THE POWER TO KNOW®

SAS/STAT® 9.2 User's Guide The TCALIS Procedure (Experimental) (Book Excerpt)

SAS® Documentation

Chapter 88

The TCALIS Procedure (Experimental)

Contents

Overview: TCALIS 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 TCALIS, 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 TCALIS with maximum likelihood (default) or generalized least squares estimation. If the number of observations is sufficiently large, Browne's asymptotically distribution-free (ADF) estimation method can be used.

You can use the TCALIS 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

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

- FACTOR—supports the input of factor-variable relations
- LINEQS—like the EQS program (Bentler 1995), uses equations to describe variable relationships
- LISMOD—utilizes LISREL (Jöreskog and Sörbom 1985) model matrices for defining models
- MSTRUCT—supports direct parameterization 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)
- 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 when one modeling language is more convenient than the others. This will be discussed in the section "Which Modeling Language?" on page 6706.

In addition to basic model specification, you can set various parameter constraints in PROC TCALIS. 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 TCALIS 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
- weight matrix and its descriptive properties

After a model is fitted and accepted by the researcher, PROC TCALIS 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
- 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 TCALIS procedure:

- diagonally weighted least squares (DWLS, with optional weight matrix input)
- generalized least squares (GLS, with optional weight matrix input)
- maximum likelihood (ML, for multivariate normal data); this is the default method
- 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 TCALIS do not exhaust all alternatives in the field. For example, partial least squares (PLS) is not implemented. See the section "Estimation Criteria" on page 6880 for details about estimation criteria used in PROC TCALIS. 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 TCALIS. For general path analysis with latent variables, consider using PROC TCALIS.

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 TCALIS 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 (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 TCALIS behave reasonably well in most common applications.

With initial estimates, PROC TCALIS 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 TCALIS 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 TCALIS:

- 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)

In addition to the ability to save output tables as data sets by using the ODS OUTPUT statement, PROC TCALIS 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, 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 TCALIS 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 TCALIS procedure uses ODS Graphics to create graphs as part of its output. High-quality residual histograms are available in PROC TCALIS. See Chapter 21, "Statistical Graphics Using ODS," for general information about ODS Graphics. See the section "ODS Graphics" on page 6944 and the PLOTS= option on page 6733 for specific information about the statistical graphics available with the TCALIS procedure.

A Guide to the PROC TCALIS Documentation

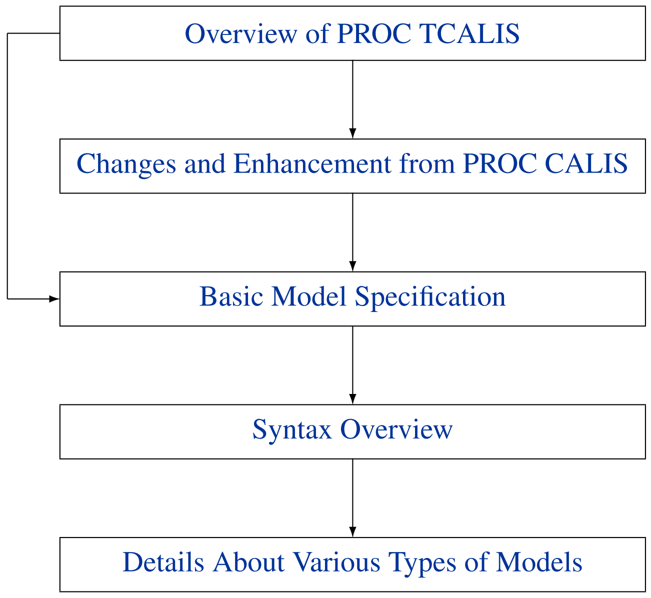
The TCALIS procedure fits structural equation models using a variety of modeling languages. This chapter provides documentation for all of them. Additionally, some sections provide introductions to the model specification, the theory behind the software, and other technical details. While some introductory 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 the books by Bollen (1989b) and Loehlin (2004). Therefore, reading this chapter sequentially is not a good strategy for learning about PROC TCALIS. This section provides a guide or "road map" to the rest of the PROC TCALIS 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 skills levels of using the TCALIS procedure (basic, intermediate, and advanced) and their milestones:

Level	**Milestone**
Basic	You are able to specify simple models, but might make mistakes
Intermediate	You are able to specify more sophisticated models with few syntactic and semantic mistakes
Advanced	You are able to use the advanced options provided by PROC TCALIS

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

Guide to the Basic Skills Level


Overview of PROC TCALIS
Changes and Enhancement from PROC CALIS
Basic Model Specification
Syntax Overview
Details About Various Types of Models

Overview of PROC TCALIS

The section "Overview: TCALIS Procedure" on page 6673 gives you an overview of the PROC TCALIS procedure and an overall picture of the TCALIS procedure but without the details.

Changes and Enhancement from PROC CALIS

Read the section "Changes and Enhancement from PROC CALIS" on page 6687 if you have previous experience with PROC CALIS, the predecessor of PROC TCALIS. If you have not used PROC CALIS previously, you may skip this section.

Basic Model Specification

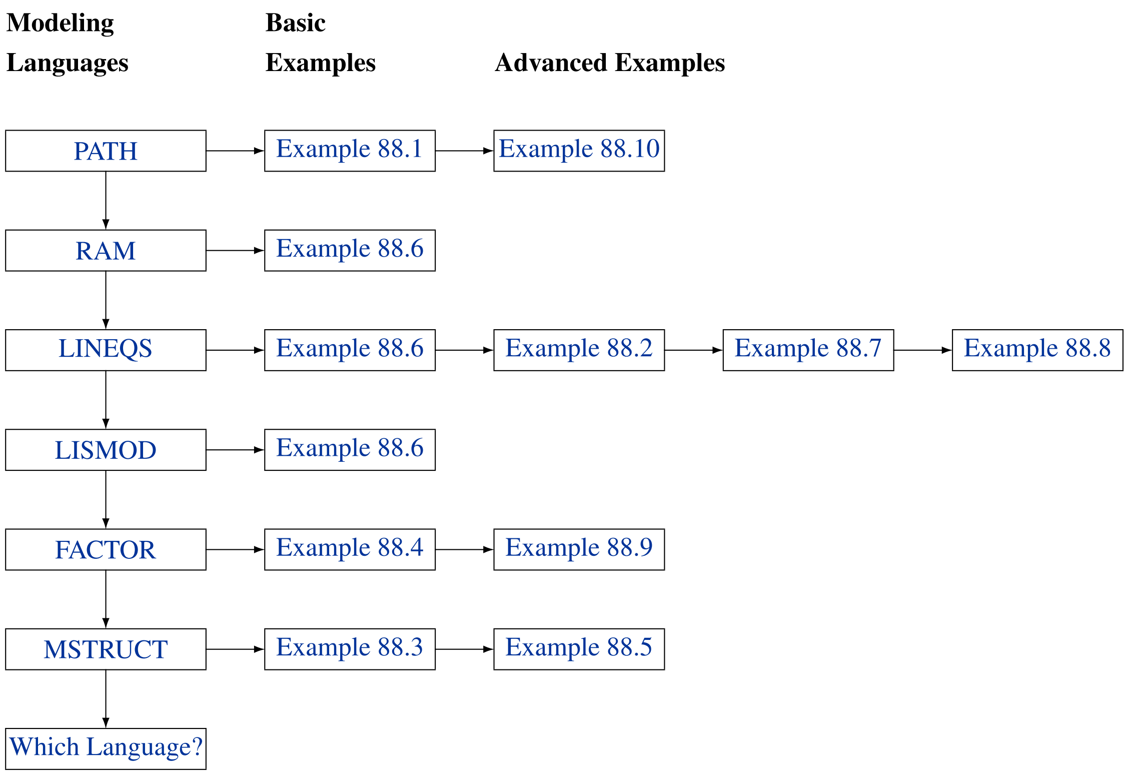
The structural equation example in the section "Getting Started: TCALIS Procedure" on page 6696 provides the best starting point for learning the basic model specification. You learn how to represent your theory by using a path diagram and then translate the diagram into the PATH model for PROC TCALIS 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 6698. The corresponding results are shown and discussed in Example 88.1.

After you learn about the PATH modeling language and an example of its application, you can either continue to learn more modeling languages or skip to the syntax overview. However, you do not need to learn all of the modeling languages in PROC TCALIS. Any one of the modeling languages (LINEQS, LISMOD, PATH, or RAM) is sufficient for specifying a very wide class of structural equation models. PROC TCALIS provides different kinds of modeling languages because different researchers might have previously learned different modeling languages or approaches.

You can go to the next step for an overview of the PROC TCALIS syntax and learn other modeling languages at a later time. See "Getting Started: TCALIS Procedure" on page 6696 for a brief overview of the various modeling languages. In these examples, model specifications are shown and explained, but no analysis results are discussed. Each subsection discusses the same model but uses a different modeling language. In the subsections "LISMOD Model" on page 6701, "LINEQS Model" on page 6700, and "RAM Model" on page 6699, you learn how to specify a structural equation model equivalently by using the LINEQS, LISMOD, and RAM modeling languages, respectively. In the subsection "A Factor Model Example" on page 6702, you learn how to specify a confirmatory factor model by using the FACTOR modeling language. In the subsection "Direct Covariance Structures Analysis" on page 6704, you learn how to specify the covariance structures directly on the covariance matrix by using the MSTRUCT modeling language.

After studying the examples in the "Getting Started: TCALIS Procedure" section, you can strengthen your understanding of the various modeling languages by studying more examples such as those in section "Examples: TCALIS Procedure" on page 6945. Unlike the examples in the "Getting Started: TCALIS Procedure" section, examples in the "Examples: TCALIS Procedure" section include the analysis results, in addition to the explanations of the model specifications. The following figure provides a guide to studying the examples in the "Examples: TCALIS Procedure" section.

Figure 88.1 A Guide for Studying the Examples



In this guide, the various modeling languages are listed in sequence. However, only the PATH modeling language is a prerequisite of other modeling languages. You are not required to learn the languages in the order presented in this guide. For example, you can skip the RAM modeling language and learn the LINEQS model specification after you finish the PATH model language. Similarly, you can learn the LISMOD language without going through the RAM or LINEQS modeling languages if you are familiar with the traditional Keesling-Wiley-Jöreskog measurement and structural models (Keesling 1972; Wiley 1973; Jöreskog 1973).

You can learn each modeling language by studying the examples in the "Getting Started: TCALIS Procedure" sections and in the "Examples: TCALIS Procedure" sections. Examples are listed for each modeling language with the basic examples listed before the more advanced examples. The basic examples are required, and the more advanced examples can be skimmed in the first reading. You might need to revisit the more advanced examples as your understanding grows. A summary of the examples is presented next.

Example 88.1. Path Analysis: Stability of Alienation

In this example, you learn 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, initial model specification, initial estimation method, optimization history and results, residual analyses, residual graphics, estimation results, squared multiple correlations, and standardized results.

Example 88.2. Simultaneous Equations with Mean Structures and Reciprocal Paths

In this econometric example, you learn 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 meta-parameters by using the PARAMETERS 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 TCALIS statement.

Example 88.3. A Direct Covariance Structures Model

In this example, you 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 88.4. Confirmatory Factor Analysis: Cognitive Abilities

Confirmatory factor analysis is illustrated in this example by using the FACTOR modeling language. In addition, you use the MODIFICATION option in the PROC TCALIS statement to compute LM test indices for model modifications.

Example 88.5. Testing Equality of Two Covariance Matrices Using A Multiple-Group Analysis

In this example, a simple multiple-group analysis is illustrated by the MSTRUCT modeling language. You also learn how to use the `ods select` statement to customize your printed output.

Example 88.6. Illustrating Various General Modeling Languages

This example extends Example 88.1, 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 88.1 and get equivalent results. This example shows the connections between the general modeling languages supported in PROC TCALIS. A good understanding of Example 88.1 is a prerequisite for this example.

Example 88.7. Fitting a Latent Growth Curve Model

This is an advanced example that illustrates the use of structural equation modeling techniques for fitting latent growth curve models. In this example, 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.

Example 88.8. Higher-Order and Hierarchical Factor Models

This 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 and the SAS programming statements. Relationships between the second-order factor model and the bifactor model are numerically illustrated.

Example 88.9. Linear Relations Among Factor Loadings

This is an advanced example of a first-order confirmatory factor analysis by using the FACTOR modeling language. In this example, you learn how to use the PARAMETERS and the 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 88.10. A Multiple-group Model for Purchasing Behavior

This 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 a SAS macro to organize your model specification, the REFMODEL statement to reference properly defined models, and the SIMTEST statement to test a priori simultaneous hypotheses.

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 6706 provides some guidelines and suggestions.

Syntax Overview

Now that you have some basic understanding of the modeling languages of PROC TCALIS, you can look at the global structure of PROC TCALIS specifications. The section "Syntax: TCALIS Procedure" on page 6707 shows the syntactic structure of PROC TCALIS. However, reading the "Syntax: TCALIS Procedure" section sequentially might not be a good strategy. The statements used in PROC TCALIS are classified in the section "Classes of Statements" on page 6708. Understanding this section is a prerequisite for understanding single-group and multiple-group analyses in PROC TCALIS. Syntax for single-group analyses is described in the section "Single-Group Analysis Syntax" on page 6711, and syntax for multiple-group analyses is described in the section "Multiple-Group Multiple-Model Analysis Syntax" on page 6712.

You might also want to get an overview of the options on the PROC TCALIS statement in the section "PROC TCALIS Statement" on page 6713. However, you can skip the detailed listing of the available options on the PROC TCALIS statement. Most of these details serve as references, so they are consulted only when needed. You can just read the summary tables for the available options on the PROC TCALIS statement in the following subsections:

- "Data Set Options" on page 6713
- "Model and Estimation Options" on page 6714
- "Options for Fit Statistics" on page 6714
- "Options for Statistical Analysis" on page 6715

- "Global Display Options" on page 6716
- "Optimization Options" on page 6717

Details About Various Types of Models

Several subsections in the section "Details: TCALIS Procedure" on page 6811 will help you gain a deeper understanding of the various types of modeling languages:

Language	**Section**
FACTOR:	"The FACTOR Model" on page 6855
LINEQS:	"The LINEQS Model" on page 6831
LISMOD:	"The LISMOD Model and Submodels" on page 6839
MSTRUCT:	"The MSTRUCT Model" on page 6869
PATH:	"The PATH Model" on page 6862
RAM:	"The RAM Model" on page 6847

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.

Guide to the Intermediate Skills 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: TCALIS Procedure" section or elsewhere can help:

- "Naming Variables and Parameters" on page 6872

  This section summarizes the naming rules and conventions for variable and parameter names in specifying models.

- "Setting Constraints on Parameters" on page 6874

  This section covers various techniques of constraining parameters in model specifications.

- "Automatic Variable Selection" on page 6879

  This section discusses how PROC TCALIS treats variables in the models and variables in the data sets. It also discusses situations where the specification of the VAR statement specification is deemed necessary.

- "Computational Problems" on page 6924

  This section discusses computational problems that occur quite commonly in structural equation modeling. It also discusses some possible remedies of the computational problem.

- "Missing Values" on page 6911

  This small section describes the default treatment of missing values.

- REFMODEL statement on page 6803 and RENAMEPARM statement on page 6804

  These statements are useful when you need to make references to well-defined models when specifying a "new" model. See Example 88.10 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.

Guide to the Advanced Skills Level

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

Advanced Data Analysis Tools

The following advanced data analysis topics are discussed:

- Assessment of fit

  The section "Assessment of Fit" on page 6892 presents the fit indices used in PROC TCALIS. 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.

- Effect partitioning

  The section "Total, Direct, and Indirect Effects" on page 6905 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 6889 describes how PROC TCALIS computes model fit degrees of freedom and how you can use some options on the PROC TCALIS 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 TCALIS statement.

- Standardized solutions

  Standardization schemes used in PROC TCALIS are described and discussed in the section "Standardized Solutions" on page 6907.

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

- Model modifications

  In the section "Modification Indices" on page 6909, 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 SIMTEST 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 6684 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 TCALIS

  You can control output displays in PROC TCALIS 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 6716 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 TCALIS

  Many individual output displays in PROC TCALIS 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 TCALIS, 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 6931 for these names in PROC TCALIS.

  The default output destination of PROC TCALIS 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: TCALIS Procedure" section are intended primarily for references—you consult them only when you encounter specific problems in the PROC TCALIS modeling or when you need to know the very fine technical details in certain special situations. Many of these reference topics in the "Details: TCALIS 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 6911.

- Estimation criteria and the mathematical functions for estimation

  The section "Estimation Criteria" on page 6880 presents formulas for various estimation criteria. The relationships among these criteria are shown in the section "Relationships among Estimation Criteria" on page 6885. 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 6886, where you can also find information about the computation of the standard error estimates in PROC TCALIS.

- Initial estimation

  Initial estimates are necessary for all kinds of iterative optimization techniques. They are described in section "Initial Estimates" on page 6914.

- Use of optimization techniques

  Optimization techniques are covered in section "Use of Optimization Techniques" on page 6915. See this section if you need to fine-tune the optimization.

- Output displays and control

  The output displays in PROC TCALIS are listed in the section "Displayed Output" on page 6927. 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 6931 for the ODS table and graph names.

- Input and output files

  PROC TCALIS 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 6811 and "Output Data Sets" on page 6815.

Changes and Enhancement from PROC CALIS

PROC TCALIS was modified from PROC CALIS. PROC TCALIS is not a simple functional enhancement of PROC CALIS. The basic computational architecture of PROC TCALIS is quite different from that of PROC CALIS. Hence, some discrepancies between the two procedures are less apparent. There are also some temporary discrepancies due to issues that are still being resolved.

In this section, new functionalities in PROC TCALIS are described, followed by some lists for comparing the options and statements available in PROC CALIS and PROC TCALIS. Finally, ODS table names in PROC CALIS are listed with their corresponding new names in PROC TCALIS.

New Features

There are several notably new features in PROC TCALIS:

- new modeling languages
- multiple-group analysis
- improved mean structures analysis
- general parametric functions testing
- customizable fit summary table
- improved standardized results
- improved effects analysis
- customizable Lagrange multiplier tests

- more rotation options in exploratory factor analysis
- input order respecting
- improved OUTRAM= data set format

These new functionalities are outlined in the following sections.

New Modeling Languages in PROC TCALIS

To accommodate various modeling backgrounds and philosophies of researchers, more modeling languages are supported in PROC TCALIS. Three new modeling languages are provided: LISMOD, MSTRUCT, and PATH.

The LISMOD modeling language is a matrix-based parameter specification method modified from the LISREL model developed by Jöreskog and Sörbom. You can specify parameters as matrix entries by using the LISMOD language. For details, see the LISMOD statement on page 6764.

The MSTRUCT modeling language is also a matrix-based parameter specification method. You can specify parameters directly in the structured mean and covariance matrices in this language. For details, see the MSTRUCT statement on page 6784.

The PATH modeling language provides a tool to specify causal relations among variables by using paths (represented by arrows). It is especially suitable for path analysis, although it can also be applied to general structural models. For details, see the PATH statement on page 6791.

Although the FACTOR and RAM modeling languages are not new, their syntax has been changed for easier specifications. The FACTOR and RAM modeling languages are both matrix-based specification methods in PROC CALIS. When you specify parameters in your model, you must use row and column numbers of the model matrices to refer to the variables involved. However, in PROC TCALIS these matrix-based languages are replaced by the more intuitive approach. For confirmatory factor analysis in PROC TCALIS, you can specify the factors to variables paths (or loadings) by using the factor and variable names directly in the FACTOR statement. For the RAM specification in PROC TCALIS, you no longer need to implicitly assume a certain order for the variables and factors in matrices, as you do in PROC CALIS. Also, you no longer need to use the matrix numbers to refer to the types of parameter specifications. In each parameter specification of the RAM statement in PROC TCALIS, the variables and the parameter type involved are specified directly by using the variable names or meaningful keywords for parameter types.

To provide a quick and easy way to specify similar models, the REFMODEL statement is provided in PROC TCALIS. Using this statement, you can specify a new model by referring to another well-defined model. Supporting options PARM_PREFIX= and PARM_SUFFIX= and the RENAMEPARM statement enable you to change parameter names efficiently.

Multiple-Group Analysis

You can do multiple-group analysis in PROC TCALIS. Groups can also be fitted by multiple models simultaneously. You can use multiple GROUP statements to define independent groups of data.

Within the scope of each GROUP statement, you can set group-specific attributes and options for the associated group. See the GROUP statement on page 6756 for details.

You can use multiple MODEL statements to define models and the groups they fit. Within the scope of each MODEL statement, you specify your model by using one of the modeling languages provided by PROC TCALIS. You can use different modeling languages for different models. You can also set model-specific analysis and options for the model within the scope of a MODEL statement. See the MODEL statement on page 6781 for details.

Improved Mean Structures Analysis

In PROC CALIS, the mean structures are analyzed by means of augmented uncorrected moment matrices. This approach is an viable option only for maximum likelihood estimation. Often, this approach creates some interpretation problems in standardized results, R-square calculations, and so on. It is also difficult to set the mean parameters by using this approach.

In PROC TCALIS, the mean structures are analyzed directly as a term in the objective function being optimized. This method is applicable to all estimation methods and yields more interpretable results. You can use the MEANSTR option in the PROC TCALIS or MODEL statements to specify the analysis of mean structures explicitly. Alternatively, when you specify Intercept terms in LINEQS models, parameters in the MEAN statements, or parameters in the intercept or mean vector in the MATRIX statements, the mean structures of the model will be analyzed automatically.

