM model (McDonald 1978, 1980) covariance structures and the other emulates the EQS model (Bentler 1995) covariance structures.

Emulating the RAM model by the COSAN Modeling Language

In the RAM model, you specify all information regarding the path effects or coefficients (that is, single-headed arrows in the path diagram) in the so-called A (\(_A\_\)) matrix. You specify all the information regarding the variances and covariances (that is, the double-headed arrows in the path diagram) in the P (\(_P\_\)) matrix. See the section “The RAM Model” on page 1425 for more details about the mathematical model for RAM. Once you define these two matrices, the implied covariance structures for the observed variables are derived by the formula

\[
\Sigma = J \ast (I - A)^{-1} \ast P \ast (I - A)^{-1'} \ast J'
\]

where \(I\) is an identity matrix and \(J\) is a selection matrix that contains 0 or 1 as its elements for selecting the covariance structures elements for the observed variables.

For example, in the RAM model specification in Example 29.23, you essentially use the following RAM model specification:
In the RAM statement, you specify all the parameters in the _A_ and _P_ matrices, and PROC CALIS generates the corresponding covariance structures for analysis. However, with the COSAN modeling language, in addition to the parameter in the model matrices, you need to supply the matrix formula for the covariance structures, as shown in the preceding formula for \( \Sigma \).

Before discussing how you can specify the COSAN model that corresponds to this RAM model specification, it is useful to look at the initial model matrices that are generated by the preceding RAM model specification. To do this, you use the PRIMAT option in the PROC CALIS statement.
Output 29.29.1 and Output 29.29.2 show the initial _A_ and _P_ matrices, respectively, for the RAM model.

**Output 29.29.1 Initial _A_ Matrix of the RAM Model**

<table>
<thead>
<tr>
<th></th>
<th>Anomie67</th>
<th>Powerless67</th>
<th>Anomie71</th>
<th>Powerless71</th>
<th>Education</th>
<th>SEI</th>
<th>Alien67</th>
<th>Alien71</th>
<th>SES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8330</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Anomie71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0000</td>
<td>0</td>
</tr>
<tr>
<td>Powerless71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8330</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Education</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0000</td>
<td>0</td>
</tr>
<tr>
<td>SEI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alien67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alien71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SES</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Output 29.29.2 Initial _P_ Matrix of the RAM Model**

<table>
<thead>
<tr>
<th></th>
<th>Anomie67</th>
<th>Powerless67</th>
<th>Anomie71</th>
<th>Powerless71</th>
<th>Education</th>
<th>SEI</th>
<th>Alien67</th>
<th>Alien71</th>
<th>SES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67</td>
<td>[theta1]</td>
<td>0</td>
<td>[theta5]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless67</td>
<td>0</td>
<td>0</td>
<td>[theta2]</td>
<td>[theta5]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Anomie71</td>
<td>[theta5]</td>
<td>0</td>
<td>[theta1]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless71</td>
<td>0</td>
<td>[theta5]</td>
<td>0</td>
<td>[theta2]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Education</td>
<td>[theta5]</td>
<td>0</td>
<td>[theta1]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SEI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>[theta3]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alien67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alien71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SES</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Essentially, to specify the same model by the COSAN modeling language, you need to provide the same information in these two initial model matrices and the covariance structure formula for $\Sigma$ in the COSAN model specification, which is shown in the following statements:
proc calis data=Wheaton nobs=932 nose;
  cosan
    var= Anomie67 Powerless67 Anomie71 Powerless71 Education SEI,
         J(9, IDE) * A(9, GEN, IMI) * P(9, SYM);
  matrix A
      [1 2 8 , 7] = 1.0 0.833 beta,
      [3 4 , 8] = 1.0 0.833 ,
      [5 6 7 8 , 9] = 1. lambda gamma1 gamma2;
  matrix P
      [1,1] = theta1-theta2 theta1-theta4 ,
      [7,7] = psi1 psi2 phi,
      [3,1] = theta5 ,
      [4,2] = theta5 ;
  vnames
    J = [Anomie67 Powerless67 Anomie71 Powerless71
         Education SEI Alien67 Alien71 SES],
    A = J,
    P = A;
run;

In the PROC CALIS statement, you provide the data set in the DATA= option and the number of observations in the NOBS= option. You use the NOSE option to turn off the computation of the standard error estimates.

In the VAR= option of the COSAN statement, you provide the list of observed variables for the analysis. You do not specify the latent variables in the VAR= option in the COSAN statement as you do in the VAR= option in the RAM statement. Then, you specify the formula for the covariance structures for the set of variables in the VAR= list. Because the covariance structure formula is symmetric, you only need to specify “half” of it. That is, the specification \( J(9, \text{IDE}) \ast A(9, \text{GEN, IMI}) \ast P(9, \text{SYM}) \) in the COSAN statement automatically expands to

\[
J \ast (I - A)^{-1} \ast P \ast (I - A)^{-1'} \ast J'
\]

which is the required covariance structures. The arguments in the matrices represent the number of columns, the matrix type, and the transformation type (optional), respectively. For example, the notation \( A(9, \text{GEN, IMI}) \) means that matrix \( A \) has nine columns and it is a general (GEN) rectangular or square matrix. You do not specify the number of rows for matrix \( A \) explicitly, but PROC CALIS can deduce that because matrix \( A \) follows matrix \( J \) in the multiplication. To make matrix multiplication conformable, the number of rows for matrix \( A \) must be the same as the number of columns for matrix \( J \), which is nine. The IMI notation means the identity-minus-inverse transformation, which results in putting \( (I - A)^{-1} \) in the expression. Matrix \( P \) in the covariance structure formula is a \( 9 \times 9 \) symmetric matrix. It does not have any transformation in the formula. Matrix \( J \) in the covariance structure formula is a so-called generalized identity matrix (IDE), which has six rows and nine columns. Basically, you use this matrix to select the observed variables in the covariance structure formula. The exact form of this matrix will become clear when the PROC CALIS output is shown.

Next, you use two MATRIX statements to specify the parameters in the model matrices \( A \) and \( P \), for RAM model matrices _A_ and _P_, respectively. For example, in the first entry of the MATRIX statement for the \( A \) matrix, you specify the elements \( [1, 7] \), \( [2, 7] \), and \( [8, 7] \) by 1.0, 0.833, and \( \beta \) respectively. The first two elements are fixed constants, while the last one is a free parameter named \( \beta \). Similarly, you specify all the fixed or free parameters in matrix \( A \), which reflects the same pattern you specify for the _A_ matrix of the RAM model, as shown in Output 29.29.1.
For the $P$ matrix, you specify the parameters in the same fashion. Because $P$ is defined as a symmetric matrix, you need to specify only the lower triangular elements. In the first entry of the MATRIX statement for the $P$ matrix, you specify the $[1, 1]$ element, but the trailing parameter list has six parameters. The $[1, 1]$ notation here is interpreted as the starting location of the matrix. It proceeds to $[2, 2]$, $[3, 3]$, $[4, 4]$ and so on. The length of the trailing parameter list determines the number of elements being specified. Therefore, the last parameter in this entry is for $P[6, 6]$, which is a free parameter $\theta_4$. Similarly, you define all other parameters in the $P$ matrix, which reflects the same pattern you specify for the $P_1$ matrix of the RAM model, as shown in Output 29.29.2.

In the VNAME statement, you can specify the column variable names for the model matrices. You provide a set of nine variable names for the column of matrix $J$ in the pairs of brackets. The first six names are those of the observed variables in the COSAN model, while the last six names are for latent factors. How about the row variable names for matrix $J$? Because matrix $J$ is the first matrix in the covariance structure formula, its row names are automatically the same as the names of the observed variables in the VAR= list of the COSAN statement. Next, you specify the column variable names of matrix $A$. You equate that to matrix $J$, meaning that the column variable names in matrix $A$ are the same those for matrix $J$. How about the row variable names for matrix $A$? Because matrix $A$ follows matrix $J$ in the covariance structure formula, its row names are automatically same as the column names for matrix $J$. Lastly, you define that the column names for matrix $P$ are the same as those for matrix $A$.

Notice that column names serve only as labels. PROC CALIS does not know the identities of the row and column variables. For example, the first column of matrix $A$ is Anomie67, which is also a name for an observed variable in the COSAN model. Keeping other specifications intact, you could name this column by any other name without affecting the model estimation. It is recommended that you use sensible names that help you remember the identities of the row and column variables, such as this example shows.

Output 29.29.3 shows the modeling information and the observed variables in the COSAN model. PROC CALIS analyzed the covariance structures of the six observed variables listed in Output 29.29.3.

**Output 29.29.3** Modeling Information of the COSAN Model for the Wheaton Data: RAM Emulation

<table>
<thead>
<tr>
<th>Modeling Information</th>
<th>Maximum Likelihood Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set</td>
<td>WORK.WHEATON</td>
</tr>
<tr>
<td>N Obs</td>
<td>932</td>
</tr>
<tr>
<td>Model Type</td>
<td>COSAN</td>
</tr>
<tr>
<td>Analysis</td>
<td>Covariances</td>
</tr>
</tbody>
</table>

**Observed Variables (N = 6) in the Model**

Anomie67 Powerless67 Anomie71 Powerless71 Education SEI
Output 29.29.4 shows the covariance structures and some properties of the model matrices. The covariance structure formula for $\Sigma$ is defined as required. You can also check the matrix properties in this output to see if they are what you intend them to be.

**Output 29.29.4** The Covariance Structures and Model Matrices of the COSAN Model for the Wheaton Data: RAM Emulation

<table>
<thead>
<tr>
<th>COSAN Model Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma = J^*\text{inv}(I-A)^<em>P^</em>(\text{inv}(I-A))^*J$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Summary of Model Matrices</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Matrix</strong></td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>J</td>
</tr>
<tr>
<td>P</td>
</tr>
</tbody>
</table>

Output 29.29.4 shows that $J$ is a $6 \times 9$ “identity” matrix $(I||0)$. Essentially, $J$ is a selection matrix that contains either 0 or 1 as its elements. The role of matrix $J$ in the covariance structure formula is to extract first six rows and columns in the inner covariance structures $(I - A)^{-1} \ast P \ast (I - A)^{-1/2}$ (which is $9 \times 9$) to form the covariance structures only for the observed variables (which is $6 \times 6$). But how can this identity matrix have more columns (9) than rows (6)? In common mathematical notation, an identity matrix must always be a square matrix. However, for convenience in notation, PROC CALIS generalizes it to the IDE type. An IDE matrix that has the same numbers of columns and rows is a square identity matrix. If an IDE matrix has more columns than rows, it denotes an identity matrix concatenated (to the right) by a null matrix (that is, the $(I||0)$ notation). If an IDE matrix has more rows than columns, it denotes an identity matrix appended (to the bottom) by a null matrix (that is, the $(I//0)$ notation). The generalized definition for the IDE matrix offers an efficient way to define selection matrix, such as the $J$ matrix shown in this example.

Output 29.29.5 shows the model fit chi-square of the COSAN model. This is the same model fit as in Output 29.17.6 of Example 29.17, as expected.

**Output 29.29.5** Model Fit of the COSAN Model for the Wheaton Data: RAM Emulation

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
</tbody>
</table>
Output 29.29.6 shows the estimates in the $A$ matrix.

**Output 29.29.6** Estimate of the $A$ Matrix by the COSAN Model Specification

<table>
<thead>
<tr>
<th>Model Matrix $A$ (9 x 9 General Square Matrix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67 Powerless67 Anomie71 Powerless71 Education SEI Alien67 Alien71 SES</td>
</tr>
<tr>
<td>Anomie67</td>
</tr>
<tr>
<td>Powerless67</td>
</tr>
<tr>
<td>Anomie71</td>
</tr>
<tr>
<td>Powerless71</td>
</tr>
<tr>
<td>Education</td>
</tr>
<tr>
<td>SEI</td>
</tr>
<tr>
<td>Alien67</td>
</tr>
<tr>
<td>Alien71</td>
</tr>
<tr>
<td>SES</td>
</tr>
</tbody>
</table>

The estimates in Output 29.29.6 from the COSAN model specification are essentially the same as those from the RAM model specification, as shown in the matrix form in **Output 29.29.7**.

**Output 29.29.7** Estimate of the $A$ Matrix by the RAM Model Specification

<table>
<thead>
<tr>
<th>RAM <em>A</em> Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67 Powerless67 Anomie71 Powerless71 Education SEI Alien67 Alien71 SES</td>
</tr>
<tr>
<td>Anomie67</td>
</tr>
<tr>
<td>Powerless67</td>
</tr>
<tr>
<td>Anomie71</td>
</tr>
<tr>
<td>Powerless71</td>
</tr>
<tr>
<td>Education</td>
</tr>
<tr>
<td>SEI</td>
</tr>
<tr>
<td>Alien67</td>
</tr>
<tr>
<td>Alien71</td>
</tr>
<tr>
<td>SES</td>
</tr>
</tbody>
</table>
Output 29.29.8 shows the estimates in the $P$ matrix.

**Output 29.29.8** Estimate of the $P$ Matrix by the COSAN Model Specification

<table>
<thead>
<tr>
<th>Model Matrix $P$</th>
<th>Anomie67</th>
<th>Powerless67</th>
<th>Anomie71</th>
<th>Powerless71</th>
<th>Education</th>
<th>SEI</th>
<th>Alien67</th>
<th>Alien71</th>
<th>SES</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(9 \times 9$ Symmetric Matrix)</td>
<td>3.6078 [theta1]</td>
<td>0</td>
<td>0.9058 [theta5]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless67</td>
<td>0</td>
<td>3.5950 [theta2]</td>
<td>0</td>
<td>0.9058 [theta5]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Anomie71</td>
<td>0.9058 [theta5]</td>
<td>0</td>
<td>3.6078 [theta1]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless71</td>
<td>0</td>
<td>0.9058 [theta5]</td>
<td>0</td>
<td>3.5950 [theta2]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Education</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.9938 [theta3]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SEI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>259.5738 [theta4]</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alien67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5.6705 [psi1]</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alien71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.5148 [psi2]</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SES</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6.6162 [phi]</td>
</tr>
</tbody>
</table>

Again, aside from very minor numerical differences, the estimates shown in Output 29.29.8 from the COSAN model specification are essentially the same as those from the RAM model specification, as shown in the matrix form in Output 29.29.9.

**Output 29.29.9** Estimate of the $P$ Matrix by the RAM Model Specification

<table>
<thead>
<tr>
<th>RAM $P$ Matrix</th>
<th>Anomie67</th>
<th>Powerless67</th>
<th>Anomie71</th>
<th>Powerless71</th>
<th>Education</th>
<th>SEI</th>
<th>Alien67</th>
<th>Alien71</th>
<th>SES</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(9 \times 9$ Symmetric Matrix)</td>
<td>3.6080 [theta1]</td>
<td>0</td>
<td>0.9058 [theta5]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless67</td>
<td>0</td>
<td>3.5949 [theta2]</td>
<td>0</td>
<td>0.9058 [theta5]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Anomie71</td>
<td>0.9058 [theta5]</td>
<td>0</td>
<td>3.6080 [theta1]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless71</td>
<td>0</td>
<td>0.9058 [theta5]</td>
<td>0</td>
<td>3.5949 [theta2]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Education</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.9937 [theta3]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SEI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>259.5764 [theta4]</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alien67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5.6705 [psi1]</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Alien71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.5148 [psi2]</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SES</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6.6162 [phi]</td>
</tr>
</tbody>
</table>
Emulating the EQS model by the COSAN Modeling Language

The LINEQS modeling language in PROC CALIS enables you to specify the functional relationships among variables by using the equation input, much the same way that you can do with the EQS software (Bentler 1995). The covariance structure formula for the observed variables in the EQS model is

$$ \Sigma = J \cdot (I - \text{Beta})^{-1} \cdot \text{Gamma} \cdot \text{Phi} \cdot (I - \text{Beta})^{-1'} \cdot J' $$

where $I$ is an identity matrix, $J$ is a selection matrix that contains 0 or 1 as its elements for selecting the covariance structures elements for the observed variables, $\text{Beta}$ is a square matrix for specifying relationships among the endogenous variables, $\text{Gamma}$ is a matrix for specifying relationships between the endogenous variables and the exogenous variables, and $\text{Phi}$ is a matrix for specifying the variances and covariances of the exogenous variables. Notice that in the EQS model, error or disturbance variables are counted as exogenous variables in the model.

In Example 29.23, you use the following LINEQS specification for the Wheaton data:

```plaintext
proc calis nobs=932 data=Wheaton primat nose;
  lineqs
    Anomie67 = 1.0 * f_Alien67 + e1,
    Powerless67 = 0.833 * f_Alien67 + e2,
    Anomie71 = 1.0 * f_Alien71 + e3,
    Powerless71 = 0.833 * f_Alien71 + e4,
    Education = 1.0 * f_SES + e5,
    SEI = lambda * f_SES + e6,
    f_Alien67 = gamma1 * f_SES + d1,
    f_Alien71 = gamma2 * f_SES + beta * f_Alien67 + d2;
  variance
    E1 = theta1,
    E2 = theta2,
    E3 = theta1,
    E4 = theta2,
    E5 = theta3,
    E6 = theta4,
    D1 = psi1,
    D2 = psi2,
    f_SES = phi;
  cov
    E1 E3 = theta5,
    E2 E4 = theta5;
run;
```

In the LINEQS statement, you specify all the functional relationships among variables. In the VARIANCE and COV statements, you specify all the variance and covariance parameters in the model. None of the parameters is specified as a matrix element in the LINEQS model. The default output by PROC CALIS does not print the EQS model matrices. To print these model matrices, you use the PRIMAT option in the PROC CALIS statement. Output 29.29.10, Output 29.29.11, and Output 29.29.12 show the initial specification of these model matrices:
**Output 29.29.10**  The Initial \_EQSBETA\_ Matrix by the LINEQS Model Specification

<table>
<thead>
<tr>
<th>Initial EQSBETA Matrix</th>
<th>Anomie67</th>
<th>Anomie71</th>
<th>Education</th>
<th>Powerless67</th>
<th>Powerless71</th>
<th>SEI</th>
<th>f_Alien67</th>
<th>f_Alien71</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0000</td>
<td>0</td>
<td>1.0000</td>
</tr>
<tr>
<td>Anomie71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0000</td>
<td>0</td>
</tr>
<tr>
<td>Education</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8330</td>
<td>0</td>
</tr>
<tr>
<td>Powerless71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8330</td>
<td>0</td>
</tr>
<tr>
<td>SEI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f_Alien67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f_Alien71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Output 29.29.11**  The Initial \_EQSGAMMA\_ Matrix by the LINEQS Model Specification

<table>
<thead>
<tr>
<th>Initial EQSGAMMA Matrix</th>
<th>f_SES</th>
<th>e1</th>
<th>e3</th>
<th>e5</th>
<th>e2</th>
<th>e4</th>
<th>e6</th>
<th>d1</th>
<th>d2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67</td>
<td>0</td>
<td>1.0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Anomie71</td>
<td>0</td>
<td>0</td>
<td>1.0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Education</td>
<td>1.0000</td>
<td>0</td>
<td>0</td>
<td>1.0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SEI</td>
<td>[lambda]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f_Alien67</td>
<td>[gamma1]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0000</td>
<td>0</td>
</tr>
<tr>
<td>f_Alien71</td>
<td>[gamma2]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
Output 29.29.12 The Initial _EQSPHI_ Matrix by the LINEQS Model Specification

<table>
<thead>
<tr>
<th></th>
<th>Initial_EQSPHI_Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_SES</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>e1</td>
<td>0                [theta1] [theta5] 0 0 0 0 0</td>
</tr>
<tr>
<td>e3</td>
<td>0                [theta5] [theta1] 0 0 0 0 0</td>
</tr>
<tr>
<td>e5</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>e2</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>e4</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>e6</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>d1</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>d2</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

In the COSAN modeling language, you need to provide the three initial model matrices and the covariance structure formula for \( \Sigma \), which is shown in the following statements:

```plaintext
proc calis cov data=Wheaton nobs=932 nose;
   cosan
      var = Anomie67 Anomie71 Education Powerless67 Powerless71 SEI,
         J(8, IDE) * Beta(8, GEN, IMI) * Gamma(9, GEN) * Phi(9, SYM);
   matrix Beta
      [1 4 8 , 7] = 1.0 0.833 beta,
      [2 5 , 8] = 1.0 0.833 ;
   matrix Gamma
      [3 6 7 8 , 1] = 1.0 lambda gamma1 gamma2,
      [1,2] = 8 * 1.0;
   matrix Phi
      [1,1] = phi 2*theta1 theta3 2*theta2 theta4 psi1 psi2,
      [3,2] = theta5 ,
      [6,5] = theta5 ;
   vnames J = [Anomie67 Anomie71 Education Powerless67 Powerless71 SEI
               f_Alien67 f_Alien71],
   Beta = J,
   Gamma = [f_SES e1 e3 e5 e2 e4 e6 d1 d2],
   Phi = Gamma;
run;
```

In the PROC CALIS statement, you provide the data set in the DATA= option and the number of observations in the NOBS= option. You use the NOSE option to turn off the computation of the standard error estimates.

In the VAR= option of the COSAN statement, you provide the list of observed variables for the analysis. You arrange the observed variables in such a way that they are in the same order as in Output 29.29.10, Output 29.29.10, and Output 29.29.12. This is useful for comparing the results from the LINEQS and COSAN model specifications. After the specification of the observed variables, you specify the covariance
structure model in the COSAN statement. Again, you only need to specify “half” of it. That is, the specification \( J(8, IDE) \times Beta(8, GEN, IMI) \times Gamma(9, GEN) \times Phi(9, SYM) \) in the COSAN statement automatically expands to

\[
\Sigma = J \times (I - Beta)^{-1} \times Gamma \times Phi \times Gamma' \times (I - Beta)^{-1'} \times J'
\]

which is the required covariance structures. Matrix properties and transformation types are defined in the arguments for the matrices.

Next, you use three matrix statements to specify the parameters in the matrix elements. The specifications here reflect exactly the initial specifications for the LINEQS model matrices as shown in Output 29.29.10, Output 29.29.10, and Output 29.29.12.

In the VNAMES statement, you specify the column variable names for the matrices. The column variable names of the \( J \) matrix include all the observed variable names and the names of the intended endogenous latent factors \( f_{\text{Alien67}} \) and \( f_{\text{Alien71}} \). The column variable names for the \( Beta \) matrix are the same as those for matrix \( J \). The column variables for the \( Gamma \) matrix include the intended latent factor \( f_{\text{SES}} \) and error variable names \( e1–e6 \) and \( d1–d2 \), which are arranged in such a way that they match the order of the error variables in the LINEQS output shown in Output 29.29.12.

Output 29.29.13 shows the covariance structures and some properties of the model matrices. The covariance structure formula for \( \Sigma \) is defined as required. You can also check the matrix properties in this output to see if they are what you intend them to be.

**Output 29.29.13** The Covariance Structures and Model Matrices of the COSAN Model for the Wheaton Data: EQS Emulation

<table>
<thead>
<tr>
<th>COSAN Model Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Sigma = J \times \text{inv}(I - Beta) \times Gamma \times Phi \times Gamma' \times \text{inv}(I - Beta)' \times J' )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Summary of Model Matrices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>Beta</td>
</tr>
<tr>
<td>Gamma</td>
</tr>
<tr>
<td>J</td>
</tr>
<tr>
<td>Phi</td>
</tr>
</tbody>
</table>

Output 29.29.14 shows the model fit chi-square of the current COSAN model. As expected, this is the same model fit as in Output 29.17.6 of Example 29.17 and in Output 29.29.5.

**Output 29.29.14** Model Fit of the COSAN Model for the Wheaton Data: EQS Emulation

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
</tbody>
</table>

Output 29.29.15 shows the estimates of the \( Beta \) matrix by the COSAN model specification. These estimates are essentially the same as the estimates of the \( _{\text{EQSBETA}} \) matrix obtained from the LINEQS model specification, as shown in Output 29.29.16.
**Output 29.29.15** Estimate of the Beta Matrix by the COSAN Model Specification

<table>
<thead>
<tr>
<th>Model Matrix Beta</th>
</tr>
</thead>
<tbody>
<tr>
<td>(8 x 8 General Square Matrix)</td>
</tr>
<tr>
<td>Anomie67   Anomie71 Education Powerless67 Powerless71 SEI f_Alien67 f_Alien71</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>Anomie67        0   0   0   0   0   0   1.0000   0</td>
</tr>
<tr>
<td>Anomie71        0   0   0   0   0   0   1.0000   0</td>
</tr>
<tr>
<td>Education       0   0   0   0   0   0   0          0</td>
</tr>
<tr>
<td>Powerless67     0   0   0   0   0   0   0.8330   0</td>
</tr>
<tr>
<td>Powerless71     0   0   0   0   0   0   0          0.8330</td>
</tr>
<tr>
<td>SEI             0   0   0   0   0   0   0          0</td>
</tr>
<tr>
<td>f_Alien67       0   0   0   0   0   0   0          0</td>
</tr>
<tr>
<td>f_Alien71       0   0   0   0   0   0   0.5931    [beta] 0</td>
</tr>
</tbody>
</table>

**Output 29.29.16** Estimate of the _EQSBETA_ Matrix by the LINEQS Model Specification

<table>
<thead>
<tr>
<th><em>EQSBETA</em> Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67       Anomie71 Education Powerless67 Powerless71 SEI f_Alien67 f_Alien71</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>Anomie67        0   0   0   0   0   0          1.0000   0</td>
</tr>
<tr>
<td>Anomie71        0   0   0   0   0   0          1.0000   0</td>
</tr>
<tr>
<td>Education       0   0   0   0   0   0          0          0</td>
</tr>
<tr>
<td>Powerless67     0   0   0   0   0   0          0.8330   0</td>
</tr>
<tr>
<td>Powerless71     0   0   0   0   0   0          0          0.8330</td>
</tr>
<tr>
<td>SEI             0   0   0   0   0   0          0          0</td>
</tr>
<tr>
<td>f_Alien67       0   0   0   0   0   0          0          0</td>
</tr>
<tr>
<td>f_Alien71       0   0   0   0   0   0          0.5931    [beta] 0</td>
</tr>
</tbody>
</table>

Output 29.29.17 shows the estimates of the Gamma matrix by the COSAN model specification. Again, these estimates are essentially the same as the estimates of the _EQSGAMMA_ matrix obtained from the LINEQS model specification, as shown in Output 29.29.18.
Example 29.29: Fitting the RAM and EQS Models by the COSAN Modeling Language

Output 29.29.17 Estimate of the Gamma Matrix by the COSAN Model Specification

<table>
<thead>
<tr>
<th>Model Matrix Gamma</th>
<th>f_SES</th>
<th>e1</th>
<th>e3</th>
<th>e5</th>
<th>e2</th>
<th>e4</th>
<th>e6</th>
<th>d1</th>
<th>d2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67</td>
<td>0.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Anomie71</td>
<td>0</td>
<td>0.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Education</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SEI</td>
<td>5.3689</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f_Alien67</td>
<td>-0.6299</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f_Alien71</td>
<td>-0.2409</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
</tr>
</tbody>
</table>

Output 29.29.18 Estimate of the _EQSGAMMA_ Matrix by the LINEQS Model Specification

<table>
<thead>
<tr>
<th><em>EQSGAMMA</em> Matrix</th>
<th>f_SES</th>
<th>e1</th>
<th>e3</th>
<th>e5</th>
<th>e2</th>
<th>e4</th>
<th>e6</th>
<th>d1</th>
<th>d2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomie67</td>
<td>0.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Anomie71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Education</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Powerless71</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SEI</td>
<td>5.3688</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f_Alien67</td>
<td>-0.6299</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f_Alien71</td>
<td>-0.2409</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
<td>0</td>
</tr>
</tbody>
</table>

Finally, Output 29.29.19 shows the estimates of the Phi matrix by the COSAN model specification. These estimates are essentially the same as the estimates of the _EQSPHI_ matrix obtained from the LINEQS model specification, as shown in Output 29.29.20.
### Output 29.29.19 Estimate of the $\Phi$ Matrix by the COSAN Model Specification

<table>
<thead>
<tr>
<th>Model Matrix Phi (9 x 9 Symmetric Matrix)</th>
<th>f_SES</th>
<th>e1</th>
<th>e3</th>
<th>e5</th>
<th>e2</th>
<th>e4</th>
<th>e6</th>
<th>d1</th>
<th>d2</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_SES 6.6162 [phi]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>e1 0 3.6078 [theta1] [theta5]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>e3 0 0.9058 3.6078 [theta5] [theta1]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>e5 0 2.9938 0.000 [theta3]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>e2 0 0.000 3.5950 [theta2] [theta5]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>e4 0 0.9058 3.5950 [theta5] [theta2]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>e6 0 0.000 0.000 259.5738 [theta4]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>d1 0 5.6705 [psi1]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>d2 0 4.5148 [psi2]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