As a result of the improved mean structures analysis, AUG, NOINT, UCOV, and UCORR options are obsolete in PROC TCALIS.

General Parametric Function Testing

You can test any differentiable parametric functions separately or simultaneously by using the TESTFUNC and the SIMTEST statements. A parametric function can be either a parameter in the model or a computed function defined by the SAS Programming statements. PROC TCALIS will analytically generate the necessary partial derivatives for computing the test statistics.

Customizable Fit Summary Table

In PROC TCALIS, you can customize the display of the fit summary table by selecting a subset of the fit indices to display. See the FITINDEX statement on page 6752 for details. You can also choose a particular type of chi-square correction for model fit chi-square statistics. A new OUTFIT= option enables you to store the fit indices in an external data set.

Improved Standardized Results

Standardized parameter estimates with standard errors are provided by default in PROC TCALIS. You can turn off the printing of standard error estimates by the NOSE option. To suppress the printing of the entire standardized results, you can use the NOSTAND option.

The standardized results in PROC TCALIS are somewhat different from that of PROC CALIS. In particular, in PROC TCALIS path coefficients attached to error terms will remain equal to 1 after standardization. The error variances are rescaled appropriately so as to maintain mathematical consistency. In contrast, after standardization PROC CALIS will make all error variances equal to 1 and the path coefficients attached to error terms will not be 1 in general. For interpretation, this is not desirable because error terms, by nature, should be a non-deterministic term added without modification (that is, multiplied by a path coefficient) to the deterministic terms in an equation. In this sense, the standardized method in PROC TCALIS is more interpretable.

Improved Effect Analysis

There are several improvements regarding the effects partitioning in PROC TCALIS. First, standardized effects are displayed in addition to unstandardized effects. Second, standard error estimates are provided for the standardized and unstandardized effects. Third, you can customize the effects analysis by using the EFFPART statement. This will enable you to display only those effects of interest. See the EFFPART statement on page 6743 statement for details.

Customizable Lagrange Multiplier Tests

In PROC TCALIS, you can set your own regions of the parameter space for the Lagrange multiplier (LM) tests. In the LMTESTS statement, you define sets of parameter regions. In each set, you include the regions of interest. In the output, LM statistics ranked within sets are displayed. The parameter that improves the model fit the most appears first. You can also set other display options in the LMTESTS statement.

More Rotation Options in Exploratory Factor Analysis

PROC TCALIS provides more orthogonal and oblique rotation options for exploratory factor analytic solutions. See the ROTATE= option in the FACTOR statement on page 6744 for details.

Input Order Respecting

When you use the LINEQS statement, PROC TCALIS will display equations in the order you specify in the input. The terms within each equation are also ordered the same way you specify them. Unfortunately, PROC CALIS does not have these properties. PROC CALIS might display equations and terms in a certain order that is not consistent with the input.

PROC TCALIS also respects the order of parameter specification in the following statements:

- COV
- MEAN
- PATH

- PCOV
- PVAR
- RAM
- STD

If you want to order the specification by parameter types, you can use the ORDERSPEC option.

PROC TCALIS also respects order when displaying model and group results. By default, the output results for models or groups follow the order of your input. By using the ORDERMODELS and ORDERGROUPS options, the output results for models or groups are ordered by the model or group numbers provided in the specification. The ORDERALL option combines all these ordering options.

Improved OUTRAM= Data Set Format

The OUTRAM= data sets in PROC CALIS stores the model specifications in terms of the RAM model matrix entries, even if the original model is specified by the LINEQS or FACTOR modeling language. The problem is that the modeler who did not write the original code in the RAM modeling language might not understand the contents of the OUTRAM= data set. This inconsistency is eliminated in PROC TCALIS by means of the new OUTMODEL= option (although you can still use the OUTRAM= option for the same purpose). In the OUTMODEL= data sets, different types of models would have different types of observations. The types of observations resemble closely the original modeling language used. See the OUTMODEL= option and the section “OUTMODEL= SAS-data-set” on page 6819 for more details.

Inactive Statements and Options

Inactive statements and options are those available in PROC CALIS, but not available in PROC TCALIS.

The COSAN statement is currently inactive in PROC TCALIS. COSAN model specification might be available in future releases of PROC TCALIS.

The following options are currently inactive in PROC TCALIS, but might be added back in future releases:

Option	**Description**	**TCALIS Treatment**
FDCODE	uses the finite difference method for partial derivatives	uses analytic differentiations by default
HESSALG	specifies the algorithm for computing the Hessian	
NODIAG	suppresses the fitting of the diagonal elements in the covariance or correlation matrices	
OUTJAC	outputs the Jacobian pattern in an external file	
PJACPAT	prints the Jacobian pattern	

The following options are inactive in PROC TCALIS, and there is no plan to put these options back:

Option	Description	TCALIS Treatment
AUGMENT	uses the augmented moment matrices with mean structure analysis	specifies the mean structures in models directly
NOINT	uses the uncorrected covariance or correlation matrices for analyses	eliminates this option
PREDET	displays the predetermined elements in moment matrices	eliminates this display
PVEC	displays the final estimates, std errors, t-values, and gradients in separate vector forms	uses the PRINT or PALL option to display these results in a single table
UCORR	analyzes the uncorrected correlation matrix	eliminates this option
UCOV	analyzes the uncorrected covariance matrix	eliminates this option

Note that even though the UCORR and UCOV options are not supported in PROC TCALIS, you can still provide SAS data sets with TYPE=UCORR or TYPE=UCOV in the DATA= option. PROC TCALIS will utilize the information provided in the data set to compute the corrected covariance or correlation matrix for analysis.

New Statements and Options

The following are new statements in the TCALIS procedure:

Statement	Description
DETERM on page 6742	sets groups of variables for computing the determination coefficients
EFFPART on page 6721	displays and partitions the effects in the model
FITINDEX on page 6752	controls the fit summary output
GROUP on page 6756	defines groups and controls the corresponding data processing
LISMOD on page 6764	specifies a LISREL model
LMTESTS on page 6768	defines the Lagrange multiplier test regions
MEAN on page 6780	specifies the mean parameters
MODEL on page 6781	defines models and controls the corresponding modeling and analysis options
MSTRUCT on page 6784	specifies the direct covariance and mean model
OUTFILES on page 6788	controls multiple output data sets
PATH on page 6791	specifies the PATH model
PCOV on page 6794	specifies the covariance and partial covariance parameters
PVAR on page 6795	specifies the variance and partial variance parameters
REFMODEL on page 6803	specifies the base model for references
RENAMEPARM on page 6804	renames parameters
SIMTEST on page 6806	defines simultaneous parametric function tests
TESTFUNC on page 6809	tests individual parametric functions

The following are new options available in the PROC TCALIS statement:

Option	Description
CHICORRECT= on page 6719	specifies the chi-square correction factor
INMODEL= on page 6722	specifies the data set that contains the input model information
INWGTINV on page 6723	specifies that the INWGT= data set contains the inverse of the weight matrix
MEANSTR on page 6725	analyzes the mean structures
NOORDERSPEC on page 6727	displays the parameter specifications and results according to the input order
NOPARMNAME on page 6727	suppresses the display of parameter names
NOSTAND on page 6727	suppresses the display of standardized results
ORDERALL on page 6730	displays the specifications and results by the group numbers, model numbers, and parameter types
ORDERGROUPS on page 6730	displays the specifications and results for groups by the group numbers
ORDERMODELS on page 6730	displays the specifications and results for models by the model numbers
ORDERSPEC on page 6730	displays the specifications and results for parameters by the parameter types
OUTFIT= on page 6731	specifies the output data set for storing the fit statistics
OUTMODEL= on page 6731	specifies the output data set for storing the model specification and results
PARMNAME on page 6732	displays the parameter names in output

Changes in ODS Table Names

PROC CALIS assigns an ODS table name to each output it creates. Not all of these ODS table names are preserved in PROC TCALIS. In the following table, ODS table names for the CALIS procedure are listed together with the corresponding new ODS table name(s) in PROC TCALIS. Some of these ODS table names have been changed in PROC TCALIS, while others are unchanged or deleted.

CALIS ODS Table Name	TCALIS Table Name(s)
AddParms	(unchanged)
AsymStdRes	(unchanged)
AveAsymStdRes	(unchanged)
AveNormRes	(unchanged)
AveRawRes	(unchanged)
AveVarStdRes	(unchanged)
ContKurtosis	(unchanged)
ConvergenceStatus	(unchanged)
CorrExog	LINEQSCovExogStd
CorrParm	(no longer available)
CovMat	(no longer available)

CALIS ODS Table Name	TCALIS Table Name(s)
DependParms	DependParmsStart or DependParmsResults
Determination	(unchanged)
DistAsymStdRes	(unchanged)
DistNormRes	(unchanged)
DistVarStdRes	(unchanged)
EndogenousVar	LINEQSVariables or RAMVariables
EstCovExog	LINEQSCovExog
Estimates	(no longer available)
EstLatentEq	LINEQSEq
EstManifestEq	LINEQSEq
EstParms	LINEQSBeta, LINEQSBetaStd, LINEQSGamma, LINEQSGammaStd, LINEQSPhi, LINEQSPhiStd, FACTCov, FACTErrVar, FACTErrVarStd, FACTLoadings, FACTRotCov, FACTRotErrVar, or FACTRotErrVarStd
EstVarExog	LINEQSVarExog
ExogenousVar	LINEQSVariables
FactCorrExog	FACTCovStd or FACTRotCovStd
FactScoresCoef	FACTScoresRegCoef
Fit	(unchanged)
GenModInfo	(no longer available)
Gradient	ParameterEstimatesResults
InCorr	(unchanged)
InCorrDet	(unchanged)
InCov	(unchanged)
InCovDet	(unchanged)
InCovExog	LINEQSCovExogInit
Indirect Effects	(unchanged)
Information	(unchanged)
InitEstimates	(no longer available)
InitParms	LINEQSBetaInit, LINEQSGammaInit, LINEQSPhiInit, FACTCovInit, FACTErrVarInit, or FACTLoadingsInit
InitRAMEstimates	(no longer available)
InLatentEq	LINEQSEqInit
InManifestEq	LINEQSEqInit
InSymmetric	(no longer available)
InVarExog	LINEQSVarExogInit
IterHist	(unchanged)
IterStart	(unchanged)
IterStop	(unchanged)
Jacobian	(no longer available)
Kurtosis	(unchanged)
LagrangeBoundary	(unchanged)
LagrangeEquality	(unchanged)
LatentScoreCoef	LatentScoresRegCoef
ModelStatement	(no longer available)

CALIS ODS Table Name	**TCALIS Table Name(s)**
ModIndices	(replaced by various ODS tables with the LM prefix and matrix name suffixes)
NormRes	(unchanged)
PredetElements	(no longer available)
PredModel	PredCorr, or PredCov
PredModelDet	PredCorrDet, or PredCovDet
PredMomentLatent	PredCovLatent
PredMomentManLat	PredCovLatMan
ProblemDescription	(unchanged)
RAMCorrExog	RAMListStd
RAMEstimates	RAMList
RAMStdEstimates	RAMListStd
RankAsymStdRes	(unchanged)
RankLagrange	(replaced by various ODS tables with the LMRank prefix and matrix name suffixes)
RankNormRes	(unchanged)
RankRawRes	(unchanged)
RankVarStdRes	(unchanged)
RawRes	(unchanged)
RotatedLoadings	FACTRotLoadings
Rotation	FACTRotMat
SetCovExog	(no longer available)
SimpleStatistics	(unchanged)
SqMultCorr	(unchanged)
Stability	(unchanged)
StdErrs	ParameterEstimatesResults
StdLatentEq	LINEQSEqStd
StdLoadings	FACTLoadingsStd, or FACTRotLoadingsStd
StdManifestEq	LINEQSEqStd
StructEq	(no longer available)
SumSqDif	(no longer available)
TotalEffects	(unchanged)
tValues	ParameterEstimatesResults
VarSelection	(no longer available)
VarStdRes	(unchanged)
WaldTest	(unchanged)
Weights	(unchanged)
WeightsDet	(unchanged)

Getting Started: TCALIS Procedure

A Structural Equation Example

This example from Wheaton et al. (1977) illustrates the basic uses of the TCALIS 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 (1985, 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)

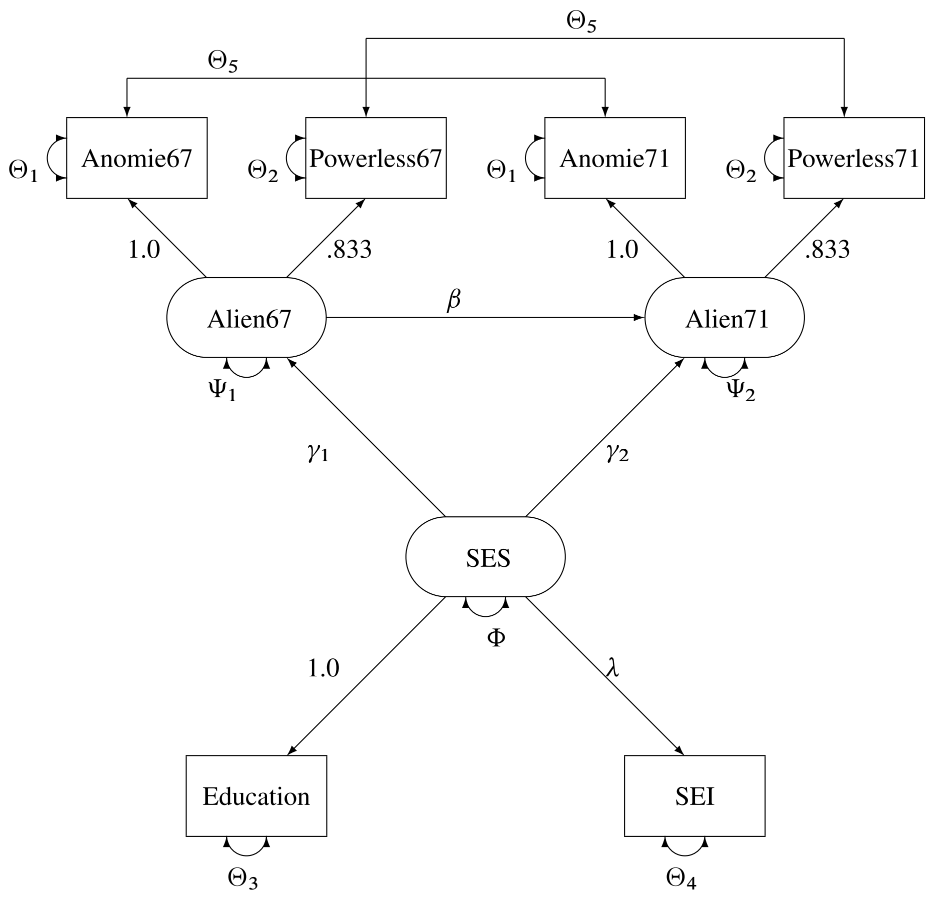
The covariance matrix of the these six variables is stored in the data set named Wheaton.

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 88.2 Path Diagram of Stability and Alienation Example



In the path diagram shown in Figure 88.2, 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 TCALIS procedure.

PATH Model

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

```
proc tcalis 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 TCALIS 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 88.1 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 6862 and the PATH statement on page 6791.

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 TCALIS resembles that of the PATH model, as shown in the following statements:

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

See Example 88.6 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 6847 and the RAM statement on page 6797.

LINEQS Model

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

```
proc tcalis 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;
   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 STD statement, and covariance parameters among exogenous variables, including errors and disturbances, are specified in COV statement.

See Example 88.6 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 6831 and the LINEQS statement on page 6758.

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:

```
proc tcalis 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 6839 and the LISMOD statement on page 6764. See also Example 88.6 for partial results of the current LISMOD model analysis.

A Factor Model Example

In addition to the general modeling languages such as PATH, RAM, LINEQS, and LISMOD, the TCALIS 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 TCALIS:

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

In this model, you want to get the varimax-rotated solution with two factors. By default, the factor extraction method is maximum likelihood (METHOD=ML). Maximum likelihood exploratory factor analysis by PROC TCALIS 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 TCALIS or FACTOR procedure to fit certain exploratory factor models, you can only use the TCALIS 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:

```
proc tcalis 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 6855 or the FACTOR statement on page 6744.

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 TCALIS is designed for this purpose. Consider the following four variables from the Wheaton et al. (1977) data set:

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:

ϕ_1:	Variance of Anomie
ϕ_2:	Variance of Powerlessness
θ_1:	Covariance between Anomie and Powerlessness
θ_2:	Covariance between Anomie measures
θ_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:

```
proc tcalis 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 phi1. 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 88.3.

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 TCALIS, 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 6869 and the MSTRUCT statement on page 6784.

Which Modeling Language?

Various modeling languages are supported in PROC TCALIS 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 TCALIS 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 TCALIS 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 TCALIS, 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.

Syntax: TCALIS Procedure

PROC TCALIS < *options* > ;
  BASEMODEL *model number* < / *options* > ;
  BOUNDS *boundary constraints* ;
  BY *variables* ;
  COV *covariance parameters* ;
  DETERM *variables* < *label* > ;
  EFFPART *effects* ;
  FACTOR < *factor options* > ;
  FITINDEX < *options* > ;
  FREQ *variable* ;
  GROUP *group number* < / *group options* > ;
  LINCON *linear constraints* ;
  LINEQS *model equations* ;
  LISMOD *variable lists* ;
  LMTESTS < *options* > ;
  MATRIX *matrix-name parameters-in-matrix* ;
  MEAN *mean parameters* ;
  MODEL *model number* < / *model options* > ;
  MSTRUCT *variable list* ;
  NLINCON *nonlinear constraints* ;
  NLOPTIONS *optimization options* ;
  OUTFILES *output files organization* ;
  PARAMETERS *parameters* ;
  PARTIAL *variables* ;
  PATH *path list* ;
  PCOV *partial covariance parameters* ;
  PVAR *partial variance parameters* ;
  RAM *ram list* ;
  REFMODEL *model number* < / *options* > ;
  RENAMEPARM *parameter renaming* ;
  SIMTEST *simultaneous tests definitions* ;
  STD *variance parameters* ;
  STRUCTEQ *set of variables* < *label* > ;
  TESTFUNC *parametric functions* ;
  VAR *variables* ;
  WEIGHT *variable* ;
  SAS Programming statements ;

Classes of Statements

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

PROC TCALIS Statement

is the main statement that invokes the TCALIS 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 TCALIS 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 TCALIS 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 TCALIS statement, but the options specified in the GROUP statement supersede those specified in the MODEL or PROC TCALIS statement for the group designated in the GROUP statement. For group options that are available in both of the GROUP and PROC TCALIS statements, see the section "Options Available in the GROUP and PROC TCALIS Statements" on page 6757. For group options that are available in the GROUP, MODEL, and PROC TCALIS statements, see the section "Options Available in GROUP, MODEL, and PROC TCALIS Statements" on page 6757. 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 TCALIS 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:

Statement	Description
FREQ on page 6755	specifies the frequency variable for the input observations
PARTIAL on page 6790	specifies the partial variables
VAR on page 6810	specifies the set of variables in analysis
WEIGHT on page 6810	specifies the weight variable for the input observations

These statements can be used after the PROC TCALIS statement or each GROUP statement. Again, the specifications within the scope of the GROUP statement supersede those specified after the PROC TCALIS 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 TCALIS statement. These options are called model options. Model options specified in the MODEL statement supersede those specified in the PROC TCALIS statement. For model options that are available in both of the MODEL and PROC TCALIS statements, see the section "Options Available in the MODEL and PROC TCALIS Statements" on page 6783. If no MODEL statement is used, a single model is assumed and the model options are specified in the PROC TCALIS 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 TCALIS Statements" on page 6757.

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:

Statement	**Description**
FACTOR on page 6744	specifies confirmatory or exploratory factor models
LINEQS on page 6758	specifies models by using linear equations
LISMOD on page 6764	specifies models in terms of LISREL-like model matrices
MSTRUCT on page 6784	specifies parameters directly in the mean and covariance matrices
PATH on page 6791	specifies models by using the causal paths of variables
RAM on page 6797	specifies models by using RAM-like lists of parameters
REFMODEL on page 6803	specifies a base model from which the target model is modified

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:

Statement	Specification	Modeling Languages
COV on page 6739	covariance parameters	FACTOR, LINEQS
MATRIX on page 6776	parameters in matrices	LISMOD, MSTRUCT
MEAN on page 6780	mean or intercept parameters	FACTOR, LINEQS, PATH
PCOV on page 6794	(partial) covariance parameters	PATH
PVAR on page 6795	(partial) variance parameters	FACTOR, PATH
RENAMEPARM on page 6804	new parameters by renaming	REFMODEL
STD on page 6807	variance parameters	LINEQS

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.

Model Analysis Statements

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

Statement	Analysis
DETERM on page 6742	sets variable groups for computing the determination coefficients; same as the STRUCTEQ statement
EFFPART on page 6721	displays and partitions the effects in the model
FITINDEX on page 6752	controls the fit summary output
LMTESTS on page 6768	defines the Lagrange multiplier test regions
SIMTEST on page 6806	defines simultaneous parametric function tests
STRUCTEQ on page 6742	sets variable groups for computing the determination coefficients; same as the DETERM statement
TESTFUNC on page 6809	tests individual parametric functions

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:

Statement	Description
BOUNDS on page 6738	defines the bounds of parameters
LINCON on page 6758	defines the linear constraints of parameters
NLINCON on page 6786	defines the nonlinear constraints of parameters
NLOPTIONS on page 6787	sets the optimization techniques and printing options

Other Statements

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

Statement	**Description**
BY on page 6738	fits a model to different groups separately
OUTFILES on page 6788	controls multiple output data sets
PARAMETERS on page 6790	defines additional parameters or superparameters
SAS programming statements on page 6806	define parameters or functions

Note that SAS programming statements include the ARRAY statement and the mathematical statements for defining parameter interdependence.

Single-Group Analysis Syntax

```
PROC TCALIS < 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 TCALIS statement.