### Output 29.29.20 Estimate of the _EQSPHI_ Matrix by the LINEQS Model Specification

<table>
<thead>
<tr>
<th><em>EQSPHI</em> Matrix</th>
<th>f_SES</th>
<th>e1</th>
<th>e3</th>
<th>e5</th>
<th>e2</th>
<th>e4</th>
<th>e6</th>
<th>d1</th>
<th>d2</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_SES 6.6163 [phi]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>e1 0 3.6080 [theta1] [theta5]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>e3 0 0.9058 3.6080 [theta5] [theta1]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>e5 0 2.9937 0.000 [theta3]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>e2 0 0.000 3.5949 [theta2] [theta5]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>e4 0 0.9058 3.5949 [theta5] [theta2]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>e6 0 0.000 0.000 259.5764 [theta4]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>d1 0 5.6705 [psi1]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>d2 0 4.5148 [psi2]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>
Example 29.30: Second-Order Confirmatory Factor Analysis

A second-order confirmatory factor analysis model is applied to a correlation matrix of Thurstone reported by McDonald (1985). The data set is shown in the following DATA step:

data Thurst (TYPE=CORR);
  title "Example of THURSTONE resp. McDONALD (1985, p.57, p.105)";
  _TYPE_ = 'CORR'; Input _NAME_ $ Obs1-Obs9;
  label obs1='Sentences' obs2='Vocabulary' obs3='Sentence Completion'
    obs4='First Letters' obs5='Four-letter Words' obs6='Suffixes'
    obs7='Letter series' obs8='Pedigrees' obs9='Letter Grouping';
  datalines;
  obs1 1. . . . . . . . .
  obs2 .828 1. . . . . . . .
  obs3 .776 .779 1. . . . . . . .
  obs4 .439 .493 .460 1. . . . . . . .
  obs5 .432 .464 .425 .674 1. . . . . . . .
  obs6 .447 .489 .443 .590 .541 1. . . . . . . .
  obs7 .447 .432 .401 .381 .402 .288 1. . . . . . . .
  obs8 .541 .537 .534 .350 .367 .320 .555 1. . . . . . . .
  obs9 .380 .358 .359 .424 .446 .325 .598 .452 1. . . . . . . .
;
Using the LINEQS modeling language, you specify the three-term second-order factor analysis model in the following statements:

proc calis data=Thurst nobs=213 corr nose;
  lineqs
    obs1 = x1 * f1 + e1,
    obs2 = x2 * f1 + e2,
    obs3 = x3 * f1 + e3,
    obs4 = x4 * f2 + e4,
    obs5 = x5 * f2 + e5,
    obs6 = x6 * f2 + e6,
    obs7 = x7 * f3 + e7,
    obs8 = x8 * f3 + e8,
    obs9 = x9 * f3 + e9,
    f1 = x10 * f4 + e10,
    f2 = x11 * f4 + e11,
    f3 = x12 * f4 + e12;
  variance
    f4 = 1.,
    e1-e9 = u1-u9,
    e10-e12 = 3 * 1.;
  bounds
    0. <= u1-u9;
run;
In the PROC CALIS statement, you specify the data set in the DATA= option and the number of observations in the NOBS= option. With the CORR option, you request the correlations be analyzed. You use the NOSE option to suppress the computation of standard error estimates.

In the LINEQS statement, the first-order loadings for the three factors, $f_1$, $f_2$, and $f_3$, each refer to three variables, $X_1$–$X_3$, $X_4$–$X_6$, and $X_7$–$X_9$, respectively. One second-order factor, $f_4$, reflects the correlations among the three first-order factors, $f_1$, $f_2$, and $f_3$.

In the VARIANCE statement, you fix the variance of $f_4$ to 1.0 for identification. The variances of error terms $e_1$–$e_9$ are free parameters $u_1$–$u_9$. The error variances for the three first-order factors are also fixed at 1.0 for identification purposes.

You also specify the boundary constraints for the error variance parameters $u_1$–$u_9$. You require them to be positive in the estimation.

Output 29.30.1 shows the estimation results.

Output 29.30.1 Estimation Results of the Second-Order Factor Model for Thurstone Data: LINEQS Model

<table>
<thead>
<tr>
<th>Linear Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs1 = 0.5151 $f_1 + 1.0000 e_1$</td>
</tr>
<tr>
<td>Obs2 = 0.5203 $f_1 + 1.0000 e_2$</td>
</tr>
<tr>
<td>Obs3 = 0.4874 $f_1 + 1.0000 e_3$</td>
</tr>
<tr>
<td>Obs4 = 0.5211 $f_2 + 1.0000 e_4$</td>
</tr>
<tr>
<td>Obs5 = 0.4971 $f_2 + 1.0000 e_5$</td>
</tr>
<tr>
<td>Obs6 = 0.4381 $f_2 + 1.0000 e_6$</td>
</tr>
<tr>
<td>Obs7 = 0.4524 $f_3 + 1.0000 e_7$</td>
</tr>
<tr>
<td>Obs8 = 0.4173 $f_3 + 1.0000 e_8$</td>
</tr>
<tr>
<td>Obs9 = 0.4076 $f_3 + 1.0000 e_9$</td>
</tr>
<tr>
<td>$f_1$ = 1.4438 $f_4 + 1.0000 e_{10}$</td>
</tr>
<tr>
<td>$f_2$ = 1.2538 $f_4 + 1.0000 e_{11}$</td>
</tr>
<tr>
<td>$f_3$ = 1.4065 $f_4 + 1.0000 e_{12}$</td>
</tr>
</tbody>
</table>

| Estimates for Variances of Exogenous Variables |
|------------------|------------------|------------------|
| Variable Type | Variable | Parameter | Estimate |
| Latent | $f_4$ | $u_1$ | 0.18150 |
| Error | $e_1$ | | |
| | $e_2$ | $u_2$ | 0.16493 |
| | $e_3$ | $u_3$ | 0.26713 |
| | $e_4$ | $u_4$ | 0.30150 |
| | $e_5$ | $u_5$ | 0.36450 |
| | $e_6$ | $u_6$ | 0.50642 |
| | $e_7$ | $u_7$ | 0.39033 |
| | $e_8$ | $u_8$ | 0.48138 |
| | $e_9$ | $u_9$ | 0.50509 |
| Disturbance | $e_{10}$ | | 1.00000 |
| | $e_{11}$ | | 1.00000 |
| | $e_{12}$ | | 1.00000 |
Alternatively, you can use the COSAN model specification for analyzing the same data set. First, under the second-order factor model, the covariance structures of the observed variables can be derived as

\[ \Sigma = F1 \times F2 \times P \times F2' \times F1' + F1 \times U2 \times F1' + U1 \]

where \( F1 \) is the 9 \( \times \) 3 first-order loading matrix for the observed variables, \( F2 \) is the 3 \( \times \) 1 second-order loading matrix for the first-order factors, \( P \) is the 1 \( \times \) 1 covariance matrix for the second-order factor \( f4 \), \( U2 \) is the 3 \( \times \) 3 error covariance matrix of the first-order factors \( f1 \)–\( f3 \) (or the covariance matrix of the error terms \( e10 \)–\( e12 \)), and \( U1 \) is the 9 \( \times \) 9 error covariance matrix for the observed variables (or the covariance matrix of the error terms \( e1 \)–\( e9 \)).

Matrix \( F1 \) contains the loading parameters \( x1 \)–\( x9 \) and matrix \( F2 \) contains the loading parameters \( x10 \)–\( x12 \). Because there is only one second-order factor \( f4 \) in the model, matrix \( P \) is a scalar, which is a fixed constant 1 in the LINEQS model. Matrix \( U2 \) is an identity matrix because all error variances are fixed at 1 and they are not correlated. Matrix \( U2 \) is a diagonal matrix that contains the parameters \( u1 \)–\( u9 \). Given this information, you can use the following statements to specify the second-order factor model as a COSAN model:

```
proc calis data=Thurst nobs=213 corr nose;
  cosan
    var = obs1-obs9,
    F1(3) * F2(1) * P(1,IDE) + F1(3) * U2(3,IDE) + U1(9,DIA);
  matrix F1
    [1 , @1] = x1-x3,
    [4 , @2] = x4-x6,
    [7 , @3] = x7-x9;
  matrix F2
    [ ,1] = x10-x12;
  matrix U1
    [1,1] = u1-u9;
  bounds
    0. <= u1-u9;
  vnames
    F1 = [f1 f2 f3],
    F2 = [f4],
    U1 = [e1-e9];
run;
```

In the PROC CALIS statement, you specify the observed variables in the VAR= option and the covariance structures for the observed variables. In the terms of the covariance structure formula, you need to specify the expressions only up the central symmetric matrices. The latter parts of these expressions are redundant and can be generated automatically by PROC CALIS, as shown in Output 29.30.2.

**Output 29.30.2** The Covariance Structures and Model Matrices of the Second-Order Factor Model: COSAN Model

<table>
<thead>
<tr>
<th>COSAN Model Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \Sigma = F1^*F2^*P^*F2'^*F1'^ + F1^*U2^*F1'^ + U1 ]</td>
</tr>
</tbody>
</table>
Output 29.30.2  continued

Summary of Model Matrices

<table>
<thead>
<tr>
<th>Matrix</th>
<th>N Row</th>
<th>N Col</th>
<th>Matrix Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>9</td>
<td>3</td>
<td>GEN: Rectangular</td>
</tr>
<tr>
<td>F2</td>
<td>3</td>
<td>1</td>
<td>GEN: Vector</td>
</tr>
<tr>
<td>P</td>
<td>1</td>
<td>1</td>
<td>IDE: Identity</td>
</tr>
<tr>
<td>U1</td>
<td>9</td>
<td>9</td>
<td>DIA: Diagonal</td>
</tr>
<tr>
<td>U2</td>
<td>3</td>
<td>3</td>
<td>IDE: Identity</td>
</tr>
</tbody>
</table>

Output 29.30.2 shows that the intended covariance structures for the observed variables are being analyzed. The matrix types are shown next. Matrix F1 is a rectangular matrix and matrix F2 is a vector, although they have the default general (GEN) matrix type. Matrices P and U2 are fixed identity (IDE) matrices in the model. For these two matrices, you do not need to specify any of their elements by using the MATRIX statement because they are already well-defined with the IDE type. Lastly, matrix U1 is a diagonal (DIA) matrix in the model.

Output 29.30.3 shows the estimates of the first-order factor loading matrix F1.

Output 29.30.3  Estimation of the F1 Matrix of the Second-Order Factor Model: COSAN Model

<p>| Model Matrix F1 (9 x 3 General Rectangular Matrix) |</p>
<table>
<thead>
<tr>
<th>f1</th>
<th>f2</th>
<th>f3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs1</td>
<td>0.5151</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>[x1]</td>
<td></td>
</tr>
<tr>
<td>Obs2</td>
<td>0.5203</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>[x2]</td>
<td></td>
</tr>
<tr>
<td>Obs3</td>
<td>0.4874</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>[x3]</td>
<td></td>
</tr>
<tr>
<td>Obs4</td>
<td>0</td>
<td>0.5211</td>
</tr>
<tr>
<td></td>
<td>[x4]</td>
<td></td>
</tr>
<tr>
<td>Obs5</td>
<td>0</td>
<td>0.4971</td>
</tr>
<tr>
<td></td>
<td>[x5]</td>
<td></td>
</tr>
<tr>
<td>Obs6</td>
<td>0</td>
<td>0.4381</td>
</tr>
<tr>
<td></td>
<td>[x6]</td>
<td></td>
</tr>
<tr>
<td>Obs7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>[x7]</td>
<td></td>
</tr>
<tr>
<td>Obs8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>[x8]</td>
<td></td>
</tr>
<tr>
<td>Obs9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>[x9]</td>
<td></td>
</tr>
</tbody>
</table>

In the MATRIX statement for F1, you specify the pattern of the loadings. In the first entry of the MATRIX statement, you specify the loadings in the following elements: [1, 1], [2, 1], and [3, 1]. They are free parameters x1–x3, respectively. Notice that the @ sign is necessary in the first entry because the elements being defined would have been [1, 1], [2, 2], and [3, 3] otherwise. The @ sign fixes the column number to 1. See the MATRIX statement for more details about the notation. Similarly, you define the other clusters of loading in the second and third entries in the MATRIX statement for F1. This explains the pattern of factor loadings in Output 29.30.3. These loading estimates x1–x9 match those by the LINEQS model specification, as shown in Output 29.30.1.
Example 29.30: Second-Order Confirmatory Factor Analysis

Output 29.30.3 shows the estimates of the second-order factor loading matrix $F_2$.

**Output 29.30.4** Estimation of the $F_2$ Matrix of the Second-Order Factor Model: COSAN Model

<table>
<thead>
<tr>
<th>Model Matrix $F_2$ (3 x 1 Column Vector)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_4$</td>
</tr>
<tr>
<td>$f_1$ 1.4438</td>
</tr>
<tr>
<td>[x10]</td>
</tr>
<tr>
<td>$f_2$ 1.2538</td>
</tr>
<tr>
<td>[x11]</td>
</tr>
<tr>
<td>$f_3$ 1.4066</td>
</tr>
<tr>
<td>[x12]</td>
</tr>
</tbody>
</table>

In the MATRIX statement for $F_2$, you do not specify the row numbers in the $[1,1]$ specification. PROC CALIS interprets this as stating that all the valid elements in the first column are being specified in the parameter list. In the current example, this means that elements $F_2[1,1]$, $F_2[2,1]$, and $F_2[3,1]$ are filled with the free parameters $x_{10}$, $x_{11}$, and $x_{12}$, respectively. Output 29.30.3 shows these specification and the corresponding estimates, which match those of the LINEQS model specification, as shown in Output 29.30.1.

Output 29.30.5 shows the estimates of the error covariance matrix $U_1$.

**Output 29.30.5** Estimation of the $U_1$ Matrix of the Second-Order Factor Model: COSAN Model

<table>
<thead>
<tr>
<th>Model Matrix $U_1$ (9 x 9 Diagonal Matrix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$ 0.1815</td>
</tr>
<tr>
<td>[u1]</td>
</tr>
<tr>
<td>$e_2$ 0 0.1649</td>
</tr>
<tr>
<td>[u2]</td>
</tr>
<tr>
<td>$e_3$ 0 0 0.2671</td>
</tr>
<tr>
<td>[u3]</td>
</tr>
<tr>
<td>$e_4$ 0 0 0 0.3015</td>
</tr>
<tr>
<td>[u4]</td>
</tr>
<tr>
<td>$e_5$ 0 0 0 0 0.3645</td>
</tr>
<tr>
<td>[u5]</td>
</tr>
<tr>
<td>$e_6$ 0 0 0 0 0 0 0.5064</td>
</tr>
<tr>
<td>[u6]</td>
</tr>
<tr>
<td>$e_7$ 0 0 0 0 0 0 0 0.3903</td>
</tr>
<tr>
<td>[u7]</td>
</tr>
<tr>
<td>$e_8$ 0 0 0 0 0 0 0 0.4814</td>
</tr>
<tr>
<td>[u8]</td>
</tr>
<tr>
<td>$e_9$ 0 0 0 0 0 0 0 0 0.5051</td>
</tr>
<tr>
<td>[u9]</td>
</tr>
</tbody>
</table>

In the MATRIX statement for $U_1$, you specify the diagonal elements of the matrix by using the starting element at $[1,1]$. The parameter assignment proceeds to $[2,2]$, $[3,3]$ and so on such that all the trailing parameters $u_1$–$u_9$ are filled. This means that the last element $U_1[9,9]$ is a free parameter named $u_9$. Output 29.30.5 confirms this intended pattern. Again, all these error variance estimates match those by the LINEQS model specification, as shown in Output 29.30.1.
Example 29.31: Linear Relations among Factor Loadings: COSAN Model Specification

This example reanalyzes the models in Example 29.27 by using the COSAN modeling language. The correlation matrix of six variables from Kinzer and Kinzer (N=326) is used (see Guttman 1957). McDonald (1980) uses this data set to demonstrate the fitting of a factor analysis model with linear constraints on factor loadings. Two factors are assumed for the data. The factor loading matrix $B$ is shown in the following:

$$
B = \begin{pmatrix}
  b_{11} & b_{12} \\
  b_{21} & b_{22} \\
  b_{31} & b_{32} \\
  b_{41} & b_{42} \\
  b_{51} & b_{52} \\
  b_{61} & b_{62}
\end{pmatrix}
$$

The loadings on the second factor are linearly related to the loadings on the first factor, as described by the following formula:

$$
b_{j2} = \alpha - b_{j1}, \quad j = 1, \ldots, 6
$$

The correlation structures are represented by

$$
P = BB' + \Psi
$$

where $\Psi = \text{diag}(\psi_{11}, \psi_{22}, \psi_{33}, \psi_{44}, \psi_{55}, \psi_{66})$ represents the diagonal matrix of unique variances for the variables. Because matrix $P$ is a correlation matrix, its diagonal elements are fixed constants 1. This means that the diagonal elements of the correlation structures must also satisfy the following condition:

$$
\Psi_{jj} = 1 - b_{j1}^2 - b_{j2}^2, \quad j = 1, \ldots, 6
$$

To analyze the correlation structures by using PROC CALIS, you formulate a covariance structure model with such correlation structures embedded in the model. That is, you want to fit the following covariance structure model to the Kinzer data:

$$
\Sigma = DPD' = D(BB' + \Psi)D' = DBB'D' + D\Psi D'
$$

where $D$ is a 6 x 6 diagonal matrix that contains the population standard deviations of the observed variables.
The following statements use the COSAN modeling language to specify this covariance structure model:

```sas
proc calis data=Kinzer nobs=326 nose;
   cosan
      var= var1-var6,
      D(6,DIA) * B(2,GEN) + D(6,DIA) * Psi(6,DIA);
   matrix B
      [ ,1] = b11 b21 b31 b41 b51 b61,
      [ ,2] = b12 b22 b32 b42 b52 b62;
   matrix Psi
      [1,1] = psi1-psi6;
   matrix D
      [1,1] = d1-d6;
   parameters alpha (1.);
   /* SAS Programming Statements to Define Dependent Parameters*/
   /* 6 constraints on the factor loadings */
   b12 = alpha - b11;
   b22 = alpha - b21;
   b32 = alpha - b31;
   b42 = alpha - b41;
   b52 = alpha - b51;
   b62 = alpha - b61;
   /* 6 Constraints on Correlation structures */
   psi1 = 1. - b11 * b11 - b12 * b12;
   psi2 = 1. - b21 * b21 - b22 * b22;
   psi3 = 1. - b31 * b31 - b32 * b32;
   psi4 = 1. - b41 * b41 - b42 * b42;
   psi5 = 1. - b51 * b51 - b52 * b52;
   psi6 = 1. - b61 * b61 - b62 * b62;
   vnames
      D = [var1-var6],
      B = [factor1 factor2],
      Psi = D;
run;
```

In the PROC CALIS statement, you specify the data set by the DATA= option and the number of observations by the NOBS= option. You also use the NOSE option to suppress the printing of the standard error estimates.

In the COSAN statement, you specify the variables for the covariance structure analysis in the VAR= option. Next, you specify the covariance structure formula for the variables. When generating the covariance structure expressions for the terms, PROC CALIS examines the matrix type of the last matrix in each term to determine how the expression is generated. If the last matrix in a term is not a symmetric matrix (including diagonal or identity matrix), the transpose of the last matrix would be included in the expression. This ensures that a symmetric matrix expression is formed for the covariance structures. For example, the first term in the current covariance structure formula is $D(6,DIA) \times B(2,GEN)$. Because $B$ is not a symmetric matrix, the expression generated by PROC CALIS is

$$D \times B \times B' \times D'$$
However, for the second term $D(6, \text{DIA}) \cdot \Psi(6, \text{DIA})$, matrix $\Psi$ is a symmetric matrix so that the expression generated by PROC CALIS is

$$D \cdot \Psi \cdot D'$$

Output 29.31.1 shows the covariance structure model and the model matrices. With $\Psi$ representing the unique variance matrix $\Psi$, the printed covariance structure formula for $\Sigma$ is clearly what you intend to specify.

**Output 29.31.1** The Covariance Structures and Model Matrices: Linearly Constrained Loadings

<table>
<thead>
<tr>
<th>COSAN Model Structures</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma = D^*B^<em>B^<em>D' + D^</em>\Psi</em>D'$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Summary of Model Matrices</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Matrix</strong></td>
<td><strong>N Row</strong></td>
</tr>
<tr>
<td>$B$</td>
<td>6</td>
</tr>
<tr>
<td>$D$</td>
<td>6</td>
</tr>
<tr>
<td>$\Psi$</td>
<td>6</td>
</tr>
</tbody>
</table>

In the MATRIX statements, you specify the parameters in the model matrices. You use parameters with the $b$ prefix to name the two columns of loadings of the factor matrix $B$. You use free parameters $\psi_1$–$\psi_6$ for the diagonal elements of the $\Psi$ matrix, and free parameters $d_1$–$d_6$ for the diagonal elements of the $D$ matrix. Next, you use a PARAMETERS statement to define an independent parameter $\alpha$ in the model. This parameter takes an initial value of 1.0. Using this independent parameter and six SAS programming statements, you define the loadings in the second column of matrix $B$ as functions of the loadings in the first column of the same matrix.

You use six more SAS programming statements to define the unique variance parameters $\psi_1$–$\psi_6$ as dependent parameters of the factor loadings. These constraints ensure that the embedded correlation structures have diagonal elements fixed at 1.0.

Lastly, you use the VNAMES statement to label the column names of the model matrices. The column names of the diagonal matrix $D$ are the same as the observed variables. The column names of matrix $B$ are for the factor names.

As compared with the covariance structure specification (that is, the second specification) by the LINEQS model in Example 29.27, the current COSAN specification seems to be more direct and concise in specifying the parameter constraints. Because of the direct references to the matrix elements in the COSAN modeling language, you can set the required 12 constraints in a very straightforward way as the 12 SAS programming statements in the preceding specification. However, with the LINEQS model specification language in Example 29.27, you need 18 more SAS programming statements to define the correct constraints for the same covariance structure model.
Output 29.31.2 shows the fit summary table. The chi-square test statistic is 14.63 with \( df = 8 \) \((p = 0.067)\). These are the same model fitting results as using the LINEQS model specification, as shown in Output 29.27.4 of Example 29.27.

### Output 29.31.2 Model Fit: Linearly Constrained Loadings with Embedded Correlation Structures

<table>
<thead>
<tr>
<th>Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Chi-Square DF</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
</tr>
</tbody>
</table>

Output 29.31.3 shows the estimation of the loading matrix \( B \). These estimates of factor loadings are essentially the same as those obtained from the LINEQS model specification, as shown in Output 29.27.6, except that the two columns of the loading matrix \( B \) are switched. The column switching is not a concern because the factor labels are arbitrary.

### Output 29.31.3 Estimation of the \( B \) Matrix by the COSAN Model Specification

<table>
<thead>
<tr>
<th>Model Matrix ( B ) (6 x 2 General Rectangular Matrix)</th>
</tr>
</thead>
</table>
| \( \begin{array}{cc}
| \text{factor1} & \text{factor2} \\
| \text{var1}    & 0.6318 \ [b11] \ 0.3422 \ [b12] \\
| \text{var2}    & 0.6531 \ [b21] \ 0.3210 \ [b22] \\
| \text{var3}    & 0.4822 \ [b31] \ 0.4918 \ [b32] \\
| \text{var4}    & 0.3985 \ [b41] \ 0.5755 \ [b42] \\
| \text{var5}    & 0.1971 \ [b51] \ 0.7769 \ [b52] \\
| \text{var6}    & 0.3074 \ [b61] \ 0.6666 \ [b62] \\
\end{array} \) |

Output 29.31.4 shows the estimation of the scaling matrix \( D \). All these standard deviation estimates for the observed variables match those obtained from the LINEQS model specification, as shown in Output 29.27.6.
Output 29.31.4 Estimation of the $D$ Matrix by the COSAN Model Specification

\begin{tabular}{cccccc}
\hline
 & var1 & var2 & var3 & var4 & var5 & var6 \\
\hline
var1 & 1.0077 & 0 & 0 & 0 & 0 & 0 \\
& [d1] & & & & & \\
var2 & 0 & 0.9971 & 0 & 0 & 0 & 0 \\
& [d2] & & & & & \\
var3 & 0 & 0 & 0.9908 & 0 & 0 & 0 \\
& [d3] & & & & & \\
var4 & 0 & 0 & 0 & 0.9909 & 0 & 0 \\
& [d4] & & & & & \\
var5 & 0 & 0 & 0 & 0 & 0.9964 & 0 \\
& [d5] & & & & & \\
var6 & 0 & 0 & 0 & 0 & 0 & 1.0169 \\
& [d6] & & & & & \\
\hline
\end{tabular}

Output 29.31.5 shows the estimation of the unique covariance matrix $\Psi$. All these unique variance parameter estimates match those obtained from the LINEQS model specification, as shown in Output 29.27.6.

Output 29.31.5 Estimation of the $\Psi$ Matrix by the COSAN Model Specification

\begin{tabular}{ccccccc}
\hline
 & var1 & var2 & var3 & var4 & var5 & var6 \\
\hline
var1 & 0.4837 & 0 & 0 & 0 & 0 & 0 \\
& [\psi1] & & & & & \\
var2 & 0 & 0.4705 & 0 & 0 & 0 & 0 \\
& [\psi2] & & & & & \\
var3 & 0 & 0 & 0.5256 & 0 & 0 & 0 \\
& [\psi3] & & & & & \\
var4 & 0 & 0 & 0 & 0.5100 & 0 & 0 \\
& [\psi4] & & & & & \\
var5 & 0 & 0 & 0 & 0 & 0.3576 & 0 \\
& [\psi5] & & & & & \\
var6 & 0 & 0 & 0 & 0 & 0 & 0.4612 \\
& [\psi6] & & & & & \\
\hline
\end{tabular}

Finally, Output 29.31.6 shows the estimation of the independent parameter $\alpha$. The same estimate of $\alpha$ is shown in Output 29.27.6.

Output 29.31.6 Estimation of the Independent Parameter $\alpha$ by the COSAN Model Specification

\begin{tabular}{ccc}
\hline
Additional Parameters & Type & Parameter & Estimate \\
\hline
Independent & $\alpha$ & 0.9740 \\
\hline
\end{tabular}
Example 29.32: Ordinal Relations among Factor Loadings

The same data set as in Example 29.31 is used in McDonald (1980) for analysis with ordinally constrained factor loadings. In Example 29.27, the results of the linearly constrained factor analysis show that the loadings of the two factors are ordered as 2, 1, 3, 4, 6, 5. McDonald (1980) then tests the hypothesis that the factor loadings are all nonnegative and can be ordered in the following manner:

\[
\begin{align*}
    b_{11} & \geq b_{21} \geq b_{31} \geq b_{41} \geq b_{51} \geq b_{61} \\
    b_{12} & \leq b_{22} \leq b_{32} \leq b_{42} \leq b_{52} \leq b_{62}
\end{align*}
\]

In this example, you implement these ordinal relationships by using the LINCON statement in the following COSAN model specification:

```sas
proc calis data=Kinzer nobs=326 nose;
    cosan
        var= var1-var6,
            D(6,DIA) * B(2,GEN) + D(6,DIA) * Psi(6,DIA);
    matrix B
        [ ,1]= b11 b21 b31 b41 b51 b61,
        [ ,2]= 0. b22 b32 b42 b52 b62;
    matrix Psi
        [1,1]= psi1-psi6;
    matrix D
        [1,1]= d1-d6 ;
    lincon
        b61 <= b51,
        b51 <= b41,
        b41 <= b31,
        b31 <= b21,
        b21 <= b11,
        0. <= b22,
        b22 <= b32,
        b32 <= b42,
        b42 <= b52,
        b52 <= b62;
    /* SAS Programming Statements */
    /* 6 Constraints on Correlation structures */
    psi1 = 1. - b11 * b11;
    psi2 = 1. - b21 * b21 - b22 * b22;
    psi3 = 1. - b31 * b31 - b32 * b32;
    psi4 = 1. - b41 * b41 - b42 * b42;
    psi5 = 1. - b51 * b51 - b52 * b52;
    psi6 = 1. - b61 * b61 - b62 * b62;
    vnames
        B = [factor1 factor2],
        Psi = [var1-var6],
        D = Psi;
run;
```
As in Example 29.31, correlation structures are analyzed in the current example so that the unique variance parameters $\psi_1-\psi_6$ are defined as functions of the loadings in the SAS programming statements. However, the loading parameters are no longer not constrained in the current model. Instead, you impose ordinal constraints on the loading parameters. First, $b_{21}$ is fixed at 0 for identification purposes. Then, you use the LINCON statement to specify the ordinal relations of the factor loadings.