Multiple-Group Multiple-Model Analysis Syntax

PROC TCALIS < *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 TCALIS 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, SIMTEST, and TESTFUNC. Because these statements are not model-specific, you can put these statements anywhere in a PROC TCALIS 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 TCALIS 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 TCALIS Statement

PROC TCALIS < *options* > **;**

This statement invokes the procedure. There are many options in the PROC TCALIS 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:

Option	Description
DATA=	inputs the data
INEST=	inputs the initial values and constraints
INMODEL=	inputs the model specifications
INWGT=	inputs the weight matrix
OUTEST=	outputs the estimates and their covariance matrix
OUTFIT=	outputs the fit indices
OUTMODEL=	outputs the model specifications
OUTSTAT=	outputs the statistical results
OUTWGT=	outputs the weight matrix

Model and Estimation Options

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

Option	Description
CORRELATION	analyzes correlation matrix
COVARIANCE	analyzes covariance matrix
DEMPHAS=	emphasizes the diagonal entries
EDF=	defines number of observations by the number of error degrees of freedom
INWGTINV	specifies that the INWGT= data set contains the inverse of the weight matrix
MEANSTR	analyzes the mean structures
METHOD=	specifies the estimation method
NOBS=	defines the number of observations
RANDOM=	specifies the seed for randomly generated initial values
RDF=	defines nobs by the number of regression df
RIDGE=	specifies the ridge factor for the covariance matrix
START=	specifies a constant for initial values
VARDEF=	specifies the variance divisor
WPENALTY=	specifies the penalty weight to fit correlations
WRIDGE=	specifies the ridge factor for the weight matrix

Options for Fit Statistics

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

Option	Description
ALPHAECV=	specifies the α level for computing the confidence interval of ECV (Browne and Cudeck 1993)
ALPHARMS=	specifies the α level for computing the confidence interval of RMSEA (Steiger and Lind 1980)
CHICORRECT=	specifies the chi-square correction factor
CLOSEFIT=	defines the close fit value
DFREDUCE=	reduces the degrees of freedom for model fit chi-square test
NOADJDF	requests no degrees-of-freedom adjustment be made for active constraints
NOINDEXTYPE	suppresses the printing of fit index types
OUTFIT=	specifies the output data set for storing fit indices

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:

Option	Description
ASYCOV=	specifies the formula for computing asymptotic covariances
BIASKUR	computes the skewness and kurtosis without bias corrections
EFFPART | TOTEFF	displays total, direct, and indirect effects
G4=	specifies the algorithm for computing standard errors
KURTOSIS	computes and displays kurtosis
MODIFICATION	computes modification indices
NOMOD	suppresses modification indices
NOSTAND	suppresses the standardized output
NOSTDERR	suppresses standard error computations
PCORR	displays analyzed and estimated moment matrix
PCOVES	displays the covariance matrix of estimates
PDETERM	computes the determination coefficients
PESTIM	prints parameter estimates
PINITIAL	prints initial pattern and values
PLATCOV	computes the latent variable covariances and score coefficients
PLOTS=	specifies ODS Graphics selection
PWEIGHT	displays the weight matrix
RESIDUAL=	specifies the type of residuals being computed
SIMPLE	prints univariate statistics
SLMW=	specifies the probability limit for Wald tests
STDERR	computes the standard errors

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:

Option	Description
NOPRINT	suppresses the displayed output
PALL	displays all displayed output (ALL)
PRINT	adds default displayed output
PSHORT	reduces default output (SHORT)
PSUMMARY	displays fit summary only (SUMMARY)

In contrast to individual output printing options described in the section "Options for Statistical Analysis" on page 6715, 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:

Options	PALL	PRINT	default	PSHORT	PSUMMARY
fit indices	*	*	*	*	*
linear dependencies	*	*	*	*	*
PESTIM	*	*	*	*	
iteration history	*	*	*	*	
PINITIAL	*	*	*		
SIMPLE	*	*	*		
STDERR	*	*	*		
RESIDUAL	*	*			
KURTOSIS	*	*			
PLATCOV	*	*			
TOTEFF	*	*			
PCORR	*				
MODIFICATION	*				
PWEIGHT	*				
PCOVES					
PDETERM					
PRIMAT					

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:

Option	Description
NOORDERSPEC	displays model specifications and results according to the input order
NOPARMNAME	suppresses the printing of parameter names in results
ORDERALL	orders all output displays according to the model numbers, group numbers, and parameter types
ORDERGROUPS	orders the group output displays according to the group numbers
ORDERMODELS	orders the model output displays according to the model numbers
ORDERSPEC	orders the model output displays according to the parameter types within each model
PARMNAME	displays parameter names in model specifications and results
PRIMAT	displays estimation results in matrix form

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.

Option	Description
ASINGULAR=	specifies the absolute singularity criterion for inverting the information matrix
COVSING=	specifies the singularity tolerance of the information matrix
FCONV=	specifies the relative function convergence criterion
GCONV=	specifies the gradient convergence criterion
INSTEP=	specifies the initial step length (RADIUS=, SALPHA=)
LINESEARCH=	specifies the line-search method
LSPRECISION=	specifies the line-search precision (SPRECISION=)
MAXFUNC=	specifies the maximum number of function calls
MAXITER=	specifies the maximum number of iterations
MSINGULAR=	specifies the relative M singularity of the information matrix
OMETHOD | TECHNIQUE=	specifies the minimization method
SINGULAR=	specifies the singularity criterion for matrix inversion
UPDATE=	specifies the update method for some optimization techniques
VSINGULAR=	specifies the relative V singularity of information matrix

Listing of PROC TCALIS Statement Options

ALPHAECV=α

specifies a $(1-\alpha)100\%$ confidence interval $(0 \le \alpha \le 1)$ for the Browne and Cudeck (1993) expected cross-validation index (ECVI). The default value is $\alpha = 0.1$, which corresponds to a 90% confidence interval for the ECVI.

ALPHARMS=α

specifies a $(1-\alpha)100\%$ confidence interval $(0 \le \alpha \le 1)$ 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 | ASING=*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:

$$|d_{j,j}| \le \max(ASING, VSING * |H_{j,j}|, MSING * \max(|H_{1,1}|, \ldots, |H_{n,n}|))$$

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 $\mathbf{D}^{-1}\mathbf{H}\mathbf{D}^{-1}$ is decomposed (where $D^2 = diag(H)$), and the singularity criteria are modified correspondingly.

ASYCOV | ASC=*name*

specifies the formula for asymptotic covariances used in the weight matrix $\mathbf{W}$ 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's (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's (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's (1986) formula (3.2) (identical to DeLeeuw's (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.

CAUTION: Using the WLS and DWLS methods with the ASYCOV=CORR option assumes that you are fitting a correlation (rather than a covariance) structure. Because the fixed diagonal of a correlation matrix for some models does not contribute to the model's degrees of freedom, you can specify the DFREDUCE=*i* option to reduce the degrees of freedom by the number of manifest variables used in the model.

See the section "Counting the Degrees of Freedom" on page 6889 for more information.

BIASKUR

computes univariate skewness and kurtosis by formulas uncorrected for bias.

See the section "Measures of Multivariate Kurtosis" on page 6911 for more information.

CHICORRECT | 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.

Valid *names* for the CHICORRECT= value are as follows:

EQVARCOV | COMPSYM 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 represents the number of variables and $n = (N-1)$, with N denoting the number of observations in a single group analysis.

CIRCULARITY | CIRCULAR | TYPEH | SPHERICITY specifies the correction factor due to Mauchly (1940) for testing circularity or Huynh and Feldt Type H matrix. The correction factor is:

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

where p represents the number of variables and $n = (N-1)$, with N denoting the number of observations in a single group analysis.

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

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

where p represents the number of variables, k represents the number of groups, and $n_i = (N_i - 1)$, with N_i denoting the number of observations in the i-th group.

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 TCALIS, 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 TCALIS 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.

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 6811. If the DATA= option is omitted, the most recently created SAS data set is used.

DEMPHAS | 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 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 | DFRED=*i*

reduces the degrees of freedom of the model fit χ^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. If the DFREDUCE= option is used in a correlation structure analysis, PROC TCALIS does not additionally reduce the degrees of freedom by the number of constant elements in the diagonal of the predicted model matrix, which is otherwise done automatically. See the section "Counting the Degrees of Freedom" on page 6889 for more information.

EDF | DFE=*n*

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

EFFPART | PARTEFF | 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 TCALIS computes and displays the "stability coefficient of reciprocal causation"— that is, the largest modulus of the eigenvalues of the $\boldsymbol{\beta}$ 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 TCALIS does not show results for total effects and their partitioning. See the section "Stability Coefficient of Reciprocal Causation" on page 6907 for more information about the computation of the stability coefficient.

FCONV | 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,

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

where *FSIZE* can be defined by the FSIZE= option in the NLOPTIONS statement. The default value is $r = 10^{-\mathit{FDIGITS}}$, where *FDIGITS* either can be specified in the NLOPTIONS statement or is set by default to $-\log_{10}(\epsilon)$, where ϵ is the machine precision.

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 6880 for more details.

GCONV | GTOL=*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,

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

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),

$$\frac{\parallel g(x^{(k)}) \parallel_2^2 \quad \parallel s(x^{(k)}) \parallel_2}{\parallel g(x^{(k)}) - g(x^{(k-1)}) \parallel_2 \max(|f(x^{(k)})|, FSIZE)} \leq r$$

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

INEST | INVAR | 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 TCALIS analysis. The initial estimates are taken from the values of the PARMS observation in the INEST= data set.

INMODEL | INRAM=*SAS-data-set*

specifies an input data set that contains in RAM list form all information needed to specify an analysis model. The INMODEL= data set is described in the section "Input Data Sets" on page 6811. Typically, this input data set is an OUTMODEL= data set (possibly modified) from a previous PROC TCALIS analysis. If you use an INMODEL= data set to specify the analysis model, you should not use any of the main or subsidiary model specification statements, but you can use the BOUNDS and PARAMETERS statements and program statements. If the INMODEL= option is omitted, you should define the analysis model with one of the main model specification statements.

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 6924.

INWGT< (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 6880 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 6811. Typically, this input data set is an OUTWGT= data set from a previous PROC TCALIS 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 6911 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*

specifies the line-search method for the CONGRA, QUANEW, and NEWRAP optimization techniques. Refer to 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 $i = 2$.

LIS=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.
LIS=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.
LIS=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.
LIS=4	specifies a line-search method that needs the same number of function and gradient calls for stepwise extrapolation and cubic interpolation.
LIS=5	specifies a line-search method that is a modified version of LIS=4.
LIS=6	specifies golden-section line search (Polak 1971), which uses only function values for linear approximation.
LIS=7	specifies bisection line search (Polak 1971), which uses only function values for linear approximation.
LIS=8	specifies the Armijo line-search technique (Polak 1971), which uses only function values for linear approximation.

LSPRECISION | LSP=*r*
SPRECISION | 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.

OMETHOD=	UPDATE=	LSP default
QUANEW	DBFGS, BFGS	$r = 0.4$
QUANEW	DDFP, DFP	$r = 0.06$
CONGRA	all	$r = 0.1$
NEWRAP	no update	$r = 0.9$

For more details, refer to 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.

OMETHOD=	**MAXFUNC default**
LEVMAR, NEWRAP, NRRIDG, TRUREG	i = 125
DBLDOG, QUANEW	i = 500
CONGRA	i = 1000

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 | MAXIT=*i* < *n* >

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

OMETHOD=	**MAXITER default**
LEVMAR, NEWRAP, NRRIDG, TRUREG	i = 50
DBLDOG, QUANEW	i = 200
CONGRA	i = 400

The default is used if you specify MAXITER=0 or if you omit the MAXITER option.

The optional second value n is valid only for 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

```
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.

MEANSTR

invokes the analysis of mean structures. By default, no mean structures are analyzed. You can also analyze the mean structures by specifying any mean or intercept parameters in your model without using the MEANSTR option.

METHOD | MET | M=*name*

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

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.
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 χ^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 6909 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 | 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:

$$|d_{j,j}| \leq \max(ASING, VSING * |H_{j,j}|, MSING * \max(|H_{1,1}|, \ldots, |H_{n,n}|))$$

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 $\mathbf{D}^{-1}\mathbf{H}\mathbf{D}^{-1}$ is decomposed (where $D^2 = diag(H)$), 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 6891 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 TCALIS 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.

NOMOD

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

NOORDERSPEC

prints the model results in the order they appear in the input specifcations. This is the default printing behavior. You can use this option to reset the default at a local model when the ORDERSPEC option might have been set in a statement at a more global level.

NOPARMNAME

suppresses the printing of parameter names in the model results. The default is to print the parameter names.

NOPRINT | 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.

NOSTAND

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

NOSTDERR | NOSE

specifies that standard errors should not be computed. 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).

OMETHOD | OM=*name*

TECHNIQUE | 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 TCALIS 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 400 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 if there are fewer than 40 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. In releases prior to Release 6.11, this option invokes the NRRIDG 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.
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 (1982a, 1982b) VMCWD algorithm 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 μ to the original update of Powell (1978a, 1978b) 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 40 and fewer than 400 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 40 parameters ($t < 40$), OMETHOD=LEVMAR (Levenberg-Marquardt) is the default optimization technique. For $40 \leq t < 400$, OMETHOD=QUANEW (quasi-Newton) is the default method, and for $t \geq 400$, 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 6915 for more details.

UPDATE | 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.
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.

ORDERALL

prints the model and group results in the order of the model or group numbers, starting from the smallest number. It also arrange 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 default 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.

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= SAS-data-set" on page 6815 for a description of the OUTEST= data set. If you want to create a permanent SAS data set, you must specify a two-level name. Refer to the chapter titled "SAS Data Files" in *SAS Language Reference: Concepts* for more information about permanent data sets.

OUTFIT=*SAS-data-set*

creates an output data set that contains the values of the fit indices. See the section "OUTFIT= SAS-data-set" on page 6829 for details.

OUTMODEL | 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 TCALIS. The OUTMODEL= data set also contains a set of fit indices; the section "OUTMODEL= SAS-data-set" on page 6819 provides more details. If you want to create a permanent SAS data set, you must specify a two-level name.

Refer to the chapter titled "SAS Data Files" in *SAS Language Reference: Concepts* for more information about permanent data sets.

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 TCALIS. See the section "OUTSTAT= SAS-data-set" on page 6824 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 6733). 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 permanent SAS data set, you must specify a two-level name.

Refer to the chapter titled "SAS Data Files" in *SAS Language Reference: Concepts* for more information about permanent data sets.

OUTWGT=*SAS-data-set*

creates an output data set that contains the elements of the weight matrix $\mathbf{W}$ or the its inverse $\mathbf{W}^{-1}$ 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 TCALIS. See the section "OUTWGT= SAS-data-set" on page 6828 for the description of the OUTWGT= data set. If you want to create a permanent SAS data set, you must specify a two-level name.

Refer to the chapter titled "SAS Data Files" in *SAS Language Reference: Concepts* for more information about permanent data sets.

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 behavior. You can use this option to reset the default at a local MODEL statement when the NOPARMNAME option might have been set in a statement at a more global level.

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 TCALIS 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 TCALIS, 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 TCALIS 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.

PINITIAL | PIN

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

PLATCOV | PLC

displays the following:

- the estimates of the covariances among the latent variables
- the estimates of the covariances between latent and manifest variables
- 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.

PLOTS | PLOT < = *plot-request* >

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

specifies the ODS graphical plots. Currently, the only available ODS graphical plots in PROC TCALIS are for residual histograms. Also, when the residual histograms are requested, the bar charts of residual tallies are suppressed. To display these bar charts with the residual histograms, you must use the RESIDUAL(TALLY) option.

When you specify only one *plot-request*, you can omit the parentheses around the *plot-request*. For example:

PLOTS=ALL
PLOTS=RESIDUALS

The following table shows the available *plot-requests*:

Plot-request	Plot Description
ALL	all available plots
NONE	no ODS graphical plots
RESIDUALS	distribution of residuals

To request the ODS graphical plots, you must also specify the `ods graphics on` statement. For more information about the ODS GRAPHICS statement, see Chapter 21, "Statistical Graphics Using ODS."

PRIMAT | PMAT
displays parameter estimates, approximate standard errors, and *t* values in matrix form if you specify the analysis model using the RAM or LINEQS statement.

PRINT | PRI
adds the options KURTOSIS, RESIDUAL, PLATCOV, and TOTEFF to the default output.

PSHORT | SHORT | PSH
excludes the output produced by the PINITIAL, SIMPLE, and STDERR options from the default output.

PSUMMARY | SUMMARY | PSUM
displays the fit assessment table only.

PWEIGHT | PW
displays the weight matrix **W** used in the estimation. The weight matrix is displayed after the WRIDGE= and the WPENALTY= options are applied to it. However, if you specify an INWGT= data set by the INWGT= and INWGTINV options (or the INWGT(INV)= option alone) in the same analysis, this option displays the elements of the inverse of the weight matrix.

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 6914 for more information.

RDF | 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 TCALIS 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.

RESIDUAL | 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. This information is displayed by default when you specify the PRINT option.

Three types of normalized or standardized residual matrices can be chosen with the RESIDUAL= specification.

RESIDUAL= NORM	normalized residuals
RESIDUAL= VARSTAND	variance standardized residuals

RESIDUAL= ASYSTAND	asymptotically standardized residuals

When ODS graphical plots of 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.

See the section "Assessment of Fit" on page 6892 for more details.

RIDGE< =*r* >

defines a ridge factor r for the diagonal of the covariance or correlation matrix $\mathbf{S}$ that is analyzed. The matrix $\mathbf{S}$ is transformed to:

$$\mathbf{S} \longrightarrow \tilde{\mathbf{S}} = \mathbf{S} + r(\mathrm{diag}(\mathbf{S}))$$

If you do not specify r in the RIDGE option, PROC TCALIS tries to ridge the covariance or correlation matrix $\mathbf{S}$ 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.

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

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 | 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 | SP=*r*

is an alias for the LSPRECISION= option.

START =*r*

specifies initial estimates for parameters as multiples of the r value. In all TCALIS 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 TCALIS. 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, $10|r|$, or $100|r|$. For variance parameters, $100|r|$ is used for covariance structure analyses and $10|r|$ 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 6914 for more information.

STDERR | SE

displays 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). 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.

VARDEF= DF | N | WDF | WEIGHT | WGT

specifies the divisor used in the calculation of covariances and standard deviations. The default value is VARDEF=DF. 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 jth observation, and the summation is performed only over observations with positive weight.

Value	Description	Divisor
DF	degrees of freedom	$N - k - 1$
N	number of observations	N
WDF	sum of weights DF	$\sum_j^N w_j - k - 1$
WEIGHT | WGT	sum of weights	$\sum_j^N w_j$

VSINGULAR | 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:

$$|d_{j,j}| \le \max(ASING, VSING * |H_{j,j}|, MSING * \max(|H_{1,1}|, \ldots, |H_{n,n}|))$$

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 $\mathbf{D}^{-1}\mathbf{H}\mathbf{D}^{-1}$ is decomposed (where $D^2 = diag(H)$), and the singularity criteria are modified correspondingly.

WPENALTY | WPEN=*r*

specifies the penalty weight $r \ge 0$ 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

$$\mathbf{F}_{WLS} = \sum_{i=2}^{n} \sum_{j=1}^{i-1} \sum_{k=2}^{n} \sum_{l=1}^{k-1} w^{ij,kl}(s_{ij} - c_{ij})(s_{kl} - c_{kl}) + r \sum_{i}^{n} (s_{ii} - c_{ii})^2$$

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**, and it is effective only with the ASYCOV=CORR option. 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 6880 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

$$\mathbf{W} \longrightarrow \tilde{\mathbf{W}} = \mathbf{W} + r(\mathrm{diag}(\mathbf{W}))$$

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).

BASEMODEL Statement

REFMODEL | BASEMODEL *model_number* < / *options* > **;**

where *model_number* represents a model number between 1 and 9999.

See the REFMODEL statement on page 6803 for details.

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 larger 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 TCALIS treat strict inequality constraints < or > as if they were just inequality constraints <= or >=. For example, if you require x be strictly bigger 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 TCALIS 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 χ^2 value, it adds difficulties in interpretation.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC TCALIS to obtain separate analyses on observations in groups defined by the BY variables in the input data set. When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

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 BY statement option NOTSORTED or DESCENDING in the BY statement for the TCALIS 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.

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

For more information about the BY statement, refer to the discussion in *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, refer to the discussion in the *SAS Procedures Guide.*

COV Statement

COV *assignment* < , *assignment* . . . > **;**

where *assignment* represents
variables < $*$ *variables2* > = *parameter-spec*

The COV statement is a subsidiary model specification statement for the FACTOR and LINEQS models. In the LINEQS model, the COV statement defines the covariances among the exogenous variables, including errors and disturbances. In the FACTOR model, the COV statement defines the factor covariances. In each *assignment* of the COV statement, you specify variables in the *variables* and the *variables2* lists, followed by the covariance parameter specification in the *parameter-spec* list. Covariances with only values given are fixed constants. Covariances with names given are free parameters to estimate. You can also specify initial values for these free parameters by putting the parenthesized initial values after the parameter names. The *assignment*s in the COV statement must be separated by commas. You can use only one COV statement in each model specification.

Consider a LINEQS model with exogenous variables V1, V2, V3, and V4. The following is a COV statement specification of the LINEQS model:

```
cov V2 V1 = 0.3,
    V3 V1 = phi1 (0.4),
    V3 V2 = phi2,
    V4 V3 = phi2;
```

In this statement, you specify **`cov(V2,V1)`** as a fixed constant at 0.3, **`cov(V3,V1)`** as parameter phi1 with an initial value at 0.4, **`cov(V3,V2)`** and **`cov(V4,V3)`** as a free parameter phi2 without an initial value.

Note that the *variables* and *variables2* lists on the left-hand side of the equal sign of the COV statement should contain only names of exogenous variables. In LINEQS models, exogenous variables can be either observed or latent (including errors and disturbances). However, in the FACTOR models, only covariances among latent factors can be specified.

If the right-hand-side parameter list is longer than the left-hand-side variable list, the right-hand-side list is shortened to the length of the variable list. If the right-hand-side list is shorter than the variable list, the right-hand-side list is filled with repetitions of the last item in the list.

You can use one of two alternatives to specify covariance parameters. The first alternative uses only one variable list and refers to all distinct pairs of variables within the list. The second alternative uses two variable lists separated by an asterisk and refers to all pairs of variables among the two lists.

Within-List Covariances

Using k variable names in the *variables* list on the left-hand side of an equal sign in a COV statement means that the parameter list (*parameter-spec*) on the right-hand side refers to all $k(k-1)/2$ distinct variable pairs. Order is very important. The order relation between the left-hand-side variable pairs and the right-hand-side parameter list is illustrated by the following within-list covariances specification in the COV statement:

```
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;
```

Because `cov(a,b)` is the same as `cov(b,a)`, you need to specify only one of them for a covariance parameter of any two variables.

When you use prefix names on the right-hand sides, you do not have to count the exact number of parameters. For example, the following statement generates distinct parameter names by appending distinct numbers to the prefix phi:

```
cov E1-E4 = phi__ ;
```

Between-List Covariances

Using k_1 and k_2 variable names in the two lists (separated by an asterisk) on the left-hand side of an equal sign in a COV statement means that the covariance parameter list on the right-hand side refers to all $k_1 \times k_2$ distinct variable pairs. Again, order is very important. The order relation between the left-hand-side variable pairs and the right-hand-side parameter list is illustrated by the following between-list covariances specification:

```
cov E1 E2 * E3 E4 = phi1-phi4;
```

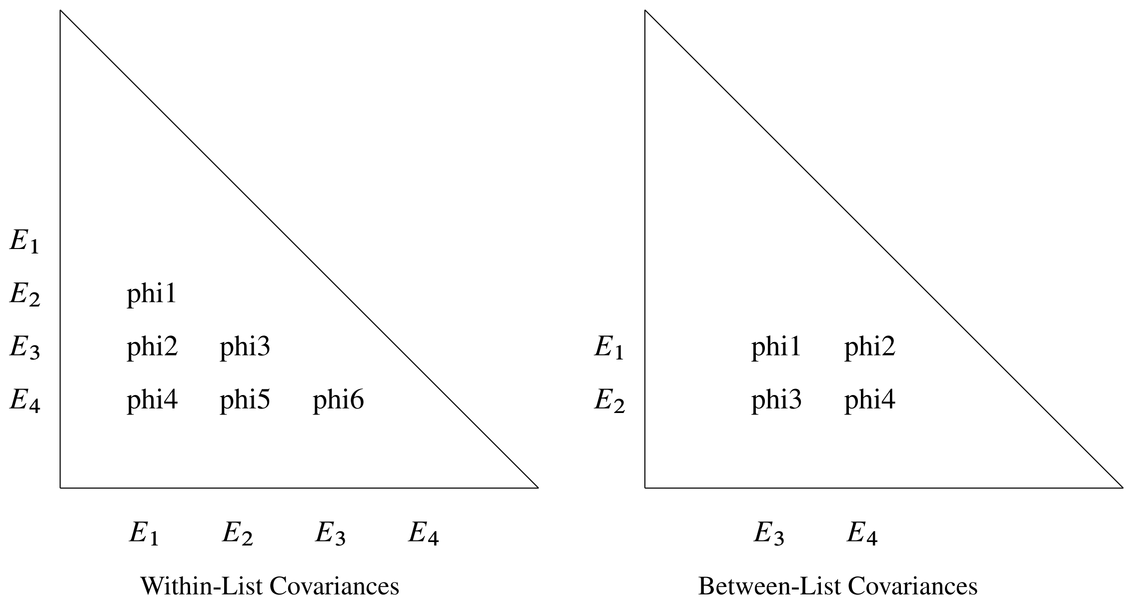
This is equivalent to the following specification:

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

Using prefix names on the right-hand sides lets you achieve the same purpose without counting the number of parameters. That is, you can specify the between-list covariances equivalently by the following statement:

```
cov  E1 E2 * E3 E4 = phi__ ;
```

Figure 88.3 Within-List and Between-List Covariances



Automatic Covariance Parameters among Exogenous Manifest Variables

In LINEQS models, all covariances among exogenous manifest variables are free parameters to estimate unless you specify otherwise in the COV statement. The parameter names for these covariances are generated by PROC TCALIS with the prefix _Add and appended with integer suffixes. If you want any of these covariances be fixed values, you must specify them explicitly in the COV statement.

Emphasis must be put on the **manifest variables** for automatic covariance parameters generation. Covariaces among exogenous latent variables by default are fixed zeros unless they are specified otherwise in the COV statement. The same applies to any covariance between an exogenous latent variable and an exogenous manifest variable. Only among the set of exogenous mainfest variables are the covariance parameters automatically generated if they are not specified explicitly in the COV statement. This is different than the case of variance parameters. Variances of **all** exogenous variables are automatically generated if they are not explicitly specified in the STD statement.

Modifying a Parameter Specification from a Reference Model

If you define a new LINEQS model by using a reference (or base) model in the REFMODEL statement, in some situations you might want to modify some parameter specifications from the COV statement of the reference model before transferring the specifications to the new model. If you want to change a particular COV parameter specification from the reference model, you can simply put the corresponding parameter location with a new parameter specification in the new model. If you want to delete a particular parameter location from the reference model, you can put that parameter location with a missing parameter value in the new model. For example, if the covariance between variables V1 and V2 are defined in the reference model and you do not want this parameter location in your new model, you can use the following statement to delete the parameter location in the new model:

```
cov  V1 V2 = .;
```

Note that the missing value syntax is valid only when you use with the REFMODEL statement. See the section "Modifying a LINEQS Model from a Reference Model" on page 6761 for a more detailed example of LINEQS model respecification.

In LINEQS models, all covariances among exogenous manifest variables are free parameters to estimate unless you specify otherwise in the COV statement. This is also true for LINEQS models specified under the REFMODEL statement. In the reference model, covariance parameters generated by PROC TCALIS are named with the _Add prefix and appended with integer suffixes. These parameter specifications in the 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 model respecifications do the remaining unspecified covariances among exogenous manifest variables generate additional covariance parameters with the _Add prefix and integer suffixes. In this way, the covariance parameters in the new model are not constrained to be the same as the corresponding parameters in the reference model. If you want any of these covariance parameters to be constrained across the models, you must specify them explicitly in the COV statement of the reference model so that the 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:
variables < *direction variables2* >
and *direction* is the direction of the effect, as indicated by one of the following: –>, >, <–, or <.

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 TCALIS 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 endongenous, 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 exploratory factor analysis options, *CFA_spec* is confirmatory factor analysis specification defined as:

factor-variables-relation < , *factor-variables-relation* > . . .

and each *factor-variables-relation* is defined as:

factor –> variables = parameter-spec

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 an *EFA_option* (exploratory factor analysis option) is specified immediately after the FACTOR statement, or neither *EFA_options* nor *factor-variable-relations* are specified in the FACTOR statement, an exploratory factor model is analyzed. Otherwise, specifying an *CFA_spec* (confirmatory factor analysis specification) invokes a confirmatory factor model. These two types of models are discussed in the next two sections.

Exploratory Factor Analysis

For the exploratory factor model with orthogonal factors, PROC TCALIS assumes the following model structures for the population covariance or correlation matrix $\mathbf{C}$:

$$\mathbf{C} = \mathbf{F}\mathbf{F}' + \mathbf{U}$$

where $\mathbf{F}$ is the factor loading matrix and $\mathbf{U}$ 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 $\mathbf{C}$, 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 $\mathbf{F}$. 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 $\tilde{\mathbf{F}} = \mathbf{F}\boldsymbol{\Theta}$ satisfies the same model structures as $\mathbf{F}$ does, where $\boldsymbol{\Theta}$ is any orthogonal matrix so that $\boldsymbol{\Theta}'\boldsymbol{\Theta} = \boldsymbol{\Theta}\boldsymbol{\Theta}' = \mathbf{I}$. Mathematically, the covariance or correlation structures can be expressed as:

$$\mathbf{C} = \tilde{\mathbf{F}}\tilde{\mathbf{F}}' + \mathbf{U} = \mathbf{F}\boldsymbol{\Theta}\boldsymbol{\Theta}'\mathbf{F}' + \mathbf{U} = \mathbf{F}\mathbf{F}' + \mathbf{U}$$

To obtain an identified orthogonal factor solution as a starting point, the $n(n-1)/2$ elements in the upper triangle of $\mathbf{F}$ are constrained to zeros in PROC TCALIS. 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 TCALIS 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:

$$\mathbf{C} = \tilde{\mathbf{F}}\mathbf{P}\tilde{\mathbf{F}}' + \mathbf{U}$$

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 6855 and "Exploratory Factor Analysis Models" on page 6857 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 TCALIS procedures are not attempted here, some general points can be made here. In general, the FACTOR procedure provides more factor analysis options than the TCALIS procedure does, although both procedures have some unique factor analysis features that are not shared by the other. PROC TCALIS 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 TCALIS. Because the initial unrotated factor solution obtained by PROC FACTOR uses a different set of identification constraints than that of PROC TCALIS, you would observe different initial ML factor solutions for the procedures. Nonetheless, the initial solutions by both procedures are statistically equivalent.

The following exploratory factor analysis options are available in the FACTOR statement.

COMPONENT | 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 $\mathbf{F}\mathbf{F}'$ is equal to the number n of components in $\mathbf{F}$. If n is smaller than the number of variables in the moment matrix $\mathbf{C}$, 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.

HEYWOOD | HEY

constrains the diagonal elements of $\mathbf{U}$ to be nonnegative. Equivalently, you can constrain these elements to positive values by the BOUNDS statement.

GAMMA=*p*

specifies the orthomax weight used with the option ROTATE=ORTHOMAX. Alternatively, you can use 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 GAMMA= value is one, resulting in the varimax rotation.

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 $\mathbf{C} = \mathbf{U}$, with $\mathbf{U} = diag$. 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:

$$|f_{new} - f_{old}|/K < \epsilon$$

where f_{new} and f_{old} are simplicity function values of the current cycle and the previous cycle, respectively, $K = max(1, |f_{old}|)$ is a scaling factor, and ϵ 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.

ROTATE | R=*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(*p1*,*p2*) | ORCF(*p1*,*p2*) specifies the orthogonal Crawford-Ferguson rotation (Crawford and Ferguson 1970) with the weights *p1* and *p2* for variable parsimony and factor parsimony, respectively. See the definitions of weights in Chapter 33, "The FACTOR Procedure."

ORTHGENCF(*p1*,*p2*,*p3*,*p4*) | ORGENCF(*p1*,*p2*,*p3*,*p4*) specifies the orthogonal 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 1572 in Chapter 33, "The FACTOR Procedure."

ORTHOMAX< (*p1*) > | ORMAX< (*p1*) > specifies the orthomax rotation (see Harman 1976) with orthomax weight *p1*. If ROTATE=ORTHOMAX is used, the default *p1* value is 1 unless specified otherwise in the GAMMA= option. Alternatively, ROTATE=ORTHOMAX(*p1*) specifies *p1* as the orthomax weight or the GAMMA= value. For the definitions of the orthomax weight, see the section "Simplicity Functions for Rotations" on page 1572 in Chapter 33, "The FACTOR Procedure."

PARSIMAX | PA specifies orthogonal parsimax rotation. This corresponds to the specification ROTATE=ORTHOMAX with

$$\text{GAMMA} = \frac{p \times (n-1)}{p+n-2}$$

PRINCIPAL | PC specifies a principal axis rotation. If ROTATE=PRINCIPAL is used with a factor rather than a component model, the following rotation is performed:

$$\mathbf{F}_{new} = \mathbf{F}_{old}\mathbf{T}, \quad \text{with} \quad \mathbf{F}'_{old}\mathbf{F}_{old} = \mathbf{T}\boldsymbol{\Lambda}\mathbf{T}'$$

where the columns of matrix $\mathbf{T}$ contain the eigenvectors of $\mathbf{F}'_{old}\mathbf{F}_{old}$.

QUARTIMAX | QMAX | Q specifies orthogonal quartimax rotation. This corresponds to the specification ROTATE=ORTHOMAX(0).

VARIMAX | V specifies orthogonal varimax rotation. This corresponds to the specification ROTATE=ORTHOMAX with GAMMA=1.

Valid *names* for oblique rotations are as follows:

BIQUARTIMIN | BIQMIN specifies biquartimin rotation. It corresponds to the specification ROTATE=OBLIMIN(.5) or ROTATE=OBLIMIN with TAU=.5.

COVARIMIN | CVMIN specifies covarimin rotation. It corresponds to the specification 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 1572 in Chapter 33, "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 1572 in Chapter 33, "The FACTOR Procedure."

OBLIMIN< (*p1*) > | OBMIN< (*p1*) > specifies the oblimin rotation with oblimin weight *p1*. If ROTATE=OBLIMIN is used, the default *p1* value is zero unless specified otherwise in the TAU= option. Alternatively, 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 1572 in Chapter 33, "The FACTOR Procedure."

OBPARSIMAX | OPA specifies oblique parsimax rotation.

OBQUARTIMAX | OQMAX specifies oblique quartimax rotation. This is the same as the QUARTIMIN 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-variable-relations* ;
  PVAR *partial variance parameters* ;
  COV *covariance parameters* ;
  MEAN *mean parameters* ;

If you specify at least one *factor-variable-relation*, a confirmatory factor analysis is assumed. Whenever applicable, you can use the PVAR, COV, and MEAN statements to specify the variance, partial variance, covariance, and mean parameters. The model structures for the covariance matrix **C** of the confirmatory factor model are described in the following equation:

$$\mathbf{C} = \mathbf{F}\mathbf{P}\mathbf{F}' + \mathbf{U}$$

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 $\boldsymbol{\mu}$ of the confirmatory factor model are described in the following equation:

$$\boldsymbol{\mu} = \boldsymbol{\alpha} + \mathbf{F}\boldsymbol{\nu}$$

where $\boldsymbol{\alpha}$ is the intercept vector for the observed variables and $\boldsymbol{\nu}$ is the vector for factor means. See the sections “The FACTOR Model” on page 6855 and “Confirmatory Factor-Analysis Models” on page 6858 for more details about confirmatory factor models.

You can use the PVAR statement to specify factor variance parameters in the diagonal of matrix $\mathbf{P}$, or the error variances in the diagonal of matrix $\mathbf{U}$. These variances, by default, are automatically added as free parameters in the factor model even if you do not specify them. However, if you want to set constraints on these parameters, it is useful to define them explicitly by using the PVAR statement. For example, you might like to set some 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.

You can use the COV statement to specify factor covariance parameters. These factor covariances are the off-diagonal elements of matrix $\mathbf{P}$. Unlike the factor variances, factor covariances are assumed to be fixed zeros by default. Note that you cannot use the COV statement to specify the error covariances—they are always fixed zeros in the confirmatory factor-analysis model.

You can use the MEAN statement to specify the means of the factors in vector $\boldsymbol{\nu}$ and the intercepts of the manifest variables in vector $\boldsymbol{\alpha}$. All the factor means and intercepts are fixed zeros if you do not specify them explicitly.

Suppose that you want to fit a factor model with nine manifest variables V1-V9. In this factor model, you assume a general factor and three group-factors, with three variables defining each group-factor. The following FACTOR statement shows the specification of such a model:

```
factor
   g_factor -> V1-V9 (9 * g_load__),
   factor_a -> V1-V3 (3 * load_a__),
   factor_b -> V4-V6 (3 * load_b__),
   factor_c -> V7-V9 (3 * load_c__);
```

In this example, you are fitting a factor matrix of the following form:

```
            g_factor   factor_a   factor_b   factor_c

     V1      g_load1    load_a1
     V2      g_load2    load_a2
     V3      g_load3    load_a3
     V4      g_load4               load_b1
     V5      g_load5               load_b2
     V6      g_load6               load_b3
     V7      g_load7                          load_c1
     V8      g_load8                          load_c2
     V9      g_load9                          load_c3
```

The blanks in the factor matrix denotes fixed zeros. Because you do not specify the covariances by using the COV statement, g_factor, factor_a, factor_b, and factor_c are all uncorrelated in the model.

Modifying a FACTOR 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 a factor model, either exploratory or confirmatory. The reference model is referred to as the old model, while the model 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 FACTOR model, you can use the following extended FACTOR modeling language to make modifications from the old model before transferring the specifications to the new model.

Before discussing the syntax for modifying factor models, there are two cases where using the REFMODEL statement for defining new factor models are not recommended. First, if your old model is an exploratory factor-analysis model, then specification by using the FACTOR modeling language in the new model will replace the old model completely. In this case, the use of the REFMODEL statement is superfluous and should be avoided. Second, 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 will also replace the old model completely. Again, in this case 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 6748). 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:

```
proc tcalis;
   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.

In the new model, there is an additional parameter specification that does not exist in the old model. 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 '.'. This means 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 replaces 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 TCALIS statement. The options set in the FITINDEX statement will overwrite those set in the PROC TCALIS statement.

For the listing of fit indices and their definitions, see the section "Overall Model Fit Indices" on page 6895. Note that not all fit indices are available with all estimation methods, which is specified by the METHOD= option of the PROC TCALIS statement. See the section "Fit Indices and Estimation Methods" on page 6902 for more details.

The options of the FITINDEX statement are as follows:

ALPHAECV=α

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 TCALIS statement on page 6718.

ALPHARMS=α

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 TCALIS statement on page 6718.

CHICORRECT | CHICORR = *name* **|** *c*

specifies a correction factor c for the chi-square statistics for model fit. See the CHICORRECT= option of the PROC TCALIS statement on page 6719.

CLOSEFIT=*p*

defines the criterion value p for indicating a close fit. See the CLOSEFIT= option of the PROC TCALIS statement on page 6720.

DFREDUCE=*i*

reduces the degrees of freedom of the χ^2 test by i. See the DFREDUCE= option of the PROC TCALIS statement on page 6720.

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 TCALIS statement on page 6727.

NOINDEXTYPE

disables the display of index types in the fit summary table. See the NOINDEXTYPE option of the PROC TCALIS statement on page 6727.

OFF | 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 | 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 TCALIS:

ABSOLUTE	Absolute or stand-alone fit indices that measures the model fit without using a baseline model.
ALL	All fit indices available in PROC TCALIS.
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.
PARSIMONY	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 TCALIS:

INDCHI	Chi-square statistic for the independence model.
INDCHIDF	Degrees of freedom of the chi-square statistic for the independence model.
NACTCON	Number of active constraints.
NMOMENTS	Number of elements in the moment matrices being modeled.
NOBS	Number of observations assumed in the analysis.
NPARMS	Number of independent parameters.
NVAR	Number of variables.

Output Control of Absolute Fit Indices

You can use the following *names* to refer to the individual absolute fit indices available in PROC TCALIS:

CHISQ	Chi-square statistic for model fit.
CN	Hoelter's critical N.
CONTRIBUTION	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.
FUNCVAL	Optimized function value.
GFI	Goodness-of-fit index by Jöreskog and Sörbom.
PROBCHI	P-value of the chi-square statistic for model fit.
PROBELLIPTIC	P-value of the elliptical chi-square statistic.
RMSR	Root mean square residual.
SRMSR	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 TCALIS:

AGFI	Adjusted GFI.
AIC	Akaike information criterion.
CAIC	Bozdogan corrected AIC.
CENTRALITY	McDonald centrality measure.
ECVI	Expected cross-validation index.
LL_ECVI	Lower confidence limit for ECVI.
LL_RMSEA	Lower confidence limit for RMSEA.
PGFI	Parsimonious GFI.
PROBCLFIT	Probability of close fit.
RMSEA	Root mean squares of error approximation.
SBC	Schwarz Bayesian criterion.
UL_ECVI	Upper confidence limit for ECVI .
UL_RMSEA	Upper confidence limit for RMSEA.

Output Control of Incremental Fit Indices
You can use the following *names* to refer to the individual incremental fit indices available in PROC TCALIS:

BENTLERCFI	Bentler comparative fit index.
BENTLERNFI	Bentler-Bonett normed fit index.
BENTLERNNFI	Bentler-Bonett non-normed fit index.
BOLLENNFI	Bollen normed fit index (Rho1).
BOLLENNNFI	Bollen non-normed 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 TCALIS statement on page 6731. See the section "OUTFIT= SAS-data-set" on page 6829 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 TCALIS 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 TCALIS 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 6755
- PARTIAL statement on page 6790
- VAR statement on page 6810
- WEIGHT statement on page 6810

For example, consider the following statements:

```
proc tcalis;
   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 TCALIS statements. If you set these group-specific *options* in the PROC TCALIS 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 transfered from the PROC TCALIS 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*

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 TCALIS statement.

Options Available in the GROUP and PROC TCALIS Statements

These options are available in the GROUP and PROC TCALIS statements:

Option	Description
DATA= on page 6720	specifies the input data set
INWGT= on page 6723	specifies the data set that contains the weight matrix
OUTSTAT= on page 6731	specifies the data set for storing the statistical results
OUTWGT= on page 6731	specifies the data set for storing the weight matrix

See the section "Listing of PROC TCALIS Statement Options" on page 6718 for more details about these options. If you specify these options in the PROC TCALIS 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 TCALIS Statements

These options are available in the GROUP, MODEL and PROC TCALIS statements:

Option	Description
BIASKUR on page 6719	computes the skewness and kurtosis without bias corrections
EDF= on page 6721	defines nobs by the number of error df
KURTOSIS on page 6723	computes and displays kurtosis
NOBS= on page 6727	defines the number of observations (nobs)
PCORR on page 6732	displays analyzed and estimated moment matrix
PLOTS= on page 6733	specifies ODS Graphics selection
PWEIGHT on page 6810	displays the weight matrix
RDF | DFR= on page 6734	defines nobs by the number of regression df
RESIDUAL | RES on page 6734	computes the default residuals
RESIDUAL | RES= on page 6734	specifies the type of residuals
RIDGE on page 6735	specifies the ridge factor for covariance matrix
SIMPLE on page 6735	prints univariate statistics
VARDEF= on page 6736	specifies variance divisor
WPENALTY= on page 6736	specifies the penalty weight to fit correlations
WRIDGE= on page 6737	specifies the ridge factor for the weight matrix

If you specify these options in the PROC TCALIS 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

$<+|-><coefficient\ *>\ parameter\ <<+|-><coefficient\ *>\ parameter \ldots>$

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_i separated by a comparison operator. Valid operators are <=, <, >=, >, and = or, equivalently, LE, LT, GE, GT, and EQ. PROC TCALIS 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 easily express boundary constraints in LINCON statements, for many applications it is much more convenient to specify both the BOUNDS and the LINCON statements in the same PROC TCALIS call.

LINEQS Statement

LINEQS *equation < , equation . . . >* **;**

where *equation* represents:

dependent = term < + term . . . >

and each *term* represents one of the following:

- *coefficient-name < (number) > variable-name*
- *prefix-name < (number) > variable-name*
- *< number > 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 TCALIS statement or a MODEL statement. To completely specify a LINEQS model, you might need to add some subsidiary model specification statements. The following is the syntax for the LINEQS modeling language:

> **LINEQS** *model equations* ;
>   **STD** *partial variance parameters* ;
>   **COV** *covariance parameters* ;
>   **MEAN** *mean parameters* ;

In the LINEQS statement, you use equations to specify the linear functional relations among manifest or latent variables. Equations in the LINEQS statement are separated by commas.

In the STD 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 four types of parameters you can specify in equations, as shown in the following example:

```
lineqs
   V1 = 1.       F1 + E1,
   V2 = b2       F1 + E2,
   V3 = b2       F1 + E3,
   V4 = b4 (.4)  F1 + E4;
```

In this example, you have manifest variables V1–V4, 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), and one error variable (E-variable). The following four types of parameters have been specified:

- free or constrained parameters with starting values

  The coefficient parameter b4 in the fourth equation is a free parameter to estimate. A starting value for a parameter can be specified inside parentheses following the parameter name. In this case, the starting value for b4 is 0.4. As shown in the equations, it is not constrained with any other coefficient parameters because b4 is specified at only one location.

- free or constrained parameters without starting values

  The coefficient parameter b2 in the second and the third equations is a parameter to estimate. No starting value is given for the parameter. With the same parameter name b2, the path coefficients with predictor variable F1 in the second and the third equation are thus implicitly constrained to be equal.

- explicitly fixed parameter values

  Any fixed constant preceding a predictor variable is a fixed parameter (for example, the constant coefficient 1 for F1 in the first equation).

- implicitly fixed parameter values

  If there is neither a parameter nor a constant preceding a predictor variable in an equation, the coefficient associated with that predictor variable is assumed to be a fixed parameter at one.

For example, the coefficients preceding all error variables (E-variables) are fixed at 1. Any coefficients associated with unspecified predictor variables in equations are fixed zeros.

Parameters with no starting values will be initialized by various heuristic and effective methods in PROC TCALIS. See the section "Initial Estimates" on page 6914 for details.

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 double underscores '__'. Whenever a prefix name is encountered, the TCALIS 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.

Certainly, coefficient parameters are only a subset of parameters of a model. Other parameters like variance, covariance and mean parameters should be specified in the subsidiary model statements for the LINEQS modeling language—that is, the COV, MEAN, and STD statements.

Representing Latent Variables in the LINEQS Model

Because latent variables are widely used in structural equation modeling, PROC TCALIS needs a way to identify different types of latent variables 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 6832 for details about these naming rules. Essentially, latent factors (systematic sources) must start with letter 'F' or 'f'. Error terms must start with letter 'E', 'e', 'D', or 'd'. Prefix 'E' or 'e' represents the error term of an endogenous manifest variable. Prefix 'D' or 'd' represent represents 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 not distinguished in PROC TCALIS. Essentially, only the distinction between latent factors (systematic sources) and errors or disturbances (unsystematic sources) is critical in specifying a proper model analyzed by PROC TCALIS. Manifest variables in PROC TCALIS do not need to follow additional naming rules beyond those required by the general SAS system—they only need to have references in the input data set.

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 never appear in any places of any equation but are specified in the VAR statement.

The syntactic rules for equations are far from sufficient to define the system of equations that the LINEQS model would accept. You must also 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 equations. This also means that error and disturbance variables are always exogenous in the LINEQS model.

- An endogenous variable appearing 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 appearing on the left-hand side of an equation cannot appear on the right-hand side of the same equation.
- Each equation must contain one and only one **unique** error term, be it an E-variable for manifest outcome variable or a D-variable for latent outcome variable. If, indeed, you want to specify an equation without an error term, you might equivalently set the variance of the error term to a fixed zero in the STD statement.

Mean Structures in Equations

To fit a LINEQS model with mean structures, you can specify the MEANSTR option in the PROC TCALIS or the associated MODEL statement. Alternatively, you can use the Intercept variable in equations for including intercept terms or the MEAN statement to specify the mean parameters. Conceptually, the Intercept variable is a special “variable” containing the value one 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 +          F1 + E1,
   V2 =                   + b2       F1 + E2,
   V3 = a2      Intercept + b2       F1 + E3,
   V4 = a2      Intercept + b4 (.4)  F1 + E4;
```

In the first equation, a1, with a starting value at 10, is the intercept parameter for v1. The intercept parameters for V3 and V4 are constrained to be the same. They are named a2 in the equations. Even though there is no Intercept variable specified in the equation for endogenous variable V2, its intercept parameter is assumed to be a fixed zero. This is because once you use at least one Intercept variable in an equation or you specify at least one mean parameter in the MEAN statement, mean structure analysis is activated and all mean and intercept parameters have default values at zero, unless you specify parameter names or other fixed values explicitly in the equations or in the MEAN statement.

Modifying a LINEQS 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 a LINEQS model. The reference model will be referred to as the old model, while the model being defined 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

LINEQS modeling language 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 6758):

> **LINEQS** *model equations* **;**
>   **STD** *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 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 old 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, in the following two-group analysis you specify model 2 by referring to model 1 in the REFMODEL statement.

```
proc tcalis;
   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;
      std
         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 V3, in the old model are duplicated in the new model. All specifications in the STD and COV statements, except the covariance between E1 and E2, in the old model are also duplicated in the new model.
- Addition: The parameter cve23 for the covariance between E2 and E3 is added in the new model.
- Deletion: The specification of covariance between E1 and E2 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 V3 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 V2 as the outcome variable.

LISMOD Statement

LISMOD < *var_lists* > **;**

where *var_lists* represent one or more of the following:

- YVAR | YV | Y = *variables*
- XVAR | XV | X = *variables*
- ETAVAR | ETAV | ETA = *variables*
- XIVAR | XIV | XI | KSIVAR | KSIV | KSI = *variables*

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 *variable 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 structural model: _ALPHA_, _KAPPA_, _BETA_, _GAMMA_, _PHI_, _PSI_
- Matrices in measurement model for y-variables: _NUY_, _LAMBDAY_, _THETAY_
- Matrices in measurement model for x-variables: _NUX_, _LAMBDAX_, _THETAX_

See the section "Model Matrices in the LISMOD Model" on page 6841 for definitions of these matrices and their roles in the LISMOD modeling language. See the MATRIX statement on page 6776 for the details of parameter specification.

In the LISMOD statement, you can set the following four lists of variables:

- YVAR= list is for manifest variables y that are directly related to the endogenous latent eta-variables. Variables in the list are called y-variables.
- XVAR= list is for manifest variables x that are directly related to the exogenous latent ksi-variables. Variables in the list are called x-variables.
- ETAVAR= list is for endogenous latent variables η. Variables in the list are called eta-variables.
- XIVAR= list is for exogenous latent variables ξ. Variables in the list are called xi- or ksi-variables.

The order of variables in the lists of the LISMOD statement is important. The order 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 set 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 yvar= or xvar= lists is defined. See the section "LISMOD Submodels" on page 6843 for the details about LISMOD submodels that PROC TCALIS can handle.

An example of a LISMOD model specification is shown as follows:

```
proc tcalis;
   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 x1–x3, six y-variables y1–y6, one ξ-variable xi, and two η-variables eta1–eta2. The numbers of variables in these lists define the dimensions of the LISMOD model matrices. For example, matrix _LAMBDAY_ is 6×2, with y1–y6 as the row variables and eta1–eta2 as the column variables. Matrix _THETAX_ is 3×3, with x1–x3 as the row and column variables. In the MATRIX statements, you specify parameters in the elements of the matrices. After the matrix name, you specify in the parentheses '[' 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 MATRIX statement on page 6776 for a detailed description. In this example, the first specification of the parameters in the _LAMBDAY_ matrix starts from [,1]—meaning that it starts from the first column and will proceed downwards. As a result, the [1,1] element is a fixed constant 1.0, the [2,1] element is a free parameter called load3, and the [3,1] element is a free parameter called load4. Similarly, in the second specification in the _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 load5 and load6, respectively.

You can also specify the parameters of a row using similar notation. 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 both starting row and column numbers specified, such as that in the first specification in matrix _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 _THETAY_ as free parameters ey1, ey2, and ey3.

With the notation [,], there are no starting row and column numbers specified. Specification will start from the first valid element in the matrix and proceeds row-wise for all valid elements in the matrix. For example, in the matrix _LAMBDAX_ statement, the [1,1] element of matrix _LAMBDAX_ is a fixed constant 1, and the [1,2] and [1,3] elements are free parameters load1 and load2, respectively.

Modifying a LISMOD 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 LISMOD 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 LISMOD 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 LISMOD modeling language (see the section "LISMOD Statement" on page 6764). 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 make reference 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, in the following two-group analysis you specify model 2 by referring to model 1 in the REFMODEL statement.

```
proc tcalis;
   group 1 / data=d1;
   group 2 / data=d2;
   model 1 / group=1;
      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;
   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 the [2,1] element in matrix _THETAY_ and the diagonal of matrix _THETAX_, which are being 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

LMTESTS *option* < *option . . .* > **;**

where *option* represents one of the following:

- *display-option*
- *test-set*

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 TCALIS 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 TCALIS 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 TCALIS 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 TCALIS 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 TCALIS: 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 FACTOR Models

Keywords for Matrix-Based Regions

FACTERRV | FACTERRV
specifies the error variances.

FACTFCOR | FACTFCOR
specifies the covariances among factors.

FACTINTE | FACTINTE
specifies the intercepts.

FACTLOAD | FACTLOAD
specifies the factor loadings.

FACTMEAN | FACTMEAN
specifies the factor means.

See the section "Model Matrices in the FACTOR Model" on page 6856 for definitions of these FACTOR model matrices.

Keywords for Non-Matrix-Based Regions

ALL
specifies all parameter locations.

COV
specifies the covariances among factors.

COVERR
specifies the covariances among errors.

COVFACT | 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

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.

See the section "Matrix Representation of the LINEQS Model" on page 6833 for definitions of these model matrices and submatrices.

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.

COVEXOGERR
specifies covariances of errors and disturbances with other independent variables.

COVLV | COVFACT
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
specifies all possible effects of latent factors on latent factors.

LV–>MV | MV<–LV
specifies all possible effects of latent factors on manifest variables.

LV<–MV | MV–>LV
specifies all possible effects of manifest variables on latent factors.

MEANS | MEAN
specifies the means of independent factors.

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

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_.

COVFACT | COVLV
specifies all covariances among latent factors in _PHI_ when the ξ-variables exist, and in _PSI_ when the η-variables exist without the presence of the ξ-variables.

FIRSTMOMENTS
specifies all intercepts and means in _NUY_, _NUX_, _ALPHA_, and _KAPPA_.

INTERCEPTS | INTERCEPT
specifies all intercepts in _NUY_, _NUX_, and _ALPHA_.

LOADING | LOADINGS
specifies the coefficients in _LAMBDAY_ and _LAMBDAX_.

LV–>LV
specifies the effects of latent variables on latent variables. Depending on the type of LISMOD model, the _BETA_ and _GAMMA_ might be involved.

LV–>MV | MV<–LV
specifies the effects of latent variables on manifest variables. Depending on the type of LISMOD model, the _LAMBDAY_, _LAMBDAX_, and _GAMMA_ matrices might be involved.

MEANS | MEAN
specifies the mean parameters. Depending on the type of LISMOD model, the _ALPHA_ and _KAPPA_ matrices might be involved.

MV–>MV
specifies effects of manifest variables on manifest variables. Depending on the type of LISMOD model, the _BETA_ and _GAMMA_ matrices might be involved.

PATHS | PATH
specifies all path coefficients. Depending on the type of LISMOD model, the _LAMBDAY_, _LAMBDAX_, _BETA_, and _GAMMA_ matrices might be involved.

The LM Test Regions for MSTRUCT Models

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

Keywords for Matrix-Based Regions

RAMA | _A_ | RAMA
specifies the _RAMA_ matrix.

RAMALPHA | RAMALPHA
specifies the _RAMALPHA_ matrix.

RAMBETA | RAMBETA
specifies the _RAMBETA_ matrix.

RAMGAMMA | RAMGAMMA
specifies the _RAMGAMMA_ matrix.

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 _RAM21_ 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
specifies effects of latent variables on latent variables.

LV–>MV | MV<–LV
specifies effects of latent variables on manifest variables.

LV<–MV | 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
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 . . . >* **;**

In MATRIX statements, you specify fixed constants or free parameters for the elements of the model matrix referred by the *matrix-name*. 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 in the matrix. You can also assign initial values for the free parameters in the *parameter-spec* list.

The MATRIX statement is a subsidiary model specification statement of the 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 LISMOD modeling language

There are 12 model matrices in the LISMOD model and they correspond to the following *matrix-names* for the LISMOD modeling language:

- matrices in the measurement model for the y-variables

LAMBDAY	the matrix of regression coefficients of the y-variables on the eta-variables
NUY	the vector of intercept terms of the y-variables
THETAY	the error covariance matrix for the y-variables

- matrices in the measurement model for the x-variables

LAMBDAX	the matrix of regression coefficients of the x-variables on the ksi-variables
NUX	the vector of intercept terms of the x-variables
THETAX	the error covariance matrix for the x-variables

- matrices in the structural model

ALPHA	the vector of intercept terms of the eta-variables
BETA	the matrix of regression coefficients of the eta-variables on the eta-variables
GAMMA	the matrix of regression coefficients of the eta-variables on the ksi-variables
KAPPA	the mean vector for the ksi-variables
PHI	the covariance matrix for the ksi-variables
PSI	the error covariance matrix for the eta-variables

Valid Matrix Names for the MSTRUCT Modeling Language

the following *matrix-names* are valid for the MSTRUCT modeling language:

COV	the covariance matrix
MEAN	the mean vector

Specifying Locations in Model Matrices

There are four different ways to specify the starting *location* of the parameter matrix in question. Each way of specification leads to different continuation direction for assigning the parameters specified in the *parameter-spec* list. Assuming that there are n ($n \geq 1$) parameters specified in the *parameter-spec* list, the four different parameter assignment schemes due to different ways of starting *location* specification are described as follows:

[*i* , *j*] The elements in the *parameter-spec* list correspond to the diagonally continued matrix elements [*i,j*], [*i*+1,*j*+1], . . . , [*i*+*n-1*,*j*+*n-1*]. The number of elements is defined by the length of the *parameter-spec* and is eventually terminated by the matrix boundaries. If the *parameter-spec* list contains just one element (constant or variable), then it is assigned to the matrix element [*i,j*].

[*i* ,] In this type of row-wise specification, the elements in the *parameter-spec* list correspond to the horizontally continued matrix elements [*i,1*], [*i,2*], . . . , [*i,n*]. For a rectangular or non-symmetric square matrix with *k* columns, the number of elements in *parameter-spec* should not exceed *k*. For a symmetric matrix, because you need to specify lower triangle elements only (upper triangular elements are duplicated from the lower triangle elements automatically), the number of elements in *parameter-spec* list for this type of row-wise specification should not exceed *i*.

[, *j*] In this type of column-wise specification, the elements in the *parameter-spec* list correspond to the vertically continued matrix elements [*i,j*], [*i*+1,*j*], . . . , [*i*+*n-1*,*j*]. For a rectangular or nonsymmetric square matrix, the value of *i* is 1. For a symmetric matrix, because you need to specify the lower triangular elements only (upper triangular elements are duplicated from the lower triangle elements automatically), the value of *i* in this type of column-wise specification is *j*.

[,] Unspecified location: The elements of the *parameter-spec* list are assigned to all valid matrix positions starting at the element [1,1] and continuing row-wise, until reaching the last element of the *parameter-spec* list or the last valid matrix position. For a symmetric matrix, the valid positions are the elements in the lower triangle since the other triangle receives the symmetric allocation automatically.

When there is no starting *location* specified, it is equivalent to using [,] for an *unspecified location*.

Specifying Parameters in Model Matrices

The *parameter-spec* list contains numeric values or parameter names, or both, that are assigned to the matrix elements starting at a specified position and proceeding in a specified direction, which are determined by the way of the starting *location* specification.

The syntax for the *parameter-spec* list is the same as the *parameter-spec* list for the STD statement. There are three types of specification in the *parameter-spec* list, as described in the following.

- Parameter names alone represent parameters in the model without starting values.
- Parameter names followed by numbers inside a pair of parentheses represent parameters in the model with starting values provided in the parentheses.
- Numbers alone represent fixed constant values for the corresponding matrix elements.

For example, consider the following *parameter-spec* list.

```
0. 1. A2-A5 (1.4 1.9 2.5) 5.
```

The first two matrix elements are assigned with fixed values 0 and 1, respectively. The next element is a parameter named A2 with no initial value provided. The next three matrix elements are assigned with parameters A3, A4, and A5 with initial values 1.4, 1.9, and 2.5, respectively. The next matrix element is assigned with a constant 5. At first glance, the initial values in the parentheses are seemingly assigned to A2, A3, and A4, respectively. However, because A2–A5 is a shorthand of four consecutive parameters, the preceding *parameter-spec* list is equivalent to:

```
0. 1. A2 A3 A4 A5 (1.4 1.9 2.5) 5.
```

With this explicit specification, it is clearer to see that the initial values in the parentheses are assigned to parameters A3–A5, respectively.

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 double underscores '__'. Whenever a prefix name is encountered, the TCALIS 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. For example, consider the following *parameter-spec* list.

```
0. 1. 4 * A__ (1.4 1.9 2.5) 5.
```

If the prefix name A__ has not been specified previously, then A1, A2, A3, and A4 are generated by the **4 * A__** syntax. However, if the prefix name A__ has been specified, say, twice, previously, then A3, A4, A5, and A6 are generated by the same **4 * A__** syntax.

To avoid unintentional equality constraints, the prefix names should not coincide with explicitly defined parameter names. For example, consider the following specification.