As shown in Output 29.32.1, the solution converges in 12 iterations. In the fit summary table, the chi-square test statistic is 8.48 ($df = 6, p = 0.20$). This indicates a good fit. However, in the model there are 11 loading parameters (the $b$’s) and 6 population standard deviation parameters (the $d$’s). The degrees of freedom should have been $4 = 21 - 11 - 6$, but why is this number 6 in the fit summary table?

**Output 29.32.1** Final Iteration Status and Fit

<table>
<thead>
<tr>
<th>Optimization Results</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>12</td>
</tr>
<tr>
<td>Function Calls</td>
<td>29</td>
</tr>
<tr>
<td>Jacobian Calls</td>
<td>14</td>
</tr>
<tr>
<td>Active Constraints</td>
<td>2</td>
</tr>
<tr>
<td>Objective Function</td>
<td>0.0260990149</td>
</tr>
<tr>
<td>Max Abs Gradient Element</td>
<td>2.7626747E-6</td>
</tr>
<tr>
<td>Lambda</td>
<td>0</td>
</tr>
<tr>
<td>Actual Over Pred Change</td>
<td>1.1572072766</td>
</tr>
<tr>
<td>Radius</td>
<td>0.0000851592</td>
</tr>
</tbody>
</table>

Convergence criterion (ABSGCONV=0.00001) satisfied.

<table>
<thead>
<tr>
<th>Fit Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>8.4822</td>
</tr>
<tr>
<td>Chi-Square DF</td>
<td>6</td>
</tr>
<tr>
<td>Pr &gt; Chi-Square</td>
<td>0.2049</td>
</tr>
</tbody>
</table>

The reason is that there are two active constraints in the solution, resulting in two free parameters fewer in the final solution than originally specified. Active constraints are those inequality constraints that are fulfilled on the boundary equalities. As shown in the “Optimization Results” table, the number of active constraints for the current fitting is two. The default treatment in PROC CALIS is to treat these active constraints as if they were going to happen for all possible repeated sampling. This might as well be seen as fitting the active equality constraints on every possible repeated sample. This results in an increase of the degrees of freedom for model fit, as adjusted in the current fit summary table in Output 29.32.1. To warn you about the degrees-of-freedom adjustment, the following messages are also printed with the output:

**WARNING:** There are 2 active boundary or linear inequality constraints at the solution. The standard errors and chi-square test statistic assume that the solution is located in the interior of the parameter space; hence, they do not apply if it is likely that some different set of inequality a constraints could be active.

**NOTE:** The degrees of freedom are increased by the number of active constraints. The number of parameters in calculating fit indices is decreased by the number of active constraints. To turn off the adjustment, use the NOADJDF option.
When active constraints are encountered, you need to be cautious about two implications. First, the estimates fall on the boundary of the parameter space originally specified. As shown in Output 29.32.2, estimates for \( b_{11} \) and \( b_{21} \) are the same, and so are the pair of estimates for \( b_{52} \) and \( b_{62} \). These pairs of parameters were originally constrained by inequalities in the model. For example, \( b_{62} \) was constrained to be at least as large as \( b_{52} \). The fact that this constraint is honored only on the bound means that a better model fit might exist with \( b_{62} \) being smaller than \( b_{52} \). Similarly, a better model fit might result without requiring \( b_{11} \) to be at least as large as \( b_{21} \). Therefore, solutions with active boundary constraints might imply that the original strict inequality constraints are not appropriate for the data.

**Output 29.32.2** Estimation of the Factor Loading Matrix B

<table>
<thead>
<tr>
<th>Model Matrix B</th>
</tr>
</thead>
<tbody>
<tr>
<td>(6 x 2 General Rectangular Matrix)</td>
</tr>
<tr>
<td>factor1</td>
</tr>
<tr>
<td>var1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>var2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>var3</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>var4</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>var5</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>var6</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Output 29.32.3** Estimation of the Scaling Matrix D and Unique Covariance Matrix Psi

<table>
<thead>
<tr>
<th>Model Matrix D</th>
</tr>
</thead>
<tbody>
<tr>
<td>(6 x 6 Diagonal Matrix)</td>
</tr>
<tr>
<td>var1</td>
</tr>
<tr>
<td>var1</td>
</tr>
<tr>
<td>var2</td>
</tr>
<tr>
<td>var3</td>
</tr>
<tr>
<td>var4</td>
</tr>
<tr>
<td>var5</td>
</tr>
<tr>
<td>var6</td>
</tr>
</tbody>
</table>
The second implication for the presence of active constraints is that the chi-square test statistic and the standard error estimates are computed as if repeated samples were fitted by the model with the presence of the active equality constraints. The degrees-of-freedom adjustment by PROC CALIS is based on this assumption. However, if the particular active constraints reflect only a rare sampling event, the degrees-of-freedom adjustment (or even the computation of the chi-square statistic and standard error estimates) might not be justified. Unfortunately, whether the active constraints are reflecting the truth of the model or pure sampling fluctuation is usually difficult to determine.

**Example 29.33: Longitudinal Factor Analysis**

The following example (McDonald 1980) illustrates both the ability of PROC CALIS to formulate complex covariance structure analysis problems by the generalized COSAN matrix model and the use of programming statements to impose nonlinear constraints on the parameters. The example is a longitudinal factor analysis that uses the Swaminathan (1974) model. For \( m = 3 \) tests, \( k = 3 \) occasions, and \( r = 2 \) factors, the covariance structure model is formulated as follows:

\[
\Sigma = F_1 F_2 F_3 L F_3^{-1} F_2^{-1} P (F_2^{-1})' (F_3^{-1})' L' F_2' F_1' + U^2
\]

\[
F_1 = \begin{pmatrix} B_1 & B_2 \\ B_3 \end{pmatrix}, \quad F_2 = \begin{pmatrix} I_2 & D_2 \\ D_2 \end{pmatrix}, \quad F_3 = \begin{pmatrix} I_2 & I_2 \\ I_2 & D_3 \end{pmatrix}
\]

\[
L = \begin{pmatrix} I_2 & 0 & 0 \\ I_2 & I_2 & 0 \\ I_2 & I_2 & I_2 \end{pmatrix}, \quad P = \begin{pmatrix} I_2 & S_2 \\ S_2 & S_3 \end{pmatrix}, \quad U = \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ U_{21} & U_{22} & U_{23} \\ U_{31} & U_{32} & U_{33} \end{pmatrix}
\]

\[
S_2 = I_2 - D_2^2, \quad S_3 = I_2 - D_3^2
\]

The Swaminathan longitudinal factor model assumes that the factor scores for each \( m \) common factor change from occasion to occasion \( (k) \) according to a first-order autoregressive scheme. The matrix \( F_1 \)
contains the $k$ factor loading matrices $B_1, B_2,$ and $B_3$ (each is $n \times m$). The matrices $D_2, D_3, S_2, S_3$ and $U_{ij}, i, j = 1, \ldots, k,$ are diagonal, and the matrices $D_i$ and $S_i, i = 2, \ldots, k,$ are subjected to the constraint

$$S_i + D_i^2 = I$$

Although the covariance structure model looks pretty complicated, it poses no problem for the COSAN model specifications. Since the constructed correlation matrix given by McDonald (1980) is singular, only unweighted least squares (METHOD=LS) estimates can be computed. The following statements specify the COSAN model for the correlation structures.

```
Title "Swaminathan's Longitudinal Factor Model, Data: MCDONALD(1980)";
Title2 "Constructed Singular Correlation Matrix, GLS & ML not possible";
data Mcdon(TYPE=CORR);
  _TYPE_ = 'CORR'; INPUT _NAME_ $ obs1-obs9;
datalines;
obs1 1.000 . . . . . . . .
obs2 .100 1.000 . . . . . . .
obs3 .250 .400 1.000 . . . . .
obs4 .720 .108 .270 1.000 . . . .
obs5 .135 .740 .380 .180 1.000 . . . .
obs6 .270 .318 .800 .360 .530 1.000 . . . .
obs7 .650 .054 .135 .730 .090 .180 1.000 . . . .
obs8 .108 .690 .196 .144 .700 .269 .200 1.000 . . . .

proc calis data=Mcdon method=ls nobs=100 corr;
cosan
  var = obs1-obs9,
  F1(6,GEN) * F2(6,DIA) * F3(6,DIA) * L(6,LOW) * F3(6,DIA,INV)
    * F2(6,DIA,INV) * P(6,DIA) + U(9,SYM);
matrix F1
  [1 , @1] = x1-x3,
  [2 , @2] = x4-x5,
  [4 , @3] = x6-x8,
  [5 , @4] = x9-x10,
  [7 , @5] = x11-x13,
  [8 , @6] = x14-x15;
matrix F2
  [1,1]= 2 * 1. x16 x17 x16 x17;
matrix F3
  [1,1]= 4 * 1. x18 x19;
matrix L
  [1,1]= 6 * 1. ,
  [3,1]= 4 * 1. ,
  [5,1]= 2 * 1. ;
matrix P
  [1,1]= 2 * 1. x20-x23;
matrix U
  [1,1]= x24-x32,
  [4,1]= x33-x38,
  [7,1]= x39-x41;
bounds 0. <= x24-x32,
```
In the PROC CALIS statement, you use the NOBS= option to specify the number of observations. The CORR option requests the analysis of the correlation matrix.

In the COSAN statement, you list the observed variables for the analysis in the VAR= option. Then you specify the formula for the covariance structures. Notice that in the covariance structure formula, some matrices are specified twice. That is, matrix F2 and F3 appear in two different places. Matrices with the same name means that they are identical—which certainly makes sense. In addition, you can apply different transformations to the same matrix in different locations of the matrix formula. For example, you do not transform matrix F2 in the first location, but the same matrix is inverted (INV) later in the expression. Similarly for matrix F3.

Next, you define the parameters in the six distinct model matrices by six MATRIX statements. Each matrix has some specific patterns under the covariance structure model. For the F1 matrix, it has the following pattern for the free parameters in the model:

```
col1  col2  col3  col4  col5  col6
row1  x
row2  x  x
row3  x  x
row4  x
row5  x  x
row6  x  x
row7  x
row8  x  x
row9  x  x
```

To specify these parameters, you can use some shorthand notation in the MATRIX statement. For example, in the first entry of the MATRIX statement for matrix F1, you use the notation \([1, @1]\). This means that the parameter specification starts with the \([1, 1]\) element and proceeds to the next element while fixing the column number at 1. Hence, parameters x1–x3 are specified for the F1[1, 1], F1[2, 1], and F1[3, 1] elements, respectively. Similarly, you specify other parameters in the F1 matrix in a column by column fashion.

If you do not use the @ sign in the specification, the parameters are assigned differently. For example, in the specification of the L matrix, the first entry in the corresponding MATRIX statement also starts with the \([1, 1]\) element. But it proceeds down to \([2, 2]\), \([3, 3]\), and so on because the @ sign is not used to fix any column or row number. As a result, the MATRIX statement for L specifies the following pattern:

```
col1  col2  col3  col4  col5  col6
row1  1  
row2  1  
row3  1  1   
row4  1  1  
row5  1  1  1  
row6  1  1  1  1
```
The unspecified elements are fixed zeros in the model.

Similarly, you specify the diagonal matrices \( F_2 \), \( F_3 \), and \( P \), and the symmetric matrix \( U \).

You also set bounds for some parameters in the BOUNDS statement and some dependent parameters in the SAS programming statements.

Output 29.33.1 shows the correlation structures and the model matrices in the analysis. All appear to be intended.

**Output 29.33.1** The Correlation Structures and Model Matrices of the Longitudinal Factor Model

<table>
<thead>
<tr>
<th>COSAN Model Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigma = ( F_1 F_2 F_3 L \inv(F_3) P \inv(F_2) \inv(F_3) ) + ( U )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Summary of Model Matrices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix</td>
</tr>
<tr>
<td>F1</td>
</tr>
<tr>
<td>F2</td>
</tr>
<tr>
<td>F3</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>U</td>
</tr>
</tbody>
</table>

PROC CALIS finds a converged solution for the estimation problem. Output 29.33.2, Output 29.33.3, and Output 29.33.4 show the estimation results of the \( F_1 \), \( F_2 \), and \( F_3 \) matrices, respectively.

**Output 29.33.2** Estimation of the \( F_1 \) Matrix of the Longitudinal Factor Model

<table>
<thead>
<tr>
<th>Model Matrix ( F_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(9 x 6 General Rectangular Matrix)</td>
</tr>
</tbody>
</table>

| obs1 | 0.3515 | 0 | 0 | 0 | 0 | 0 |
| obs2 | 0.2871 | 0.9528 | 0 | 0 | 0 | 0 |
| obs3 | 0.7101 | 0.2059 | 0 | 0 | 0 | 0 |
| obs4 | 0 | 0 | 0.4204 | 0 | 0 | 0 |
| obs5 | 0 | 0 | 0.4303 | 0.9027 | 0 | 0 |
| obs6 | 0 | 0 | 0.8591 | 0.1772 | 0 | 0 |
| obs7 | 0 | 0 | 0 | 0 | 0.3487 | 0 |
| obs8 | 0 | 0 | 0 | 0 | 0.5924 | -0.1971 |
| obs9 | 0 | 0 | 0 | 0 | 0.9987 | 0.0871 |
**Output 29.33.3** Estimation of the F2 Matrix of the Longitudinal Factor Model

<table>
<thead>
<tr>
<th>Model Matrix F2</th>
<th>(6 x 6 Diagonal Matrix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Col1</td>
<td>Col2</td>
</tr>
<tr>
<td>Row1</td>
<td>1.0000</td>
</tr>
<tr>
<td>Row2</td>
<td>0</td>
</tr>
<tr>
<td>Row3</td>
<td>0</td>
</tr>
<tr>
<td>Row4</td>
<td>0</td>
</tr>
<tr>
<td>Row5</td>
<td>0</td>
</tr>
<tr>
<td>Row6</td>
<td>0</td>
</tr>
</tbody>
</table>

**Output 29.33.4** Estimation of the F3 Matrix of the Longitudinal Factor Model

<table>
<thead>
<tr>
<th>Model Matrix F3</th>
<th>(6 x 6 Diagonal Matrix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Col1</td>
<td>Col2</td>
</tr>
<tr>
<td>Row1</td>
<td>1.0000</td>
</tr>
<tr>
<td>Row2</td>
<td>0</td>
</tr>
<tr>
<td>Row3</td>
<td>0</td>
</tr>
<tr>
<td>Row4</td>
<td>0</td>
</tr>
<tr>
<td>Row5</td>
<td>0</td>
</tr>
<tr>
<td>Row6</td>
<td>0</td>
</tr>
</tbody>
</table>

Output 29.33.5 shows the estimation results of the L matrix, which is a fixed matrix that contains only 0 or 1 for its elements.

**Output 29.33.5** Estimation of the L Matrix of the Longitudinal Factor Model

<table>
<thead>
<tr>
<th>Model Matrix L</th>
<th>(6 x 6 Lower Triangular Matrix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Col1</td>
<td>Col2</td>
</tr>
<tr>
<td>Row1</td>
<td>1.0000</td>
</tr>
<tr>
<td>Row2</td>
<td>0</td>
</tr>
<tr>
<td>Row3</td>
<td>1.0000</td>
</tr>
<tr>
<td>Row4</td>
<td>0</td>
</tr>
<tr>
<td>Row5</td>
<td>1.0000</td>
</tr>
<tr>
<td>Row6</td>
<td>0</td>
</tr>
</tbody>
</table>
Output 29.33.6 shows the estimation results of the $P$ matrix. Notice that parameter estimate $x_{23}$ falls on the lower boundary at zero.

**Output 29.33.6** Estimation of the $P$ Matrix of the Longitudinal Factor Model

<table>
<thead>
<tr>
<th>Model Matrix $P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(6 x 6 Diagonal Matrix)</td>
</tr>
<tr>
<td>Col1</td>
</tr>
<tr>
<td>Row1 1.0000</td>
</tr>
<tr>
<td>Row2 0</td>
</tr>
<tr>
<td>Row3 0</td>
</tr>
<tr>
<td>Row4 0</td>
</tr>
<tr>
<td>Row5 0</td>
</tr>
<tr>
<td>Row6 0</td>
</tr>
</tbody>
</table>

In fact, PROC CALIS routinely checks for zero values for the estimates on the diagonal of the central symmetric matrices. In this case, you get the following messages regarding the estimation of matrix $P$:

**WARNING**: Although all predicted variances for the observed variables are positive, the corresponding predicted covariance matrix is not positive definite. It has one negative eigenvalue.

**WARNING**: The estimated variance of variable 6 is essentially zero in the central matrix $P$ of term 1 of the COSAN model.

**WARNING**: The central matrix $P$ of term 1 of the COSAN model is not positive definite. It has one zero eigenvalue.

Output 29.33.7 shows the estimation results of the $U$ matrix. Parameter estimates $x_{28}$ and $x_{32}$ fall on the lower boundary at zero. PROC CALIS issues the following messages regarding the estimation of matrix $U$:

**WARNING**: The estimated variance of obs5 is essentially zero in the central matrix $U$ of term 2 of the COSAN model.

**WARNING**: The estimated variance of obs9 is essentially zero in the central matrix $U$ of term 2 of the COSAN model.

**WARNING**: The central matrix $U$ of term 2 of the COSAN model is not positive definite. It has 3 negative eigenvalues.
Because this formulation of Swaminathan’s model in general leads to an unidentified problem, the results given here are different from those reported by McDonald (1980). The displayed output of PROC CALIS also indicates that the fitted central model matrices $P$ and $U$ are not positive-definite. The BOUNDS statement constrains the diagonals of the matrices $P$ and $U$ to be nonnegative, but this cannot prevent $U$ from having three negative eigenvalues. The fact that many of the published results for more complex models in covariance structure analysis are connected to unidentified problems implies that more theoretical work should be done to study the general features of such models.
References


References


<table>
<thead>
<tr>
<th>Subject Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>absolute fit indices, 1269, 1270</td>
</tr>
<tr>
<td>active constraints, 1270</td>
</tr>
<tr>
<td>ADF method</td>
</tr>
<tr>
<td>CALIS procedure, 1466, 1468</td>
</tr>
<tr>
<td>approximate</td>
</tr>
<tr>
<td>standard errors (CALIS), 1216, 1237, 1475, 1520</td>
</tr>
<tr>
<td>asymptotic covariance</td>
</tr>
<tr>
<td>CALIS procedure, 1206, 1467</td>
</tr>
<tr>
<td>asymptotically distribution free estimation</td>
</tr>
<tr>
<td>CALIS procedure, 1224, 1468</td>
</tr>
<tr>
<td>asymptotically distribution-free</td>
</tr>
<tr>
<td>CALIS procedure, 1466</td>
</tr>
<tr>
<td>baseline model chi-square, 1269</td>
</tr>
<tr>
<td>baseline model chi-square degrees of freedom, 1269</td>
</tr>
<tr>
<td>bifactor model</td>
</tr>
<tr>
<td>CALIS procedure, 1741</td>
</tr>
<tr>
<td>bifactor models</td>
</tr>
<tr>
<td>CALIS procedure, 1736</td>
</tr>
<tr>
<td>bifactor models example (CALIS), 1736</td>
</tr>
<tr>
<td>biquartimax method, 1260, 1261</td>
</tr>
<tr>
<td>biquartimin method, 1261</td>
</tr>
<tr>
<td>CALIS procedure</td>
</tr>
<tr>
<td>ADF method, 1466, 1468</td>
</tr>
<tr>
<td>approximate standard errors, 1216, 1237, 1475, 1520</td>
</tr>
<tr>
<td>asymptotic covariance, 1206, 1467</td>
</tr>
<tr>
<td>asymptotically distribution free estimation, 1224, 1468</td>
</tr>
<tr>
<td>asymptotically distribution-free, 1466</td>
</tr>
<tr>
<td>bifactor model</td>
</tr>
<tr>
<td>CALIS procedure, 1741</td>
</tr>
<tr>
<td>bifactor models, 1736</td>
</tr>
<tr>
<td>case-level residuals, 1494</td>
</tr>
<tr>
<td>chi-square, adjusted, 1485</td>
</tr>
<tr>
<td>coefficient of determination, 1231</td>
</tr>
<tr>
<td>comparing competing models, 1685, 1718</td>
</tr>
<tr>
<td>computational problems, 1515–1517</td>
</tr>
<tr>
<td>computational problems, identification, 1511, 1517, 1818</td>
</tr>
<tr>
<td>confidence interval, 1520</td>
</tr>
<tr>
<td>confirmatory factor analysis, 1190, 1597, 1608, 1668</td>
</tr>
<tr>
<td>constraints, 1241, 1274, 1316, 1436–1438</td>
</tr>
<tr>
<td>constraints, program statements, 1357</td>
</tr>
<tr>
<td>constraints, programming statements, 1434</td>
</tr>
<tr>
<td>COSAN, 1390</td>
</tr>
<tr>
<td>COSAN Model, 1390</td>
</tr>
<tr>
<td>COSAN model, 1242, 1390, 1392</td>
</tr>
<tr>
<td>degrees of freedom, 1215, 1225</td>
</tr>
<tr>
<td>determination coefficients, 1256</td>
</tr>
<tr>
<td>determination index, 1493</td>
</tr>
<tr>
<td>diagonally weighted least squares, 1469</td>
</tr>
<tr>
<td>direct covariance structures, 1665, 1680, 1685</td>
</tr>
<tr>
<td>direct effect, 1257</td>
</tr>
<tr>
<td>direct maximum likelihood, 1223</td>
</tr>
<tr>
<td>direct robust estimation, 1472</td>
</tr>
<tr>
<td>discrepancy function, 1464</td>
</tr>
<tr>
<td>DWLS method, 1469</td>
</tr>
<tr>
<td>effects, 1257</td>
</tr>
<tr>
<td>EQS Model, 1401</td>
</tr>
<tr>
<td>EQS program, 1184</td>
</tr>
<tr>
<td>estimating covariances, 1539</td>
</tr>
<tr>
<td>estimating covariances and means, 1544</td>
</tr>
<tr>
<td>estimation criteria, 1468</td>
</tr>
<tr>
<td>estimation methods, 1463, 1464, 1469, 1473</td>
</tr>
<tr>
<td>exploratory factor analysis, 1190</td>
</tr>
<tr>
<td>FACTOR, 1394</td>
</tr>
<tr>
<td>factor analysis model, 1258</td>
</tr>
<tr>
<td>factor analysis model, COSAN statement, 1391</td>
</tr>
<tr>
<td>factor loadings, 1382</td>
</tr>
<tr>
<td>FACTOR Model, 1394</td>
</tr>
<tr>
<td>FACTOR procedure, 1259, 1516</td>
</tr>
<tr>
<td>factor rotation, 1260</td>
</tr>
<tr>
<td>factor scores, 1381, 1384, 1521</td>
</tr>
<tr>
<td>FIML method, 1465</td>
</tr>
<tr>
<td>fit function, 1464</td>
</tr>
<tr>
<td>fitted covariance matrix, 1464</td>
</tr>
<tr>
<td>fitted mean vector, 1464</td>
</tr>
<tr>
<td>full information maximum likelihood, 1465, 1629, 1639</td>
</tr>
<tr>
<td>full information maximum likelihood and ML, 1639</td>
</tr>
<tr>
<td>Generalized COSAN Model, 1390</td>
</tr>
<tr>
<td>generalized least squares, 1464</td>
</tr>
<tr>
<td>GLS method, 1464</td>
</tr>
<tr>
<td>gradient, 1371, 1475, 1508</td>
</tr>
<tr>
<td>hessian, 1475</td>
</tr>
<tr>
<td>Hessian matrix, 1371, 1477, 1508</td>
</tr>
<tr>
<td>hierarchical factor model, 1741</td>
</tr>
<tr>
<td>hierarchical factor models, 1736</td>
</tr>
<tr>
<td>higher-order factor model, 1737</td>
</tr>
<tr>
<td>higher-order factor models, 1736</td>
</tr>
<tr>
<td>indirect effect, 1257</td>
</tr>
<tr>
<td>information matrix, 1475</td>
</tr>
<tr>
<td>initial values, 1234, 1506</td>
</tr>
<tr>
<td>input data set, 1367</td>
</tr>
</tbody>
</table>
iteratively reweighted least squares, 1470
kappa, 1505
kurtosis, 1158, 1209, 1218, 1504, 1506
latent growth curve models, 1730
latent variables, 1158, 1366
leverage observations, 1494
likelihood ratio test, 1485, 1503
linear constraints, 1750, 1804
linear regression, 1553
LINEQS, 1401
LINEQS model, 1275
LISMOD, 1408
LISMOD model, 1282
LISREL, 1408
LM tests, 1286
longitudinal factor analysis, 1812
manifest variables, 1158
masking effects of outlier, 1497
matrix inversion, 1477
matrix transformation, COSAN model, 1244
matrix types, COSAN model, 1244
maximum likelihood, 1464
measurement errors, 1574, 1580, 1586, 1591
missing patterns, 1220, 1226, 1238, 1629
ML method, 1464
MODEL procedure, 1516
modeling languages, 1354
modification indices, 1216, 1224, 1226, 1476, 1501–1503, 1521
MSTRUCT, 1416
MSTRUCT Model, 1416
MSTRUCT model, 1313
multiple-group analysis, 1759
multivariate regression, 1557
naming parameters, 1433
naming variables, 1433
observation-level residuals, 1494
ODS graph names, 1538
optimization, 1160, 1217, 1218, 1227, 1228, 1238, 1507–1509, 1514
optimization history, 1510
optimization statements, 1198
optimization, initial values, 1506, 1509
optimization, memory problems, 1509
ordinal constraints, 1809
outlier detection, 1494
output data sets, 1371
output table names, 1523
p value, 1520
parallel test items, 1608
parameter names, 1433
PATH, 1419
path analysis, 1644, 1708, 1718
path diagram, 1332, 1440
PATH model, 1323
predicted covariance matrix, 1385, 1520
predicted covariance model matrix, 1517
predicted mean vector, 1520
prefix-name, 1277
RAM, 1425
RAM model, 1348, 1392
reciprocal causation, 1215, 1500
reciprocal paths, 1658
REG procedure, 1516
renaming parameters, 1356
residual diagnostics, 1232, 1616
residuals, 1482
robust, 1158
robust estimation, 1235, 1470, 1616
robust residual diagnostics, 1497
SCORE procedure, 1229, 1231, 1384
second-order factor model, 1737
significance level, 1205
simplicity functions, 1260
singularity criterion, 1237
singularity criterion, covariance matrix, 1206, 1225, 1239
skewness, 1504
squared multiple correlation, 1493, 1521
stability coefficient, 1215, 1500
step length, 1217
structural equation, 1231, 1358
subsidiary group specification statements, 1196
subsidiary model specification statements, 1197
SYSLIN procedure, 1516
SYSNLIN procedure, 1516
r value, 1476, 1520
tau-equivalent items, 1608
test indices, constraints, 1224
testing parametric functions, 1357, 1358
testing sphericity, 1547, 1550
testing uncorrelatedness, 1545, 1550
total effect, 1257
two-stage robust estimation, 1471
ULS method, 1464
unweighted least squares, 1464
variable names, 1433
variable selection, 1439
Wald test, probability limit, 1237
weight matrix input, 1217, 1218
weighted least squares, 1466
WLS method, 1466
case-level residuals
CALIS procedure, 1494
chi-square
adjusted (CALIS), 1485
chi-square corrections, 1209
coefficient
of determination (CALIS), 1231
comparing competing models
   CALIS Procedure, 1685, 1718
comparing competing models (CALIS), 1718
comparing modeling languages example (CALIS), 1708
compatibility with the TCALIS Procedure (CALIS), 1169
computational problems
   CALIS procedure, 1515
   convergence (CALIS), 1516
   identification (CALIS), 1511, 1517, 1818
   negative eigenvalues (CALIS), 1517
   negative R-square (CALIS), 1517
   overflow (CALIS), 1515
   singular predicted covariance model (CALIS), 1517
   time (CALIS), 1517
confidence interval
   displaying (CALIS), 1520
confirmatory factor analysis
   CALIS Procedure, 1190, 1668
confirmatory factor analysis example (CALIS), 1190, 1597, 1608, 1668
conjugate
   gradient algorithm (CALIS), 1217, 1218, 1227, 1238, 1508
constraints
   boundary (CALIS), 1241, 1436
   linear (CALIS), 1274, 1438
   modification indices (CALIS), 1224, 1226
   nonlinear (CALIS), 1316
   ordered (CALIS), 1437
   program statements (CALIS), 1357
   programming statements (CALIS), 1434
   test indices (CALIS), 1224
COSAN model
   CALIS procedure, 1242, 1390, 1392
   comparing modeling languages example (CALIS), 1784
COSAN models example (CALIS), 1784
linear constraints example (CALIS), 1804
longitudinal factor analysis example (CALIS), 1812
ordinal constraints example (CALIS), 1809
second-order confirmatory factor models example (CALIS), 1799
COSAN models example (CALIS), 1784
covariance structure analysis model, see COSAN model
covarimin method, 1261
Crawford-Ferguson method, 1260, 1261
degrees of freedom
   CALIS procedure, 1215, 1225
determination coefficients (CALIS)
   dependent variables, 1256
determination index
   CALIS procedure, 1493
diagonally weighted least squares
   CALIS procedure, 1469
direct covariance structures
   CALIS Procedure, 1665, 1680, 1685
direct covariance structures example (CALIS), 1192, 1545, 1547, 1550, 1665, 1680, 1685
direct effect
   CALIS procedure, 1257
direct maximum likelihood
   CALIS procedure, 1223
direct robust estimation
   CALIS procedure, 1472
double dogleg
   algorithm (CALIS), 1217, 1227, 1238, 1508
   DWLS method
   CALIS procedure, 1469
effects
   CALIS procedure, 1257
effects (CALIS), 1257
EQS program
   CALIS procedure, 1184
equamax method, 1260, 1261
estimating covariances and means example (CALIS), 1544
estimating covariances example (CALIS), 1539
estimation criteria
   CALIS procedure, 1468
estimation methods
   CALIS procedure, 1463, 1464, 1469, 1473
exploratory factor analysis
   CALIS Procedure, 1190
exploratory factor analysis example (CALIS), 1190
factor analysis model
   COSAN statement (CALIS), 1391
   identification (CALIS), 1258
factor loadings
   CALIS procedure, 1382
FACTOR model
   confirmatory factor analysis example (CALIS), 1190, 1597, 1608, 1668
   exploratory factor analysis example (CALIS), 1190
   full information maximum likelihood example (CALIS), 1629, 1639
   linear constraints example (CALIS), 1750
   residual diagnostics example (CALIS), 1616
   robust estimation example (CALIS), 1616
factor parsimax method, 1260, 1261
FACTOR procedure
  CALIS procedure, 1259
factor scores
  CALIS procedure, 1381, 1384, 1521
displaying (CALIS), 1521
FIML method
  CALIS procedure, 1465
fitted covariance matrix
  CALIS procedure, 1464
fitted mean vector
  CALIS procedure, 1464
full information maximum likelihood
  CALIS procedure, 1465
full information maximum likelihood and ML FIML (CALIS), 1639
full information maximum likelihood example (CALIS), 1629, 1639
generalized Crawford-Ferguson method, 1260, 1261
generalized least squares
  CALIS procedure, 1464
GLS method
  CALIS procedure, 1464
gradient
  CALIS procedure, 1371, 1508
Hessian matrix
  CALIS procedure, 1371, 1477, 1508
hierarchical factor model
  CALIS procedure, 1741
hierarchical factor models
  CALIS procedure, 1736
higher-order factor model
  CALIS procedure, 1737
higher-order factor models
  CALIS procedure, 1736
higher-order factor models example (CALIS), 1736
incremental fit indices, 1269
indirect effect
  CALIS procedure, 1257
initial values
  CALIS procedure, 1214, 1234, 1506
iteratively reweighted least squares
  CALIS procedure, 1470
kurtosis
  CALIS procedure, 1158, 1209, 1218, 1504, 1506
Lagrange multiplier
test, modification indices (CALIS), 1224, 1502, 1503
latent growth curve models
  CALIS Procedure, 1730
latent variables
  CALIS procedure, 1158, 1366
Levenberg-Marquardt algorithm
  CALIS procedure, 1217, 1227, 1508
leverage observations
  CALIS procedure, 1494
likelihood ratio test
  CALIS procedure, 1485, 1503
linear constraints
  CALIS Procedure, 1750, 1804
linear constraints example (CALIS), 1750, 1804
linear regression example (CALIS), 1553
LINEQS model
  bifactor models example (CALIS), 1736
  CALIS procedure, 1275
  comparing modeling languages example (CALIS), 1184, 1708, 1784
  higher-order factor models example (CALIS), 1736
  latent growth curve models example (CALIS), 1730
  measurement errors example (CALIS), 1591
  reciprocal paths example (CALIS), 1658
  second-order confirmatory factor models example (CALIS), 1799
  structural model example (CALIS), 1188
LISMOD
  structural model example (CALIS), 1189
LISMOD model
  CALIS procedure, 1282
  comparing modeling languages example (CALIS), 1184, 1708
LM tests
  CALIS procedure, 1286
longitudinal factor analysis
  CALIS Procedure, 1812
longitudinal factor analysis example (CALIS), 1812
manifest variables
  CALIS procedure, 1158
masking effects of outlier
  CALIS procedure, 1497
matrix
  inversion (CALIS), 1477
matrix properties
  COSAN model (CALIS), 1244
matrix types
  COSAN model (CALIS), 1244
Mauchly’s test of sphericity, 1210, 1212
maximum likelihood
  CALIS procedure, 1464
measurement errors example (CALIS), 1574, 1580, 1586, 1591
missing patterns
  FIML (CALIS), 1220, 1226, 1238, 1629
ML method
  CALIS procedure, 1464
modeling language (CALIS), 1194
modification indices
  CALIS procedure, 1216, 1476, 1501
  constraints (CALIS), 1224, 1226
  displaying (CALIS), 1521
  Lagrange multiplier test (CALIS), 1224, 1502, 1503
  Wald test (CALIS), 1224, 1503
MSTRUCT model
  CALIS procedure, 1313
  direct covariance structures example (CALIS), 1192, 1545, 1547, 1550, 1665, 1680, 1685
  estimating covariances and means example (CALIS), 1544
  estimating covariances example (CALIS), 1539
multiple-group analysis
  CALIS Procedure, 1759
  multiple-group analysis (CALIS), 1759
  multivariate regression example (CALIS), 1557
Newton-Raphson algorithm
  CALIS procedure, 1217, 1218, 1227, 1508
oblimin method, 1261
observation-level residuals
  CALIS procedure, 1494
ODS graph names
  CALIS procedure, 1538
optimization
  CALIS procedure, 1160, 1507
  conjugate gradient (CALIS), 1217, 1218, 1227, 1238, 1508
  double dogleg (CALIS), 1217, 1227, 1238, 1508
  history (CALIS), 1510
  initial values (CALIS), 1506, 1509
  Levenburg-Marquardt (CALIS), 1217, 1227, 1508
  line search (CALIS), 1218, 1514
  memory problems (CALIS), 1509
  Newton-Raphson (CALIS), 1217, 1218, 1227, 1508
  nonlinear constraints (CALIS), 1509
  quasi-Newton (CALIS), 1217, 1218, 1227, 1238, 1508, 1509
  step length (CALIS), 1514
  trust region (CALIS), 1228, 1508
  trust-region (CALIS), 1217
  update method (CALIS), 1238
  optimization statements (CALIS), 1198
  ordinal constraints
    CALIS Procedure, 1809
  ordinal constraints example (CALIS), 1809
  orthomax method, 1260
outlier detection
  CALIS procedure, 1464
output data sets
  CALIS procedure, 1371
output table names
  CALIS procedure, 1523