```
matrix _mean_ = B1 B2 2 * B__;
```

If the prefix name B__ has not been specified previously, then B1 and B2 are generated for the third and the fourth elements, respectively, of the mean vector. However, these two parameter names are the same as those assigned explicitly to the first and the second elements of the mean vectors, resulting to two equality constraints that might not have been intended.

Modifying a Parameter Specification from a Reference Model

If you define a new LISMOD or MSTRUCT model by using a reference (or base) model in the REFMODEL statement, in some situations you might want to modify some parameter specifications from the MATRIX statement of the reference model before transferring the specifications to the new model. If you want to change a particular MATRIX parameter specification from the reference model, you can simply put the corresponding parameter location with a new parameter specification in the new model. If you want to delete a particular parameter location from the reference model, you can put that parameter location with the missing value '.' in the new model. For example, suppose you are defining a new LISMOD model by using the REFMODEL statement and the covariance between variables F1 and F2 is defined as a fixed or free parameter in the reference model. If you do not want this parameter specification to be copied into your new model, you can use the following specification in the new model:

```
matrix _PHI_  F1 F2 = .;
```

Notice that the missing value syntax is valid only when you use the REFMODEL statement. See the section "Modifying a LISMOD Model from a Reference Model" on page 6766 for a more detailed example of LISMOD model respecification. See the section "Modifying an MSTRUCT Model from a Reference Model" on page 6785 for a more detailed example of MSTRUCT model respecification.

MEAN Statement

MEAN *assignment* < , *assignment* . . . > **;**

where *assignment* represents:
variables = parameter-spec

The MEAN statement invokes the fitting of the mean structures of the model. The MEAN statement is one of the subsidiary model specification statements for supplementing modeling specification. You can use the MEAN statements to specify mean parameters in the FACTOR, LINEQS, and PATH modeling languages. You can also use the MEAN statement to specify the intercept parameters in the FACTOR or PATH modeling languages. The syntax of the MEAN statement is essentially the same as that of the STD or PVAR statement.

In each *assignment* of the MEAN statement, you put the *variables* on the left-hand side of the equal sign and specify the parameters on the right-hand side *parameter-spec*. Like elsewhere in the PROC TCALIS syntax, in *parameter-spec* you can put numbers for fixed values or parameter names for free or constrained parameters. Parameter names are optionally followed by numbers in parentheses for starting values. See the STD statement on page 6807 for examples.

In the FACTOR or PATH model, the variables in each *assignment* can be exogenous or endogenous. If a variable in the *variables* list is exogenous, the corresponding parameter is the mean parameter for the variable. Otherwise, the corresponding parameter is the intercept parameter for the variable. In the LINEQS model, however, only non-error type exogenous variables (that is, not E- or D- variables) can be used in the MEAN statement. The parameters are for the means of the exogenous variables. You cannot specify the intercept parameters in the MEAN statement of the LINEQS modeling language. Instead, you should specify intercepts in the equations of the LINEQS statement.

Automatic Mean Parameters among Exogenous Variables

If the names of any exogenous manifest variables in the LINEQS model are not included in the MEAN statement, the means of those exogenous manifest variables are assumed to be parameters to estimate. These parameters will be named automatically with the prefix _Add and appended with integer suffixes. However, all other mean parameters other than those for exogenous manifest variables are fixed zeros by default unless they are specified otherwise in the MEAN statement. In addition, all intercept parameters are assumed to be fixed zeros, unless they are specified otherwise in the LINEQS statement.

Modifying a Parameter Specification from a Reference Model

If you define a new FACTOR, LINEQS, or PATH model by using a reference (or base) model in the REFMODEL statement, in some situations you might want to modify some parameter specifications from the MEAN statement of the reference model before transferring the specifications to the new model. If you want to change a particular MEAN parameter specification from the reference model, you can simply put the corresponding parameter location with a new parameter specification in the new model. If you want to delete a particular parameter location from the reference model, you can put that parameter location with the missing value '.' in the new model. For example, suppose you are defining a new LINEQS model by using the REFMODEL statement and the mean of variable F1 is defined as a fixed or free parameter in the reference model. If you do not want this parameter specification to be copied into your new model, you can use the following specification in the new model:

```
mean F1 = .;
```

Notice 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 6750 for a more detailed example of FACTOR model respecification. See the section "Modifying a LINEQS Model from a Reference Model" on page 6761 for a more detailed example of LINEQS model respecification. See the section "Modifying a PATH Model from a Reference Model" on page 6792 for a more detailed example of PATH model respecification.

In LINEQS models, all means of exogenous manifest variables are free parameters to estimate unless you specify otherwise in the MEAN statement (see the section "Automatic Mean Parameters among Exogenous Variables" on page 6780). This is also true for LINEQS models specified under the REFMODEL statement. In the reference model, mean parameters generated by PROC TCALIS are named with the _Add prefix and appended with integer suffixes. These parameter specifications in the 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 options, and model respecifications are the additional mean parameters with the _Add prefix generated for the remaining unspecified means of exogenous manifest variables. In this way, the generated mean parameters for the exogenous mainfest variables in the new model are not constrained to be the same as the corresponding means in the reference model. If you want any of these mean parameters to be constrained across the models, you must specify them explicitly in the MEAN statement of the reference model so that the constrained mean specifications are transferred to the new model.

MODEL Statement

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 tcalis;
   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 TCALIS statement. If you set these *options* in the PROC TCALIS statement, they will apply to all models, unless you respecify them in the local MODEL statements. If you want to apply some *options* only to a particular model, set these *options* in the MODEL statement corresponding to that model.

You can also set group-specific *options* in the MODEL statement. These group *options* will apply to the groups that are specified in GROUP= option of the MODEL statement. See the section “Options Available in the GROUP and PROC TCALIS Statements” on page 6757 for a detailed descriptions of these group *options*.

Option Available in MODEL Statement Only

LABEL | 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 quote strings for labels.

GROUP | GROUPS=*int-list*

specifies a list of integers *int-list* of group numbers representing the groups to be fitted by the model.

Options Available in the MODEL and PROC TCALIS Statements

These options are available in the MODEL and PROC TCALIS statements. If you specify these options in the PROC TCALIS statement, they are transferred to all MODEL statements. These options might be overwritten by the respecifications in the local MODEL statements.

Option	**Description**
CORRELATION on page 6720	analyzes correlation matrix
COVARIANCE on page 6720	analyzes covariance matrix
DEMPHAS= on page 6720	emphasizes the diagonal entries
EFFPART | TOTEFF on page 6721	displays total, direct, and indirect effects
INEST= on page 6722	specifies the data set that contains the initial values and constraints
INMODEL | INRAM= on page 6722	specifies the data set that contains the model specifications
MEANSTR on page 6725	analyzes the mean structures
MODIFICATION on page 6726	computes modification indices
NOMOD on page 6727	suppresses modification indices
NOORDERSPEC on page 6727	displays model specifications and results according to the input order
NOPARMNAME on page 6727	suppresses the printing of parameter names in results
NOSTAND on page 6727	suppresses the standardized output
NOSTDERR on page 6727	suppresses standard error computations
ORDERSPEC on page 6730	orders the model output displays according to the parameter types within each model
OUTEST= on page 6730	specifies the data set that outputs the estimates and their covariance matrix
OUTMODEL | OUTRAM= on page 6731	specifies the output data set for storing the model specification and results
PARMNAME on page 6732	displays parameter names in model specifications and results
PDETERM on page 6732	computes the determination coefficients
PESTIM on page 6733	prints parameter estimates
PINITIAL on page 6733	prints initial pattern and values
PLATCOV on page 6733	computes the latent variable covariances and scoring coefficients
PRIMAT on page 6733	displays results in matrix forms
STDERR on page 6736	computes the standard errors

Options Available in MODEL, GROUP, and PROC TCALIS 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 fitted by the associated models in the MODEL statements. This is a little bit 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 TCALIS Statements" on page 6757 for a reference of these options.

MSTRUCT Statement

MSTRUCT *VAR* = variables **;**

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. To complete the specification, you must use additional MATRIX statements to specify the parameters:

MSTRUCT *variables* **;**
    MATRIX _COV_ *parameters-in-matrix* **;**
    MATRIX _MEAN_ *parameters-in-matrix* **;**

You use the MATRIX _COV_ statement to specify the covariance and variance parameters in the structured covariance matrix. When applicable, you use the MATRIX _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 6776 for details.

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 _COV_ and the _MEAN_ matrices. The variables specified in the list should be present in the input data set 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 to novice users. The LINEQS, LISMOD, PATH, or RAM modeling language should be considered first for latent variable modeling.

Modifying an MSTRUCT 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 an MSTRUCT model. The reference model will be referred to as the old model, while the model that makes reference to the 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 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 in the section "MSTRUCT Statement" on page 6784. The syntax for respecifying or modifying an MSTRUCT model takes the following form:

> **MSTRUCT ;**
> > **MATRIX _COV_** *parameters-in-matrix* **;**
> > **MATRIX _MEAN_** *parameters-in-matrix* **;**

First, 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 make reference 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 only optional for the new model.

Second, 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:

```
proc tcalis;
   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 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

NLINCON | NLC *constraint* < , *constraint* . . . > ;

where *constraint* represents one of the following:

- *number operator variable-list number operator*
- *variable-list operator number*
- *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 <=, <, >=, >, and = (or, equivalently, LE, LT, GE, GT, and EQ).

PROC TCALIS cannot enforce the strict inequalities < or > but instead treats them as <= and >=, 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:

```
nlincon    xx = 1;
xx = x1 * x1 + u1;
```

Note that x1 and u1 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:

```
nlincon  0. <= xx1-xx3,
               xx1-xx3 <= 10;
nlincon 0. <= xx1-xx3 <= 10.;
nlincon 10. >= xx1-xx3 >= 0.;
```

NLOPTIONS Statement

NLOPTIONS *options* **;**

Many options that are available in SAS/OR PROC NLP can be specified for the optimization subroutines in PROC TCALIS 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 “Nonlinear Optimization: The NLOPTIONS Statement” on page 391 in Chapter 18, “Shared Concepts and Topics.”

See the section “Use of Optimization Techniques” on page 6915 for more information about the use of optimization techniques in PROC TCALIS.

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 TCALIS 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 tcalis 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:

```
proc tcalis 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 TCALIS run. That is, you cannot specify OUTWGT=outwgt, OUTSTAT=outstat, or OUTMODEL=outmodel more than once in the PROC TCALIS run without causing file invocation problems (however, multiple specification of the same input file is not a problem).

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 TCALIS 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 TCALIS 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 TCALIS statement because the output file information is automatically propagated from the PROC TCALIS 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 *parameter(s)* < < = > *number(s)* >
< < , > *parameter(s)* < < = > *number(s)* > . . . > **;**

The PARAMETERS statement defines additional parameters that are not specified in your models. You can specify more than one PARAMETERS statement. The *parameters* can be followed by an equal sign and a number list. The values of the *numbers* list are assigned as initial values to the preceding parameters in the *parameters* list. For example, each of the following statements assigns the initial values ALPHA=.5 and BETA=-.5 for the parameters used in SAS programming statements:

```
parameters alfa beta=.5 -.5;
parameters alfa beta (.5 -.5);
parameters alfa beta .5 -.5;
parameters alfa=.5 beta (-.5);
```

The number of parameters and the number of values do not have to match. When there are fewer values than parameter names, either the RANDOM= or START= option is used. When there are more values than parameter names, the extra values are dropped. Parameters listed in the PARAMETERS statement can be assigned initial values by program statements or by the START= or RANDOM= option in the PROC TCALIS 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 6810 for more details.

CAUTION: The OUTMODEL= or OUTMODEL= 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 TCALIS statement.

PATH Statement

PATH *path* < , *path* . . . > ;

where *path* represents:

variable direction variable parameter-spec

and *direction* represents one of the following: –>, >, <–, <

In the PATH statement, you specify the paths in your model. You can specify at most one PATH statement in a model, within the scope of either the PROC TCALIS statement or a MODEL statement. To complete the PATH model specifications, you might need to add some subsidiary model specification statements. The following is the syntax for the PATH modeling language:

PATH *path lists* ;
PVAR *partial variance parameters* ;
PCOV *partial covariance parameters* ;
MEAN *mean parameters* ;

In the PATH statement, you specify paths of variables in your model, excluding paths from the errors or disturbances. Paths in the PATH statement are separated by commas. In the PVAR statement, you specify parameters for either variances or partial variances. In the PCOV statement, you specify parameters for either covariances or partial covariances. In the MEAN statement, you specify parameters for either means or intercepts. For details about these subsidiary model specification statements, refer to the syntax of the individual statements.

In each path, you have one outcome variable and one predictor variable. The variables in a path can be manifest or latent. The specified direction determines the predictor and outcome variables. The *variable* being aimed at is the outcome variable and the other *variable* is the predictor variable. The path coefficient parameter is specified after each path in *parameter-spec*. For example, in the following statement you specify a model with four paths, each with predictor variable F1 and one outcome manifest variable.

```
PATH
   V1   <-   F1     1,
   V2   <-   F1     b2,
   V3   <-   F1     b2,
   V4   <-   F1     b4  (.4);
```

As shown in the example, the *parameter-spec* is one of the following three types:

- fixed path coefficients

  The *parameter-spec* for the first path to V1 is a fixed path coefficient at 1.

- free or constrained parameters without starting values

  The *parameter-specs* for the second and third paths to V2 and V3, respectively, are the same path coefficient parameter b2 with no starting value given. Because the same parameter name is used for the two path coefficients, the coefficients are constrained in the model.

- free or constrained parameters with starting values

  The *parameter-spec* for the last path to V4 is a free parameter named b4 with a starting value at 0.4.

If an outcome variable has more than one predictor variable, you have more than one path specification for the outcome variable. The following PATH statement shows an extension of the previous specification.

```
PATH
   V1   <-   F1    1,
   V3   ->   V1    b1  (.2),
   V2   <-   F1    b2,
   V3   <-   F1    b2,
   V4   <-   F1    b4  (.4);
```

As compared with the previous example, an additional path is specified in second path specification of the current PATH statement. In the specification V3 is a predictor of V1, in addition to the predictor F1 in the first path specification. The additional path V3 –> V1 has a path coefficient parameter named b1, with a starting value at 0.2.

Note that because the display of path results follows the way you specify the paths, you should arrange the order of the input paths in the most desirable way.

Modifying a PATH 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 PATH 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 PATH modeling language to make modifications within the scope of the MODEL statement for the new model. The syntax is the same as the ordinary PATH modeling language, as in the section “PATH Statement” on page 6791. The respecification syntax for a PATH model is as follows:

PATH *path lists* **;**
  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:

```
proc tcalis;
   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 V1 and V2 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 V3 <– F1 replaces the same path in the old model with parameter load1 for the path coefficient. Thus, in the new model paths V3 <– F1 and v2 <– F1 are now constrained to have the same path coefficient parameter load1.

PCOV Statement

PCOV *assignment < , assignment . . . >* **;**

where *assignment* represents:

variables $<$ $*$ *variables2* $>$ = *parameter-spec*

The PCOV statement defines the covariance or partial covariance parameters in the PATH model. You can use the PCOV statement with the PATH modeling language only. On the left-hand side of each assignment, variables involved are listed in the variables and variables2 lists. In the parameter-spec list, covariance or partial covariance parameters are specified. Names in the parameter-spec represent covariance or partial covariance parameters to estimate. Numbers enclosed in parentheses in the parameter-spec list are initial values for the parameters. Numbers not enclosed in parentheses in the parameter-spec list are fixed values for covariances or partial covariances. The *assignment*s in the PCOV statement must be separated by commas. You can specify only one PCOV statement in each model specification.

The syntax of the PCOV statement is the same as the COV statement. Refer to the COV statement on page 6739 for details about specifying within- and between- list (partial) covariances.

The concept behind the PCOV statement, however, is broader than that of the COV statement. The PCOV statement supports the partial covariance parameter specification in addition to the covariance parameter specification. The COV statement supports only the latter type of parameters. This discrepancy is also reflected from the sets of *variables* and *variables2* 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, both exogenous and endogenous variables can be specified. If both variables are exogenous in a specification of the PCOV statement, you are defining a covariance parameter between the variables. If both variables are endogenous, you are defining a partial covariance parameter between of the variables. This partial covariance is usually interpreted as 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.

Automatic Covariance Parameters among Exogenous Manifest Variables

In the PATH model, all covariances among exogenous manifest variables are automatically free parameters to estimate unless you specify otherwise in the PCOV statement. The parameter names for these covariances will be generated by PROC TCALIS with the prefix _Add, appended with unique integer for each occurrence. If you want any of these covariances to be fixed values, you must specify them explicitly in the PCOV statement.

By default covariances among exogenous latent variables are fixed zeros unless they are specified otherwise in the PCOV statement. The same applies to any partial covariance among endogenous variables and to any partial covariance between an exogenous latent variable and an exogenous manifest variable.

This is different from the case of partial variance parameters specified in the PVAR statement. Variances of **all** exogenous variables and partial variances of **all** endogenous variables are automatically free parameters if they are not specified otherwise in the PVAR statement.

Modifying a Parameter Specification from a Reference Model

If you define a new PATH model by using a reference (or base) model in the REFMODEL statement, in some situations you might want to modify some parameter specifications from the PCOV statement of the reference model before transferring the specifications to the new model. If you want to change a particular PCOV parameter specification from the reference model, you can simply put the corresponding parameter location with a new parameter specification in the new model. If you want to delete a particular parameter location from the reference model, you can put that parameter location with the missing value '.' in the new model. For example, suppose you are defining a new PATH model by using the REFMODEL statement and 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 parameter specification to be copied into your new model, you can use the following specification in the new model:

```
PCOV F1 V2 = .;
```

Note that the missing value syntax is valid only when used with the REFMODEL statement. See the section "Modifying a PATH Model from a Reference Model" on page 6792 for a more detailed example of PATH model respecification.

In the PATH model, all covariances among exogenous manifest variables are free parameters to estimate unless you specify otherwise in the PCOV statement (see the section "Automatic Covariance Parameters among Exogenous Manifest Variables" on page 6794). This is also true for PATH models specified under the REFMODEL statement. In the reference model, covariance parameters generated by PROC TCALIS are named with the _Add prefix and appended with integer suffixes. These parameter specifications in the 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 respecifications are the additional covariance parameters with the _Add prefix generated for the unspecified covariances among exogenous manifest variables. In this way, the covariance parameters in the new model are not constrained to be the same as the corresponding parameters in the reference model. If you want any of these covariance parameters constrained across the models, you must specify them explicitly in the PCOV statement of the reference model so that the covariance specifications are transferred to the new model.

PVAR Statement

PVAR *assignment* < , *assignment* . . . > **;**

where *assignment* represents:
  variables = *parameter-spec*

The PVAR statement defines the variance or partial variance parameters in FACTOR and PATH models. In each *assignment* in the PVAR statement, you specify *variables* on the left-hand side and the *parameter-spec* on the right-hand side of an equal sign. The *assignments* in the PVAR statement must be separated by commas. The *variables* listed on the left-hand side of the equal sign are any variables defined in the model. If a variable in the list is an exogenous variable, then the corresponding parameter in the *parameter-spec* list is a variance parameter of the variable. If a variable in the list is an endogenous variable, then the corresponding parameter in the *parameter-spec* is a partial variance parameter of the variable. In most cases, the partial variance of an endogenous variable can be interpreted as the error variance for the variable.

The syntax of the PVAR statement is exactly the same as that of the STD statement. As in the STD statement, in the PVAR statement you can specify free or constrained parameters with or without starting or fixed values. You can also use prefixes to generate parameter names. Refer to the STD statement on page 6807 for details and examples.

The difference between the PVAR and the STD statements is conceptual in nature. In the STD statement, you can specify only variance parameters for exogenous variables. However, in the PVAR statement, both exogenous and endogenous variables can be put on the *variables* list. This treatment is needed in the FACTOR and PATH modeling languages because error variables in these models are not explicitly named.

Automatic Variance and Partial Variance Parameters among Variables

If the name of a variable (manifest or latent) in the FACTOR or PATH model does not appear in the PVAR statement, the variance or partial variance of the variable is assumed to be a parameter to estimate. These parameters will be named automatically with the prefix _Add and appended with integer suffixes. Therefore, the entire PVAR statement is optional in the sense that all variance and partial variance parameters will be generated automatically for the FACTOR or PATH model in PROC TCALIS. However, if you need to constrain these parameters, or assign fixed values, the PVAR statement specification must be used.

Modifying a Parameter Specification from a Reference Model

If you define a new FACTOR or PATH model by using a reference (or base) model in the REFMODEL statement, in some situations you might want to modify some parameter specifications from the PVAR statement of the reference model before transferring the specifications to the new model. If you want to change a particular PVAR parameter specification from the reference model, you can simply put the corresponding parameter location with a new parameter specification in the new model. If you want to delete a particular parameter location from the reference model, you can put that parameter location with the missing value '.' in the new model. For example, suppose you are defining a new PATH model by using the REFMODEL statement and the partial variance of variable V2 is defined as a fixed or free parameter in the reference model. If you do not want this parameter specification to be copied into your new model, you can use the following 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 6750 for a more detailed example of FACTOR model respecification. See the section "Modifying a PATH Model from a Reference Model" on page 6792 for a more detailed example of PATH model respecification.

In confirmatory factor or PATH models, all variances or partial variances among variables are free parameters to estimate unless you specify otherwise in the PVAR statement (see the section "Automatic Variance and Partial Variance Parameters among Variables" on page 6796). This is also true for confirmatory factor and LINEQS models specified under the REFMODEL statement. In the reference model, (partial) variance parameters generated by PROC TCALIS are named with the _Add prefix and appended with integer suffixes. These parameter specifications in the 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 options, and model respecifications do the remaining unspecified (partial) variances of variables generate additional (partial) variance parameters with the _Add prefix and integer suffixes. In this way, the (partial) variance parameters in the new model are not constrained to be the same as the corresponding parameters in the reference model. If you want any of these (partial) variance parameters to be constrained across the models, you must specify them explicitly in the PVAR statement of the reference model so that the (partial) variance specification is transferred to the new model.

RAM Statement

RAM *list-entry* < , *list-entry . . .* > **;**

where *list-entry* represents:

parameter-type variable1 < *direction* > < *variable2* > *parameter-spec*

and *direction* can be one of the following: –>, >, <–, <

RAM stands for the reticular action model, developed by McArdle(1980). In the RAM statement, you specify various kinds of parameters in the model. In each *list-entry*, you specify the type of parameter in *parameter-type*, variables involved in *variable1* and, if applicable, in *variable2*, path direction in *direction*, parameter name and value in *parameter-spec*. *List-entries* must be separated by commas.