p value
  displaying (CALIS), 1520
parallel test items (CALIS), 1608
parametric functions (CALIS)
  tests, 1357, 1358
parsimax method, 1261
parsimonious fit indices, 1269
path analysis
  CALIS Procedure, 1644, 1708, 1718
  path analysis example (CALIS), 1184, 1644
  path diagram
    CALIS procedure, 1440
    path diagram (CALIS), 1332
    path diagram (CALIS)
      structural model example, 1185
  path diagram example (CALIS), 1604, 1644

PATH model
  CALIS procedure, 1323
  comparing competing models example (CALIS), 1718
  comparing modeling languages example (CALIS), 1184, 1708
  linear regression example (CALIS), 1553
  measurement errors example (CALIS), 1574, 1580, 1586
  multiple-group analysis example (CALIS), 1759
  multivariate regression example (CALIS), 1557
  path analysis example (CALIS), 1557
  residual diagnostics example (CALIS), 1616
  robust estimation example (CALIS), 1616
  structural model example (CALIS), 1186
predicted covariance matrix
  CALIS procedure, 1385
  displaying (CALIS), 1520
predicted covariance model matrix
  singular (CALIS), 1517
predicted mean vector
  displaying (CALIS), 1520
prefix name
  LINEQS statement (CALIS), 1277
predicted mean vector
  constraints (CALIS), 1357, 1434
quartimax method, 1261
quartimin method, 1262
quasi-Newton algorithm
CALIS procedure, 1217, 1218, 1227, 1238, 1508, 1509

RAM model
CALIS procedure, 1348, 1392
comparing modeling languages example (CALIS), 1184, 1708, 1784
structural model example (CALIS), 1187

reciprocal causation
CALIS procedure, 1215, 1500
reciprocal paths
CALIS Procedure, 1658
reciprocal paths example (CALIS), 1658
renaming parameters
CALIS procedure, 1356
residual diagnostics
CALIS procedure, 1232
residual diagnostics example (CALIS), 1616
residuals
CALIS procedure, 1482

reticular action model, see RAM model
robust
CALIS procedure, 1158
robust estimation
CALIS procedure, 1235, 1470
robust estimation example (CALIS), 1616
robust residual diagnostics
CALIS procedure, 1497

SCORE procedure
CALIS procedure, 1229, 1231, 1384
second-order confirmatory factor models example (CALIS), 1799
second-order factor model
CALIS procedure, 1737
significance level
CALIS procedure, 1205
simplicity functions
CALIS procedure, 1260
singularity criterion
CALIS procedure, 1237
covariance matrix (CALIS), 1206, 1225, 1239
skewness
CALIS procedure, 1504
squared multiple correlation
CALIS procedure, 1493, 1521
stability coefficient
CALIS procedure, 1215, 1500
step length
CALIS procedure, 1217
structural equation (CALIS)
definition, 1231
dependent variables, 1358

structural model example (CALIS), 1184
LINEQS model, 1188
LISMOD, 1189
path diagram, 1185
PATH model, 1186
RAM model, 1187
subsidiary group specification statements (CALIS), 1196
subsidiary model specification statements (CALIS), 1197

$t$ value
CALIS procedure, 1476
displaying (CALIS), 1520
tau-equivalent items (CALIS), 1608
TCALIS procedure(CALIS), 1169
test indices
constraints (CALIS), 1224
test of a covariance matrix against a diagonal pattern, 1210, 1212
test of a covariance matrix against a fixed matrix, 1210
test of compound symmetry, 1209, 1211
test of equal variances and equal covariances, 1209, 1211
test of equality of covariance matrices, 1209, 1212
test of equality of mean vectors, 1221
test of uncorrelatedness, 1210, 1212
test of uniform means, 1221
test of zero means, 1222
test the H pattern of a covariance matrix, 1210
testing sphericity example (CALIS), 1547, 1550
testing uncorrelatedness example (CALIS), 1545, 1550
total effect
CALIS procedure, 1257
trust-region algorithm
CALIS procedure, 1217, 1228, 1508
two-stage robust estimation
CALIS procedure, 1471

ULS method
CALIS procedure, 1464
unweighted least squares
CALIS procedure, 1464

variable selection
CALIS procedure, 1439
varimax method, 1261

Wald test
modification indices (CALIS), 1224, 1503
probability limit (CALIS), 1237
weight matrix input
CALIS procedure, 1217, 1218
weighted least squares
CALIS procedure, 1466
WLS method
  CALIS procedure, 1466
Syntax Index

ALL option
  PROC CALIS statement, 1230

ALLNEWPARMS option
  REFMODEL statement, 1355

ALPHA= option
  PROC CALIS statement, 1205

ALPHAECV= option
  FITINDEX statement, 1268
  PROC CALIS statement, 1205

ALPHALEV= option
  PROC CALIS statement, 1205

ALPHAOUT= option
  PROC CALIS statement, 1205

ALPHARMSE= option
  PROC CALIS statement, 1205

ALPHARMSE= option
  FITINDEX statement, 1268

ARRANGE= option
  PATHDIAGRAM statement, 1334

ASINGULAR= option
  PROC CALIS statement, 1206

ASYCOV= option
  PROC CALIS statement, 1206

BASEFIT= option
  FITINDEX statement, 1268
  PROC CALIS statement, 1206

BASEFUNC= option
  FITINDEX statement, 1268

BIASKUR option
  PROC CALIS statement, 1209

BOUNDS statement
  CALIS procedure, 1241

BY statement
  CALIS procedure, 1241

CALIS procedure, 1195
  syntax, 1195

CALIS procedure, BOUNDS statement, 1241
CALIS procedure, BY statement, 1241
CALIS procedure, COSAN statement, 1242
CALIS procedure, COV statement, 1252
CALIS procedure, DETERM statement, 1256
CALIS procedure, EFFPART statement, 1257
CALIS procedure, FACTOR statement, 1258
  COMPONENT option, 1259
  GAMMA= option, 1259
  HEYWOOD option, 1259

N= option, 1259
NORM option, 1260
RCONVERGE= option, 1260
RITER= option, 1260
ROTATE= option, 1260

CALIS procedure, FITINDEX statement
  ALPHAECV= option, 1268
  ALPHARMSEA= option, 1268

CALIS procedure, FREQ statement, 1271
CALIS procedure, GROUP statement, 1272
  LABEL= option, 1273
  NAME= option, 1273
CALIS procedure, LINCON statement, 1274
CALIS procedure, LINEQS statement, 1275
CALIS procedure, LISMOD statement, 1282
CALIS procedure, LMTESTS statement, 1286
  default option, 1286
  lmmat option, 1286
  maxrank option, 1286
  nodefault option, 1286
  norank option, 1286

CALIS procedure, main model specification
  statements, 1197
CALIS procedure, MATRIX statement, 1295
CALIS procedure, MEAN statement, 1309
CALIS procedure, model analysis statements, 1198
CALIS procedure, MODEL statement, 1311
  GROUP= option, 1312
  GROUPS= option, 1312
  LABEL= option, 1312
  NAME= option, 1312
CALIS procedure, MSTRUCT statement, 1313
CALIS procedure, NLINCON statement, 1316
CALIS procedure, NOPTIONS statement, 1316
CALIS procedure, optimization statements, 1198
CALIS procedure, OUTFILE statement, 1317
CALIS procedure, OUTFILES statement, 1317
CALIS procedure, PARAMETERS statement, 1319
CALIS procedure, PARTIAL statement, 1322
CALIS procedure, PATH statement, 1323
CALIS procedure, PATHDIAGRAM statement, 1332
ARRANGE= option, 1334
DECP= option, 1335
DECPFIT= option, 1335
DESTROYER= option, 1335
DIAGRAM= option, 1336
DIAGRAMLABEL= option, 1336
EMPHSTRUCT option, 1337
ERRORSIZE= option, 1337
EXOGCOV option, 1337
FACTORSIZE= option, 1337
FITINDEX= option, 1338
LABEL= option, 1339
MEANPARM= option, 1339
MODEL= option, 1339
NOCOV option, 1340
NOERRCOV option, 1340
NOERRVAR option, 1340
NOESTIM option, 1340
NOEXOGCOV option, 1340
NOEXOGVAR option, 1340
NOFITTABLE option, 1340
NOFLAG option, 1340
NOMANIPARM option, 1340
NOMEAN option, 1341
NOTITLE option, 1341
NOVARIANCE option, 1341
OMITPATH= option, 1341
PARAMNAME option, 1341
STRUCTADD= option, 1342
STRUCTURAL option, 1342
TITLE= option, 1343
USEERROR option, 1343
VARPARN= option, 1343
CALIS procedure, PCOV statement, 1343
CALIS procedure, PROC CALIS statement, 1200
 ALL option, 1230
ALPHA= option, 1205
ALPHAECV= option, 1205
ALPHALEV= option, 1205
ALPHAOUT= option, 1205
ALPHARMS= option, 1205
ASINGULAR= option, 1206
ASYCOV= option, 1206
BASEFIT= option, 1206
BASEFUNC= option, 1207
BIASKUR option, 1209
CHICORR= option, 1209
CHICORRECT= option, 1209
Cl option, 1209
CLOSEFIT option, 1211
CORR option, 1230
CORRELATION option, 1211
COVARIANCE option, 1211
COVPATTERN= option, 1211
COVSING= option, 1214
DATA= option, 1214
DEMPHAS= option, 1214
DFE= option, 1215
DFR= option, 1234
DFREDUCE= option, 1215
EDF= option, 1215
ESTDATA= option, 1216
EXTENDPATH option, 1215
FCONV= option, 1215
FTOL= option, 1215
G4= option, 1216
GCONV= option, 1216
GTOL= option, 1216
INBASEFIT= option, 1206
INEST= option, 1216
INMODEL= option, 1216
INRAM= option, 1216
INSTEP= option, 1217
INVARI= option, 1216
INWTG= option, 1217
INWTGINV option, 1218
KURTOSIS option, 1218
LINESEARCH= option, 1218
LSPRECISION= option, 1219
MAXFUN= option, 1219
MAXITER= option, 1219
MAXLEVERAGE= option, 1220
MAXMISSPAT= option, 1220
MAXOUTLIER= option, 1220
MEANPATTERN= option, 1221
MEANSTR option, 1223
METHOD= option, 1223
MODIFICATION option, 1224
MSINGULAR= option, 1225
NOADJDF option, 1225
NOBS= option, 1225
NOINDEXTYPE option, 1225
NOMEANSTR option, 1225
NOMISSPAT option, 1226
NOMOD option, 1226
NOPRINT option, 1226
NOROUTERSPEC option, 1226
NOPARMNAME option, 1226
NOPRINT option, 1226
NOSTDERR option, 1226
OAT= option, 1227
OMETHOD= option, 1227
ORDERALL option, 1228
ORDERGROUPS option, 1228
ORDERMODELS option, 1228
DEMPHAS= option
    PROC CALIS statement, 1214
DESTR= option
    PATHDIAGRAM statement, 1335
DETERM statement, CALIS procedure, 1256
DFE= option
    PROC CALIS statement, 1215
DFR= option
    PROC CALIS statement, 1234
DFREDUCE= option
    FITINDEX statement, 1268
    PROC CALIS statement, 1215
DIAGRAM= option
    PATHDIAGRAM statement, 1336
DIAGRAMLABEL= option
    PATHDIAGRAM statement, 1336
EDF= option
    PROC CALIS statement, 1215
EFFPART statement, CALIS procedure, 1257
EMPHSTRUCT option
    PATHDIAGRAM statement, 1337
ERRORSIZE= option
    PATHDIAGRAM statement, 1337
ESTDATA= option
    PROC CALIS statement, 1216
EXOGCOV option
    PATHDIAGRAM statement, 1337
EXTENDPATH option
    PROC CALIS statement, 1215
FACTOR statement, CALIS procedure, 1258
FACTORSIZE= option
    PATHDIAGRAM statement, 1337
FCONV= option
    PROC CALIS statement, 1215
FITINDEX= option
    PATHDIAGRAM statement, 1338
FREQ statement
    CALIS procedure, 1271
FTOL= option
    PROC CALIS statement, 1215
G4= option
    PROC CALIS statement, 1216
GAMMA= option
    FACTOR statement, 1259
GCONV= option
    PROC CALIS statement, 1216
GROUP statement
    CALIS procedure, 1272
GROUP= option
    MODEL statement, 1312
GROUPS= option
    MODEL statement, 1312
GTOL= option
    PROC CALIS statement, 1216
HEYWOOD option
    FACTOR statement (CALIS), 1259
INBASEFIT= option
    FITINDEX statement, 1268
    PROC CALIS statement, 1206
INEST= option
    PROC CALIS statement, 1216
INMODEL= option
    PROC CALIS statement, 1216
INRAM= option
    PROC CALIS statement, 1216
INSTEP= option
    PROC CALIS statement, 1217
INVAR= option
    PROC CALIS statement, 1216
INWGT= option
    PROC CALIS statement, 1216
INWGTINV option
    PROC CALIS statement, 1218
KURTOSIS option
    PROC CALIS statement, 1218
LABEL= option
    GROUP statement, 1273
    MODEL statement, 1312
    PATHDIAGRAM statement, 1339
LINCON statement, CALIS procedure, 1274
LINEQS statement, CALIS procedure, 1275
LINESEARCH= option
    PROC CALIS statement, 1218
LISMOD statement, CALIS procedure, 1282
Immat option
    LMTESTS statement, 1286
    LMTESTS statement, CALIS procedure, 1286
LSPRECISION= option
    PROC CALIS statement, 1219
main model specification statements, CALIS procedure, 1197
MATRIX statement
    CALIS procedure, 1295
MAXFUNC= option
    PROC CALIS statement, 1219
MAXITER= option
    PROC CALIS statement, 1219
MAXLEVERAGE= option
    PROC CALIS statement, 1219
MAXMISSPAT= option
    PROC CALIS statement, 1220
MAXMISSPAT= option
    PROC CALIS statement, 1220
MAXOUTLIER= option
PROC CALIS statement, 1220
maxrank option
LMTESTS statement, 1286
MEAN= option
CALIS procedure, 1309
MEANPARAM= option
PATHDIAGRAM statement, 1339
MEANPATTERN= option
PROC CALIS statement, 1221
MEANSTR option
PROC CALIS statement, 1223
METHOD= option
PROC CALIS statement, 1223
model analysis statements, CALIS procedure, 1198
MODEL statement
CALIS procedure, 1311
MODEL= option
PATHDIAGRAM statement, 1339
MODIFICATION option
PROC CALIS statement, 1224
MSINGULAR= option
PROC CALIS statement, 1225
MSTRUCT statement, CALIS procedure, 1313
N= option
FACTOR statement (CALIS), 1259
NAME= option
GROUP statement, 1273
MODEL statement, 1312
NLAGCON statement, CALIS procedure, 1316
NLOPTIONS statement, CALIS procedure, 1316
NOADJDF option
FITINDEX statement, 1269
PROC CALIS statement, 1225
NOIDENT option
PATHDIAGRAM statement, 1286
NOMEAN option
PATHDIAGRAM statement, 1286
NOMISSPAT option
PROC CALIS statement, 1225
NOMOD option
PROC CALIS statement, 1226
NOORDERSPEC option
PROC CALIS statement, 1226
NOPARMNAME option
PROC CALIS statement, 1226
NOSAMP option
PATHDIAGRAM statement, 1340
NOSIG option
PATHDIAGRAM statement, 1340
NOSOL option
PATHDIAGRAM statement, 1340
NOSPEC option
PATHDIAGRAM statement, 1340
NOSUMMARY option
PATHDIAGRAM statement, 1340
NOVAR option
PATHDIAGRAM statement, 1340
OFSIZE option
FITINDEX statement, 1269
PROC CALIS statement, 1226
OM= option
PROC CALIS statement, 1227
OMETHOD= option
PROC CALIS statement, 1227
OMITPATH= option
PATHDIAGRAM statement, 1341
ONLVAR option
FITINDEX statement, 1269
order analysis statements, CALIS procedure, 1198
ORDERALL option
PROC CALIS statement, 1228
ORDERGROUPS option
PROC CALIS statement, 1228
ORDERMODELS option
PROC CALIS statement, 1228
ORDERSPEC option
PROC CALIS statement, 1229
OUTEST= option
PROC CALIS statement, 1229
OUTFILE statement
CALIS procedure, 1317
OUTFILES statement
   CALIS procedure, 1317
OUTFIT option
   PROC CALIS statement, 1229
OUTFIT= option
   FITINDEX statement, 1271
OUTMODEL= option
   PROC CALIS statement, 1229
OUTRAM= option
   PROC CALIS statement, 1229
OUTSTAT= option
   PROC CALIS statement, 1229
OUTVAR= option
   PROC CALIS statement, 1229
OUTWGT= option
   PROC CALIS statement, 1230
PALL option
   PROC CALIS statement, 1230
PARAMETERS statement
   CALIS procedure, 1319
PARM_PREFIX option
   REFMODEL statement, 1355
PARM_SUFFIX option
   REFMODEL statement, 1355
PARMNAME option
   PATHDIAGRAM statement, 1341
   PROC CALIS statement, 1230
PARTIAL statement
   CALIS procedure, 1322
PATH statement, CALIS procedure, 1323
PATHDIAGRAM statement
   CALIS procedure, 1332
PCORR option
   PROC CALIS statement, 1230
PCOV statement, CALIS procedure, 1343
PCOVES option
   PROC CALIS statement, 1230
PDETERM option
   PROC CALIS statement, 1231
PESTIM option
   PROC CALIS statement, 1231
PINITIAL option
   PROC CALIS statement, 1231
PLATCOV option
   PROC CALIS statement, 1231
PLOTS= option
   PROC CALIS statement, 1232
PRIMAT option
   PROC CALIS statement, 1233
PRINT option
   PROC CALIS statement, 1233
PROC CALIS statement, see CALIS procedure
PSHORT option
   PROC CALIS statement, 1233
PSUMMARY option
   PROC CALIS statement, 1233
PVAR statement, CALIS procedure, 1345
PWEIGHT option
   PROC CALIS statement, 1233
RADIUS= option
   PROC CALIS statement, 1234
RAM statement, CALIS procedure, 1348
RANDOM= option
   PROC CALIS statement, 1234
RCONVERGE= option
   FACTOR statement (CALIS), 1260
RDF= option
   PROC CALIS statement, 1234
READADDPARM= option
   PROC CALIS statement, 1234
REFMODEL statement
   CALIS procedure, 1354
   PROC CALIS statement, 1234
RENAMEPARM statement, CALIS procedure, 1356
RESIDUAL= option
   PROC CALIS statement, 1234
RIDGE= option
   PROC CALIS statement, 1235
RITER= option
   FACTOR statement (CALIS), 1260
ROBITER= option
   PROC CALIS statement, 1235
ROBPHI= option
   PROC CALIS statement, 1235
ROBUST option
   PROC CALIS statement, 1235
ROBUST= option
   PROC CALIS statement, 1235
ROTATE= option
   FACTOR statement (CALIS), 1260
SALPHA= option
   PROC CALIS statement, 1237
SHORT option
   PROC CALIS statement, 1233
SIMPLE option
   PROC CALIS statement, 1237
SIMTESTS statement, CALIS procedure, 1357
SINGULAR= option
   PROC CALIS statement, 1237
SLMW= option
   PROC CALIS statement, 1237
SMETHOD= option
   PROC CALIS statement, 1237
SPRECISION= option
   PROC CALIS statement, 1218
SPRECISION= option
   PROC CALIS statement, 1219, 1237
START= option
   PROC CALIS statement, 1237

STD statement, CALIS procedure, 1358

STDERR option
   PROC CALIS statement, 1237

STRUCTADD= option
   PATHDIAGRAM statement, 1342

STRUCTEQ statement, CALIS procedure, 1358

STRUCTURAL option
   PATHDIAGRAM statement, 1342

subsidiary group specification statements, CALIS procedure, 1196

subsidiary model specification statements, CALIS procedure, 1197

SUMMARY option
   PROC CALIS statement, 1233

TAU= option
   FACTOR statement, 1262

TECHNIQUE= option
   PROC CALIS statement, 1227

TESTFUNC statement, CALIS procedure, 1358

TITLE= option
   PATHDIAGRAM statement, 1343

TMISSPAT= option
   PROC CALIS statement, 1238

TOTEFF option
   PROC CALIS statement, 1215

UPDATE= option
   PROC CALIS statement, 1238

USEERROR option
   PATHDIAGRAM statement, 1343

VAR statement
   CALIS procedure, 1359

VARDEF= option
   PROC CALIS statement, 1239

VARIANCE statement, CALIS procedure, 1362

VARNAMES statement, CALIS procedure, 1366

VARPARM= option
   PATHDIAGRAM statement, 1343

VSINGULAR= option
   PROC CALIS statement, 1239

WEIGHT statement
   CALIS procedure, 1367

WPENALTY= option
   PROC CALIS statement, 1239

WRIDGE= option
   PROC CALIS statement, 1240

XCONV= option
   PROC CALIS statement, 1240

XTOL= option
   PROC CALIS statement, 1240