Types of RAM Parameters

The first keyword in each RAM *list-entry* is the parameter type, represented as *parameter-type* in the syntax. You can use one of the following keywords (not case-sensitive) for *parameter-type*:

_A_ or _RAMA_	a parameter in the *A* matrix of the RAM model (see the PATH keyword)
_P_ or _RAMP_	a parameter in the *P* matrix of the RAM model (see the PCOV and PVAR keywords)
_W_ or _RAMW_	a parameter in the vector for means and intercepts (see the MEAN and INTERCEPT keywords)
INTERCEPT	the intercept of an endogenous variable, manifest or latent
MEAN	the mean of an exogenous variable, manifest of latent
PATH	the path coefficient for a pair of outcome and predictor variables
PCOV	the partial covariance or covariance between a pair of variables
PVAR	the partial variance or variance of a variable

Keywords _A_, _P_, and _W_ refer to the matrix elements of the RAM model matrices _A_, _P_, and _W_, respectively. See the section "Model Matrices in the RAM Model" on page 6848 for detailed descriptions of these matrices. Elements in matrix _A_ are path coefficients; therefore the _A_ keyword is the same as the PATH keyword for specifying these matrix elements. Elements in matrix _P_ are either (partial) variances or (partial) covariances; therefore the _P_ keyword is the same as the PVAR or PCOV keyword for specifying these matrix elements. Elements in matrix _W_ are either means or intercepts; therefore the _W_ keyword is the same as the MEAN or INTERCEPT keyword for specifying these matrix elements.

Path Coefficients Parameters with Keyword _A_ or PATH

To specify the paths and the associated coefficients, you can use either the _A_ or PATH keyword for *parameter-type*. A pair of variables must be specified in *variable1* and *variable2*. The *direction* must also be inserted between the pair of variables with the PATH keyword. The following is an example of the list-entries for path coefficients:

```
path    a -> b    x1,
path    c <- b    x2,
```

The first path is from variable a to variable b with a path coefficient parameter named x1. The second path is from variable b to variable c with a path coefficient parameter named x2. However, if you use the _A_ keyword, you must not specify the *direction*, as it is implicitly assumed to be an arrow pointing to left, <–. For example, the following specification using the _A_ keyword is equivalent to the previous specification.

```
_A_    b   a     x1,
_A_    c   b     x2,
```

Partial Variance and Variance Parameters with Keyword _P_ or PVAR

To specify the variance or partial variance parameters, you can use the _P_ or PVAR keyword for *parameter-type*. If you use _P_, *variable1* and *variable2* must be the same variable name. If you use PVAR, only *variable1* needs to be specified. Specification for *direction* should be omitted. If the variable is exogenous, you are defining a variance parameter. Otherwise, you are defining a partial variance parameter. Partial variance in the RAM model can be interpreted as the error variance for an endogenous variable.

For example, assume that b is an exogenous variable in your model. In the following specification, you define the variance of b as a parameter named v1, with an initial estimate of 1.0:

```
PVAR    b     v1 (1.0),
```

This specification is equivalent to the following specification by using the _P_ keyword for *parameter-type*:

```
_P_   b  b    v1 (1.0),
```

Partial Covariance and Covariance Parameters with Keyword _P_ or PCOV

To specify the covariance or partial covariance parameters, you can use the _P_ or PCOV keyword for *parameter-type*. You must specify both *variable1* and *variable2*, where *variable1* and *variable2* are different variable names. Specification for *direction* should be omitted. Whereas it is a semantic error if *variable1* and *variable2* are the same name with the PCOV keyword, it is permissible to do so with the _P_ keyword. However, for the latter you are supposed to define a parameter location for the variance or partial variance of a variable, instead of the covariance or partial covariance parameter that is considered in this section.

The following shows a specification of a covariance parameter:

```
PCOV   a  b   v12 (0.5),
```

If both a and b are exogenous variables in your model, the preceding code specifies the covariance between a and b as a free parameter named v12, with a starting value at 0.5. Equivalently, you can use the _P_ keyword, as shown in the following specification:

```
_P_   a  b   v12 (0.5),
```

Note that if both a and b are both endogenous variables in your model, you can still use the preceding specification. But the interpretation is different. For endogenous a and b, v12 is treated as a partial covariance parameter. In almost all situations, this partial covariance is interpreted as the error covariance for variables a and b. Or, the model assumes correlated errors associated with variables a and b.

In fact, you can also specify the partial covariance between an exogenous variable and an endogenous variable. Suppose that a is exogenous and b is endogenous in your model. The preceding specifications are for specifying the covariance parameter between a and the error term for b. This is certainly a cumbersome interpretation. Fortunately, covariances between exogenous variables and error variables are considered inelegant in most statistical models. This kind of specification should be rare in practice.

Means and Intercepts with Keyword _W_, INTERCEPT, or MEAN

To specify the mean or intercept parameters, you can use the _W_, INTERCEPT, or MEAN keyword for *parameter-type*. You must specify *variable1* in the *list-entry*. Specifications for *variable2* and *direction* are not applicable and should be omitted. With the MEAN keyword, *variable1* is supposed to be an exogenous variable. With the INTERCEPT keyword, *variable1* is supposed to be an endogenous variable. With the _W_ keyword, however, *variable1* can be either exogenous or endogenous. In this case, whether a mean or an intercept parameter is being specified is determined by the TCALIS procedure.

For example, in the following specification the mean of f1 is a free parameter named mean1 while the intercept for a is a constant value 2.0:

```
mean        f1   mean1,
intercept    a   2.0,
```

Equivalently, you can specify using the _W_ keyword, as shown in the following:

```
_W_          f1    mean1,
_W_           a    2.0,
```

Variable Names for *variable1* and *variable2*

In each *list-entry*, you have to specify a variable name for *variable1* and possibly a variable name for *variable2* also. PROC TCALIS needs to distinguish names for observed and latent variables in your model. Variable names that are referenced in the input data set are for observed variables; otherwise, they are for latent variables. Therefore, you can make up any latent variable names in the RAM model specification, as long as these names follow the general SAS naming conventions and do not match any of the names in the input data set. Notice that in the RAM model specification, all latent variable names are for the latent factors, and names for error or disturbance terms are never needed. Error variances and covariances can be specified using the corresponding endogenous variables names as references in the *list-entries* with the PCOV or PVAR keyword for parameter-type. Hence, creating names for error or disturbance terms is not necessary in the RAM model specification.

Parameter Names and Initial Values

After you specify the type of parameter in *parameter-type*, the associated variables in *variable1* and *variable2* (when applicable), and the *direction* for paths (if applicable), the last piece information of *list-entry* is *parameter-spec*, which can be one of the following three types:

- fixed constants

  For example, in the following specification the intercept of variable a is fixed at 2.0.

  ```
  intercept    a    2.0,
  ```

- free or constrained parameter names without starting values

  For example, in the following specification the mean of variable f1 is a parameter named mean1.

  ```
  mean        f1    mean1,
  ```

  The starting value of mean1 is generated by PROC TCALIS.

- free or constrained parameter names with starting values

  For example, in the following specification the variance (if b is exogenous in the model) or error variance (if b is endogenous in the model) of variable b is a parameter named v1, with a starting value 1.0.

  ```
  PVAR       b     v1 (1.0),
  ```

Automatic Variance and Partial Variance Parameters

If the partial variance or variance parameter for any variable (manifest or latent) in the RAM model is not explicitly specified in the RAM *list-entries*, it will be added automatically as a free parameter. Each parameter is named with the prefix _Add and appended with a unique integer. Therefore, PVAR *parameter-type* entries for variables are optional when they are free and unconstrained parameters. However, if you need to set constraints on these parameters or to assign fixed parameter values, PVAR *list-entries* for the corresponding variables are necessary.

Automatic Covariance Parameters

All covariances among exogenous *manifest* variables in the RAM model are free parameters automatically unless specified otherwise in the *list-entries* with the PCOV keyword for *parameter-type*. Each automatic covariance parameter is named by PROC TCALIS with the _Add prefix and appended with an unique integer. If you want any of these covariances to be fixed values or constrained via name references, you must specify them explicitly in the PCOV *list-entries*.

Covariances among exogenous *latent* variables by default are fixed zeros unless specified otherwise in the PCOV *list-entries*. The same applies to any partial covariance among endogenous variables and to any partial covariance between an exogenous latent variable and an exogenous manifest variable.

Automatic Mean Parameters

All means of exogenous manifest variables in the RAM model are free parameters automatically unless specified otherwise in the MEAN *list-entries*. Each automatic mean parameter is named by PROC TCALIS with the _Add prefix and appended with an unique integer. If you want any of these means be fixed values or constrained via name references, you must specify them explicitly in the MEAN *list-entries*.

Modifying a RAM 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 that the reference model (or base model) is also a RAM model. The reference model will be 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 RAM modeling language to make modifications on the model specification. The syntax for modifications is the same as the ordinary RAM modeling language (see the section "RAM Statement" on page 6797). The syntax for respecifying/modifying the RAM model is as follows.

RAM *list-entry* < , *list-entry* . . . > **;**

where *list-entry* represents:
parameter-type variable1 < *direction* > < *variable2* > *parameter-spec*
and *direction* represents one of the following: –>, >, <–, <

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 to 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:

```
proc tcalis;
   group 1 / data=d1;
   group 2 / data=d2;
   model 1 / group=1;
      ram
         path V1 <- F1   1.,
         path V2 <- F1   load1,
         path V3 <- F1   load2,
         path F1 <- V4   b1,
         path F1 <- V5   b2
         path F1 <- V6   b3
         pvar E1         ve1,
         pvar E2         ve2,
         pvar E3         ve3,
         pvar F1         vd1,
         pvar V4         phi4,
         pvar V5         phi5,
         pvar V6         phi6,
         pcov V1 V2      cve12;
   model 2 / group=2;
      refmodel 1;
      ram
         path V3 <- F1   load1,
         pcov V1 V2      .    ,
         pcov V2 V3      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 TCALIS:

- 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.
- Deletion: The specification of partial covariance between V1 and V2 in the old model is not copied into the new model, as indicated by the missing value '.' in the specification.
- Replacement: The new path V3 <– F1 replaces the same path in the old model with another parameter for the path coefficient. Thus, in the new model paths V3 <– F1 and V2 <– F1 are constrained to have the same path coefficient parameter load1.

In the RAM model, partial variances or variances of all variables, means of exogenous mainfest variables, and the covariances among exogenous mainfest variables are automatically free parameters unless you specify otherwise in the *list-entries* of the RAM statement. See the sections "Automatic Variance and Partial Variance Parameters" on page 6801, "Automatic Covariance Parameters" on page 6801, and "Automatic Mean Parameters" on page 6801 for more details. This is also true for RAM models specified under the REFMODEL statement. In the reference model, parameters generated by PROC TCALIS are named with the _Add prefix and appended with integer suffixes. These automatically generated parameters in the 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 automatic 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 reference model. If you want any of these parameters be constrained across the models, you must specify them explicitly in the *list-entries* of the RAM statement of the reference model so that these specifications are transferred to the new model via the REFMODEL statement.

REFMODEL Statement

REFMODEL | BASEMODEL *model_number < / options >* **;**

where *model_number* represents a model number between 1 and 9999, inclusively,
and *options* are for renaming parameters.

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 put the *model_number* of the model you are making reference to. The reference model must be well-defined in the same PROC TCALIS run. In the *options*, you can rename all the parameters in the reference model by adding prefix or suffix so that the current model has a new set of parameters.

The following options are available in the REFMODEL statement:

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 qq is a parameter in the reference model for a current model with model number 3, then this option creates qq_mdl3 as a new parameter name.

PARM_PREFIX=*prefix*

inserts to all parameter names in the reference model with the *prefix* provided. For example, if qq is a parameter in the reference model for a current model, then **PARM_PREFIX=pre_** creates pre_qq as a new parameter name.

PARM_SUFFIX=*suffix*

appends to all parameter names in the reference model with the *suffix* provided. For example, if qq is a parameter in the reference model for a current model, then **PARM_SUFFIX=_suf** creates qq_suf as a new parameter name.

Instead of renaming all parameters, you can also rename parameters individually by using the RENAMEPARM statement within the scope of the 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 STD 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 STD 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;
   std 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 410 in Chapter 18, "Shared Concepts and Topics."

SIMTEST Statement

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 SIMTEST 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 θ 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 SIMTEST 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 θ_1, θ_2, θ_3, and θ_4 are parameters in the model and you want to test the equality of θ_1 and θ_2 and the equality of θ_3 and θ_4 simultaneously, you can use the following statements:

```
simtest
   Equality_test = [t1_t2_diff t3_t4_diff];
t1_t2_diff    = theta1 - theta2;
t3_t4_diff    = theta3 - theta4;
```

In the SIMTEST 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 6809 for testing parametric functions individually.

STD Statement

STD *assignment < , assignment . . . >* **;**

where *assignment* represents:
variables = parameter-spec

The STD statement defines the variance parameters in the LINEQS models. STD, instead of VAR, is used as the statement name for defining variance parameters because it avoids conflicts with the VAR statement for specifying variables for analysis. You can specify no more than one STD statement with each LINEQS statement.

In each *assignment* of the STD statement, you specify the variance parameters of the *variables*. The *variables* are specified on the left-hand side of the equal sign of each *assignment*, while the parameter in the *parameter-spec* list on the right-hand side of the equal sign might be one of the following types:

- free or constrained parameters without starting values

  Names in the *parameter-spec* list denote parameters to estimate. They might be free or constrained parameters.

- free or constrained parameters with starting values

  Names followed by numbers inside a pair of parentheses in the *parameter-spec* list denote free or constrained parameters with starting values given by the parenthesized numbers.

- fixed parameter values

  Fixed constants not inside a pair of parentheses in the *parameter-spec* list denote fixed parameter values.

The *assignments* in the STD statement must be separated by commas. The *variables* listed on the left-hand side of the equal sign should contain only names of exogenous variables—that is, variance parameters in the LINEQS model are defined only for exogenous mainfest or latent variables, errors, and disturbances.

If the *parameter-spec* list is longer than the *variables* list, the *parameter-spec* list is shortened to the length of the *variables* list. If the *parameter-spec* is shorter than the *variables* list, the *parameter-spec* list is filled with repetitions of the last item in the list.

The *parameter-spec* list can contain prefix names. A prefix name is a short name called “root” followed by two underscores, ‘__’. A prefix name without a root (that is, as in ‘__’) has its root replaced with parm. Whenever a prefix name is encountered, the TCALIS procedure generates a parameter name by appending a unique integer to the root of the prefix name. 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 explicitly defined parameter names.

To illustrate various specification in the STD statement, consider the following statement:

```
std E1-E6 = vp1 GEN__ (1. 2.) GEN__ 4. vp5 ;
```

You specify the variance of E1 as a free parameter vp1 with a starting value at 1. The variance of E2 is a parameter named GEN1, with a starting value at 2. The variance of E3 is a parameter named GEN2, without a starting value provided. The variance of E4 is fixed at 4. The variance of E5 is parameter vp5, without a starting value given. Finally, the variance of E6 is denoted also by parameter vp5 because the *parameter-spec* list is shorter than the *variables* list so that the last member in the *parameter-spec* list is repeated.

Automatic Variance Parameters among Exogenous Variables

If the names of any exogenous variables (manifest, latent, or errors) in the LINEQS model are not included in the STD statement, the variances of those exogenous variables are assumed to be parameters to estimate. These parameters will be named automatically with the prefix _Add and appended with a unique integer in each name generation.

Modifying a Parameter Specification from a Reference Model

If you define a new LINEQS model by using a reference (or base) model in the REFMODEL statement, in some situations you might want to modify some parameter specifications from the STD statement of the reference model before transferring the specifications to the new model. If you want to change a particular STD parameter specification from the reference model, you can simply put the corresponding parameter location with a new parameter specification in the new model. If you want to delete a particular parameter location from the reference model, you can put that parameter location with the missing value '.' in the new model. For example, suppose you are defining a new LINEQS model by using the REFMODEL statement and the variance of variable V1 is defined as a fixed or free parameter in the reference model. If you do not want this parameter specification to be copied into your new model, you can use the following specification in the new model:

```
std V1 = .;
```

Notice that the missing value syntax is valid only when you use the REFMODEL statement. For a more detailed example of the LINEQS model respecification (see the section "Modifying a LINEQS Model from a Reference Model" on page 6761).

In the LINEQS model, all variances among exogenous variables are free parameters to estimate unless you specify otherwise in the STD statement (see the section "Automatic Variance Parameters among Exogenous Variables" on page 6808). This is also true for LINEQS models specified under the REFMODEL statement. In the reference model, variance parameters generated by PROC TCALIS are named with the _Add prefix and appended with integer suffixes. These parameter specifications in the 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 options, and the model respecifications are the unspecified variances of exogenous variables generated as free parameters named with the _Add prefix and integer suffixes. In this way, the variance parameters in the new model are not constrained to be the same as the corresponding parameters in the reference model. If you want any of these variance parameters constrained across the models, you must specify them explicitly in the STD statement of the reference model so that the variance specification is transferred to the new model.

STRUCTEQ Statement

STRUCTEQ *variables* < / *label*> **;**

where *label* represents:
LABEL | NAME = *name*

The STRUCTEQ statement functions exactly the same as the DETERM statement.

TESTFUNC Statement

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 θ 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 θ_1, θ_2, and θ_3 are parameters in the model, and you want to test whether θ_1 and θ_2 are the same and whether θ_3 is the same as the average of θ_1 and θ_2, you can use the following statements:

```
testfunc   t1_t2_diff t3_t1t2_diff;
t1_t2_diff   = theta1 - theta2;
t3_t1t2_diff = theta3 - (theta1 + theta2)/2;
```

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 θ_1 and θ_2, and t3_t1t2_diff represents the difference between θ_3 and the average of θ_1 and θ_2.

See the SIMTEST statement if you want to test several null hypotheses simultaneously.

VAR Statement

VAR *variables* **;**

The VAR statement lists the numeric variables to be analyzed. It is one of the subsidiary group specification statements. You can use the VAR statement no more than once within the scope of each GROUP or the PROC TCALIS statement.

Variables in the VAR statement of the associated group are included in the corresponding model if the model is one of the following types: LINEQS, PATH, or RAM. Any VAR variables that are not mentioned in the model specification by using the main and subsidiary model specification statements are treated as exogenous variables in the model. This also means that for these models the use of the VAR statement in the associated groups is not necessary when all observed variables intended for analysis are already mentioned in the main and subsidiary model specification statements. Given the model specification, PROC TCALIS finds the set of observed variables for analysis automatically. You need the VAR statement only when you want to force the inclusion of those observed variables not mentioned in the model specification.

For the FACTOR model, the VAR variables of the associated groups are included in the model only for an exploratory factor analysis. The VAR statement is ignored for a confirmatory factor analysis, in which the observed variables can be defined only through the *factor-variable-relations* in the FACTOR statement.

The VAR statement of the associated group is ignored when the corresponding model is of either the LISMOD or MSTRUCT type. The observed variables for these two types of models must be defined in the variable lists of the LISMOD or MSTRUCT statement. For the LISMOD model, each observed variable for analysis is defined in either the YVAR= or XVAR= option of the LISMOD statement. For the MSTRUCT model, all observed variables for analysis are defined in the VAR= option of the MSTRUCT statement.

The VAR statement should not be confused with the PARAMETERS statement. In the PARAMETERS statement, you specify additional *parameters* in the model. Parameters are population quantities that characterize the functional relationships, variations, or covariation among variables. Unfortunately, parameters are sometimes referred to as *variables* in the optimization context. You have to make sure that all variables specified in the VAR statement correspond to the variables in the input data set, while the parameters specified in the PARAMETERS statement are population quantities that characterize distributions of the variables and their relationships.

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 TCALIS 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: TCALIS Procedure

Input Data Sets

You can use four different kinds of input data sets in the TCALIS 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 TCALIS 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 TCALIS analysis. See the section "OUTMODEL= SAS-data-set" on page 6819 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.

DATA= SAS-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 TCALIS 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 TCALIS.

For example, assuming that PROC TCALIS 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:

```
* create TYPE=COV data set;
proc corr cov nocorr data=raw outp=cov(type=cov);
run;
* analysis using correlations;
proc tcalis corr data=cov inmodel=model;
run;
* analysis using covariances;
proc tcalis 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 TCALIS 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 TCALIS, 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 TCALIS 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 TCALIS results as the previous example:

```
* using correlations;
proc tcalis corr data=raw outstat=cov inmodel=model;
run;
* using covariances;
proc tcalis 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 TCALIS analyzes the recomputed covariance matrix even when a TYPE=CORR data set is provided, as shown in the following statements:

```
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 tcalis inmodel=model;
run;
```

INEST= SAS-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 TCALIS 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 TCALIS recognizes observations with the following _TYPE_ specifications.

PARMS	specifies initial values for parameters that are defined in the model statements of PROC TCALIS. 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 TCALIS, 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.
LE | <= | <	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 TCALIS 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 TCALIS. When you are using this OUTEST= data set without changes as an INEST= data set, PROC TCALIS adds the constraints from the data set to the constraints specified by a BOUNDS and LINCON statement. Although PROC TCALIS automatically eliminates multiple identical constraints, you should avoid specifying the same constraint a second time.

INMODEL= SAS-data-set

This data set is usually created in a previous run of PROC TCALIS. 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= SAS-data-set" on page 6819 for more details about the INMODEL= data set.

INWGT= SAS-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 $w_{ij,kl}$. 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 TCALIS 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= SAS-data-set" on page 6828 for more details about the INWGT= data set.

Output Data Sets

OUTEST= SAS-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, 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 TCALIS for another analysis with either PROC TCALIS 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 88.2 are available in the OUTEST= data set, depending on the request.

Table 88.2 _TYPE_ Observations in the OUTEST= Data Set

TYPE	Description
ACTBC	if there are active boundary constraints at the solution x^*, three observations indicate which of the parameters are actively constrained, as follows:

NAME	Description
GE	indicates the active lower bounds
LE	indicates the active upper bounds
EQ	indicates the active masks

TYPE	Description
COV	contains the approximate covariance matrix of the parameter estimates; used in computing the approximate standard errors.
COVRANK	contains the rank of the covariance matrix of the parameter estimates.
CRPJ_LF	contains the Hessian matrix of the Lagrange function (based on CRPJAC).
CRPJAC	contains the approximate Hessian matrix used in the optimization process. This is the inverse of the information matrix.
EQ	if linear constraints are used, this observation contains the ith linear constraint $\sum_j a_{ij} x_j = b_i$. The parameter variables contain the coefficients a_{ij}, $j = 1, \ldots, n$, the _RHS_ variable contains b_i, and _NAME_=ACTLC or _NAME_=LDACTLC.
GE	if linear constraints are used, this observation contains the ith 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 _RHS_ variable contains b_i. If the constraint i is active at the solution x^*, then _NAME_=ACTLC or _NAME_=LDACTLC.
GRAD	contains the gradient of the estimates.
GRAD_LF	contains the gradient of the Lagrange function. The _RHS_ variable contains the value of the Lagrange function.
HESSIAN	contains the Hessian matrix.
HESS_LF	contains the Hessian matrix of the Lagrange function (based on HESSIAN).
INFORMAT	contains the information matrix of the parameter estimates (only for METHOD=ML, METHOD=GLS, or METHOD=WLS).
INITGRAD	contains the gradient of the starting estimates.
INITIAL	contains the starting values of the parameter estimates.

Table 88.2 *continued*

TYPE	Description
JACNLC	contains the Jacobian of the nonlinear constraints evaluated at the final estimates.
LAGM BC	contains Lagrange multipliers for masks and active boundary constraints.

NAME	Description
GE	indicates the active lower bounds
LE	indicates the active upper bounds
EQ	indicates the active masks

TYPE	Description
LAGM LC	contains Lagrange multipliers for linear equality and active inequality constraints in pairs of observations containing the constraint number and the value of the Lagrange multiplier.

NAME	Description
LEC_NUM	number of the linear equality constraint
LEC_VAL	corresponding Lagrange multiplier value
LIC_NUM	number of the linear inequality constraint
LIC_VAL	corresponding Lagrange multiplier value

TYPE	Description
LAGM NLC	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.

NAME	Description
NLEC_NUM	number of the nonlinear equality constraint
NLEC_VAL	corresponding Lagrange multiplier value
NLIC_NUM	number of the linear inequality constraint
NLIC_VAL	corresponding Lagrange multiplier value

TYPE	Description
LE	if linear constraints are used, this observation contains the ith 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 | LB	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.

Table 88.2 *continued*

TYPE	Description
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 NLC_GE NLC_LE	contains values and residuals of nonlinear constraints. The _NAME_ variable is described as follows: **_NAME_** — **Description** NLC — inactive nonlinear constraint NLCACT — linear independent active nonlinear constraint NLCACTLD — linear dependent active nonlinear constraint
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.
NLDACTLC	contains the number of active linear constraints at the solution x^* that are recognized as linearly dependent. The _RHS_ variable contains a missing value, and the _NAME_ variable is blank.
NOBS	contains the number of observations.
PARMS	contains the final parameter estimates. The _RHS_ variable contains the value of the objective function.
PCRPJ_LF	contains the projected Hessian matrix of the Lagrange function (based on CRPJAC).
PHESS_LF	contains the projected Hessian matrix of the Lagrange function (based on HESSIAN).
PROJCRPJ	contains the projected Hessian matrix (based on CRPJAC).
PROJGRAD	if linear constraints are used in the estimation, this observation contains the $n - n_{act}$ values of the projected gradient $g_Z = Z'g$ in the variables corresponding to the first $n - n_{act}$ parameters. The _RHS_ variable contains a missing value, and the _NAME_ variable is blank.
PROJHESS	contains the projected Hessian matrix (based on HESSIAN).
STDERR	contains approximate standard errors (only for METHOD=ML, METHOD=GLS, or METHOD=WLS).

Table 88.2 *continued*

TYPE	Description
TERMINAT	the _NAME_ variable contains the name of the termination criterion.
UPPERBD | UB	if boundary constraints are used, this observation contains the upper bounds. Those parameters not subjected to upper bounds contain missing values. The _RHS_ variable contains a missing value, and the _NAME_ variable is blank.

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= SAS-data-set

The OUTMODEL= (or OUTRAM=) data set is of TYPE=CALISMDL and contains the model specification and the computed parameter estimates and 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 TCALIS.

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 indicating the type of model specification
- a character variable _NAME_, which indicates the model type, parameter name, or variable name
- 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

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_, _VAR1_, and _VAR2_ variables for each value of the _TYPE_ variable, given the type of the model.

FACTOR Models

Value of _TYPE_	Specification Type	_NAME_	_VAR1_	_VAR2_
MDLTYPE	model type	model type		
ADDPVAR	added (partial) variance	parameter	variable	
COV	covariance	parameter	first variable	second variable
LOADING	factor loading	parameter	manifest variable	factor variable
MEAN	mean or intercept	parameter	variable	
PVAR	(partial) variance	parameter	variable	

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. Other observations specify the parameters in the model. The _NAME_ values for these observations are the parameter names. Any observation with a _TYPE_ value of COV or LOADING is for specifying a parameter associated with two variables. The _VAR1_ and _VAR2_ values of such an observation indicate the variables involved. Any observation with a _TYPE_ value of ADDPVAR, MEAN, or PVAR is for specifying a parameter associated with a single variable. The _VAR1_ value of such an observation indicates the variable involved, while the _VAR2_ value is not used.

LINEQS Models

Value of _TYPE_	Specification Type	_NAME_	_VAR1_	_VAR2_
MDLTYPE	model type	model type		
ADDCOV	added covariance	parameter	first variable	second variable
ADDMEAN	added mean	parameter	variable	
ADDSTD	added variance	parameter	variable	
COV	covariance	parameter	first variable	second variable
EQUATION	path coefficient	parameter	outcome variable	predictor variable
MEAN	mean	parameter	variable	
STD	variance	parameter	variable	

The value of the _NAME_ variable is LINEQS for the _TYPE_=MDLTYPE observation. Other observations specify the parameters in the model. The _NAME_ values for these observations are the parameter names. Any observation with a _TYPE_ value of ADDCOV, COV, or EQUATION is for specifying a parameter associated with two variables. The _VAR1_ and _VAR2_ values of such an observation indicate the variables involved. Any observation with a _TYPE_ value of ADDMEAN, ADDSTD, MEAN, or STD is for specifying a parameter associated with a single variable. The _VAR1_ value of such an observation indicates the variables involved, while the _VAR2_ value is not used.

LISMOD Models

Value of _TYPE_	Specification Type	_NAME_	_VAR1_	_VAR2_
MDLTYPE	model type	model type		
XVAR	x-variable	variable	variable number	
YVAR	y-variable	variable	variable number	
ETAVAR	eta-variable	variable	variable number	
XIVAR	xi-variable	variable	variable number	
ADDKAPPA	added _KAPPA_ entry	parameter	row number	
ADDPHI	added _PHI_ entry	parameter	row number	column number
ADDPSI	added _PSI_ entry	parameter	row number	column number
ADDTHETAX	added _THETAX_ entry	parameter	row number	column number
ADDTHETAY	added _THETAY_ entry	parameter	row number	column number
ALPHA	_ALPHA_ entry	parameter	row number	
BETA	_BETA_ entry	parameter	row number	column number
GAMMA	_BETA_ entry	parameter	row number	column number
KAPPA	_KAPPA_ entry	parameter	row number	
LAMBDAX	_LAMBDAX_ entry	parameter	row number	column number
LAMBDAY	_LAMBDAY_ entry	parameter	row number	column number
NUX	_NUX_ entry	parameter	row number	
NUY	_NUY_ entry	parameter	row number	
PHI	_PHI_ entry	parameter	row number	column number
PSI	_PSI_ entry	parameter	row number	column number
THETAX	_THETAX_ entry	parameter	row number	column number
THETAY	_THETAY_ entry	parameter	row number	column number

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 on the respective lists in the model. The _NAME_ variable of these observations stores the names of the variables, while the _VAR1_ variable stores the variable numbers on 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 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

Value of _TYPE_	Specification Type	_NAME_	_VAR1_	_VAR2_
MDLTYPE	model type	model type		
VAR	variable	variable	variable number	
COVMAT	covariance	parameter	row number	column number
MEANVEC	mean	parameter	row number	

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, while the _VAR1_ variable stores the variable numbers on 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 the value of _TYPE_ is 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

Value of _TYPE_	Specification Type	_NAME_	_VAR1_	_VAR2_
MDLTYPE	model type	model type		
ADDPCOV	added (partial) covariance	parameter	first variable	second variable
ADDMEAN	added mean	parameter	variable	
ADDPVAR	added (partial) variance	parameter	variable	
LEFT	path coefficient	parameter	outcome variable	predictor variable
MEAN	mean	parameter	variable	
PCOV	(partial) covariance	parameter	first variable	second variable
PVAR	(partial) variance	parameter	variable	
RIGHT	path coefficient	parameter	predictor variable	outcome variable

The value of the _NAME_ variable is PATH for the _TYPE_=MDLTYPE observation. Other observations specify the parameters in the model. The _NAME_ values for these observations are the parameter names. Any observation with a _TYPE_ value of ADDPCOV, LEFT, PCOV, or RIGHT is for specifying a parameter associated with two variables. The _VAR1_ and _VAR2_ values of such an observation indicate the variables involved. Any observation with a _TYPE_ value of ADDMEAN, ADDPVAR, ADDSTD, MEAN, or PVAR is for specifying a parameter associated with a single variable. The _VAR1_ value of such an observation indicates the variable involved, while the _VAR2_ value is not used.

RAM Models

Value of _TYPE_	Specification Type	_NAME_	_VAR1_	_VAR2_
MDLTYPE	model type	model type		
_A_	matrix _A_ entry	parameter	first variable	second variable
_P_	matrix _P_ entry	parameter	first variable	second variable
_W_	vector _W_ entry	parameter	variable	
ADDMEAN	added mean	parameter	variable	
ADDPCOV	added (partial) covariance	parameter	first variable	second variable
ADDPVAR	added (partial) variance	parameter	variable	
INTERCEP	intercept	parameter	variable	
LEFT	path coefficient	parameter	outcome variable	predictor variable
MEAN	mean	parameter	variable	
PCOV	(partial) covariance	parameter	first variable	second variable
PVAR	(partial) variance	parameter	variable	
RIGHT	path coefficient	parameter	predictor variable	outcome variable

The value of the _NAME_ variable is RAM for the _TYPE_=MDLTYPE observation. Other observations specify the parameters in the model. The _NAME_ values for these observations are for the parameter names. Any observation with a _TYPE_ value of _A_, _P_, ADDPCOV, LEFT, PCOV, or RIGHT is for specifying a parameter associated with two variables. The _VAR1_ and _VAR2_ values of such an observation indicate the variables involved. Any observation with a _TYPE_ value of _W_, ADDMEAN, ADDPVAR, INTERCEP, MEAN, or PVAR is for specifying a parameter associated with a single variable. The _VAR1_ value of such an observation indicates the variable involved, while the _VAR2_ value is not used.

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 those observations with _TYPE_ values prefixed by "ADD" (for example, ADDCOV, ADDMEAN, or ADDSTD). These observations represent parameter locations added automatically by PROC TCALIS in a previous run. Because the context of the new analyses might be different, these observations for added parameter locations might not be suitable in the new runs any more. Hence, these observations are *not* read as input model information. Fortunately, after reading the INMODEL= specification in the new analyses, TCALIS analyzes the new model specification again. It then adds an appropriate set of parameters in the new context when necessary.

OUTSTAT= SAS-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 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 6733)

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

Value of _TYPE_	**Contents**
CORR	correlations analyzed
COV	covariances analyzed
KURTOSIS	univariate kurtosis
MEAN	means
N	sample size
SKEWNESS	univariate skewness
STD	standard deviations
SUMWGT	sum of weights (if the WEIGHT statement is used)

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

value of _TYPE_	Contents
METHOD=NONE	
ASRES	asymptotically standardized residuals
NRES	normalized residuals
PRED	predicted moments
RES	raw residuals
SRES	variance standardized residuals
METHOD=DWLS	
DWLSASRS	DWLS asymptotically standardized residuals
DWLSNRES	DWLS normalized residuals
DWLSPRED	DWLS predicted moments
DWLSRES	DWLS raw residuals
DWLSSRES	DWLS variance standardized residuals
METHOD=GLS	
GLSASRES	GLS asymptotically standardized residuals
GLSNRES	GLS normalized residuals
GLSPRED	GLS predicted moments
GLSRES	GLS raw residuals
GLSSRES	GLS variance standardized residuals
METHOD=ML	
MAXASRES	ML asymptotically standardized residuals
MAXNRES	ML normalized residuals
MAXPRED	ML predicted moments
MAXRES	ML raw residuals
MAXSRES	ML variance standardized residuals
METHOD=ULS	
ULSASRES	ULS asymptotically standardized residuals
ULSNRES	ULS normalized residuals
ULSPRED	ULS predicted moments
ULSRES	ULS raw residuals
ULSSRES	ULS variance standardized residuals
METHOD=WLS	
WLSASRES	WLS asymptotically standardized residuals
WLSNRES	WLS normalized residuals
WLSPRED	WLS predicted moments
WLSRES	WLS raw residuals
WLSSRES	WLS variance standardized residuals

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

Value of _TYPE_	Contents
Unstandardized Effects	
DEFFECT	direct effects
DEFF_SE	standard error estimates for direct effects
IEFFECT	indirect effects
IEFF_SE	standard error estimates for indirect effects
TEFFECT	total effects
TEFF_SE	standard error estimates for total effects
Standardized Effects	
SDEFF	standardized direct effects
SDEFF_SE	standard error estimates for standardized direct effects
SIEFF	standardized indirect effects
SIEFF_SE	standard error estimates for standardized indirect effects
STEFF	standardized total effects
STEFF_SE	standard error estimates for standardized total effects
Latent Variable Scores Coefficients	
LSSCORE	latent variable (or factor) scores regression coefficients for ULS method
SCORE	latent variable (or factor) scores regression coefficients other than ULS method

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

Value of _TYPE_	Contents
ERRVAR	error variances
FCORR	factor correlations or covariances
LOADINGS	unrotated factor loadings
RFCORR	rotated factor correlations or covariances
RLOADING	rotated factor loadings
ROTMAT	rotation matrix
STDFCORR	standardized factor correlations
STDLOAD	standardized factor loadings
STDRFCOR	standardized rotated factor correlations or covariances
STDRLOAD	standardized rotated factor loadings

For the _TYPE_=ERRVAR observation, the _NAME_ variable is blank. For all other observations, the _NAME_ variable contains factor names.

OUTWGT= SAS-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 $varnam_i$ be the name of the ith variable in the analysis. In this case, the OUTWGT= data set contains n observations with the variables shown in the following table:

Variable	Contents
TYPE	WEIGHT (character)
NAME	name of variable $varnam_i$ (character)
$varnam_1$	weight w_{i1} for variable $varnam_1$ (numeric)
$\vdots$	$\vdots$
$varnam_n$	weight w_{in} for variable $varnam_n$ (numeric)

For weighted least squares estimation, the weight matrix **W** of the OUTWGT= data set contains only the nonredundant elements $w_{ij,kl}$. In this case, the OUTWGT= data set contains $n(n+1)(2n+1)/6$ observations with the variables shown in the following table:

Variable	Contents
TYPE	WEIGHT (character)
NAME	name of variable $varnam_i$ (character)
NAM2	name of variable $varnam_j$ (character)
NAM3	name of variable $varnam_k$ (character)
$varnam_1$	weight $w_{ij,k1}$ for variable $varnam_1$ (numeric)
$\vdots$	$\vdots$
$varnam_n$	weight $w_{ij,kn}$ for variable $varnam_n$ (numeric)

Symmetric redundant elements are set to missing values.

OUTFIT= SAS-data-set

You can create an OUTFIT= data set that is of TYPE=CALISFIT and 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 character variable _INDEX_ for the names of the fit indices
- a numerical variable _VALUE_ for the numerical values of the fit indices
- a character variable _PRINT_ for the character-formatted fit index values.

The possible values of _TYPE_ are:

ModelInfo:	basic modeling statistics and information
Absolute:	standalone 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 _INDEX_ When _TYPE_=ModelInfo

Value of _INDEX_	Description
N Observations	number of observations use in the analysis
N Variables	number of variables
N Moments	number of mean or covariance elements
N Parameters	number of parameters
N Active Constraints	number of active constraints in the solution
Independence Model Chi-Square	chi-square value for the independence model
Independence Model Chi-Square DF	the degrees of freedom for the independence model chi-square

Possible Values of _INDEX_ When _TYPE_=Absolute

Value of _INDEX_	Description
Fit Function	fit function value
Percent Contribution to Chi-Square	percentage contribution to the chi-square value
Chi-Square	model chi-square value
Chi-Square DF	the degrees of freedom for model chi-square test
Pr > Chi-Square	the probability of obtaining a larger chi-square than the observed value
Elliptic Corrected Chi-Square	the elliptic-corrected chi-square value
Pr > Elliptic Corr. Chi-Square	the probability of obtaining a larger elliptic-corrected chi-square value
Z-test of Wilson and Hilferty	Z-test of Wilson and Hilferty
Hoelter Critical N	the N value that will make a significant chi-square when multiplied to the fit function value
Root Mean Square Residual (RMSR)	root mean square residual
Standardized RMSR (SRMSR)	standardized root mean square residual
Goodness of Fit Index (GFI)	Jöreskog and Sörbom goodness-of-fit index

Possible Values of _INDEX_ When _TYPE_=Parsimony

Value of _INDEX_	Description
Adjusted GFI (AGFI)	goodness-of-fit index adjusted for the degrees of freedom of the model
Parsimonious GFI	Mulaik et al. modification of the GFI
RMSEA Estimate	Steiger and Lind (1980) root mean square error approximation
RMSEA Lower **r**% Confidence Limit	the lower **r**%[1] confidence limit for RMSEA
RMSEA Upper **r**% Confidence Limit	the upper **r**%[1] confidence limit for RMSEA
Probability of Close Fit	Browne and Cudeck (1993) test of close fit
ECVI Estimate	expected cross-validation index
ECVI Lower **r**% Confidence Limit	the lower **r**%[2] confidence limit for ECVI
ECVI Upper **r**% Confidence Limit	the upper **r**%[2] confidence limit for ECVI
Akaike Information Criterion	Akaike information criterion
Bozdogan CAIC	Bozdogan (1987) consistent AIC
Schwarz Bayesian Criterion	Schwarz (1978) Bayesian criterion
McDonald Centrality	McDonald and Hartmann (1992) measure of centrality

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 _INDEX_ When _TYPE_=Incremental

Value of _INDEX_	Description
Bentler Comparative Fit Index	Bentler (1995) comparative fit index
Bentler-Bonett NFI	Bentler and Bonett (1980) normed fit index
Bentler-Bonett Non-normed Index	Bentler and Bonett (1980) non-normed fit index
Bollen Normed Index Rho1	Bollen normed ρ_1
Bollen Non-normed Index Delta2	Bollen non-normed δ_2
James et al. Parsimonious NFI	James, Mulaik, and Brett (1982) parsimonious normed fit index

The LINEQS Model

The LINEQS modeling language is adapted from the EQS 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 in PROC TCALIS. 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 analyzed using the LINEQS modeling language will be called LINEQS models. It is noted that these so-called LINEQS models can also be analyzed by other general modeling languages in PROC TCALIS.

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 sum, 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 intercept specification in LINEQS model equations.

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 ξ, in the order of manifest variables, latent factors, and error variables. The vector of dependent variables is denoted by η, in the order of manifest variables and latent factors. The LINEQS model matrices are defined as follows:

α $(n_d \times 1)$:	intercepts of dependent variables
β $(n_d \times n_d)$:	effects of dependent variables (in columns) on dependent variables (in rows)
γ $(n_d \times n_i)$:	effects of independent variables (in columns) on dependent variables (in rows)
Φ $(n_i \times n_i)$:	covariance matrix of independent variables
ν $(n_i \times 1)$:	means of independent variables

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:

$$\Sigma_a = \begin{pmatrix} (I-\beta)^{-1}\gamma\Phi\gamma'(I-\beta)^{-1\prime} & (I-\beta)^{-1}\gamma\Phi \\ \Phi\gamma'(I-\beta)^{-1\prime} & \Phi \end{pmatrix}$$

The mean vector of all variables $(\eta', \xi')'$ is structured as:

$$\mu_a = \begin{pmatrix} (I-\beta)^{-1}(\alpha + \gamma\nu) \\ \nu \end{pmatrix}$$

As is shown in the structured covariance and mean matrices, the means and covariances of independent variables are direct model parameters in ν and Φ; 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 Σ_a and μ_a. Mathematically, define a selection matrix G of dimensions $n \times (n_d + n_i)$, where n is the number of observed 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 an observed row variable in Σ_a or μ_a. With each row of G selecting a distinct observed variable, the structured covariance matrix of all observed variables is represented by the following:

$$\Sigma = G\Sigma_a G'$$

The structured mean vector of all observed variables is represented by:

$$\mu = G\mu_a$$

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 β Matrix

The diagonal of the β 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 γ Matrix and the Associated Model Restrictions

The columns of the γ matrix refer to the variables in ξ, in the order of mainfest variables, latent factors, and error variables. In the LINEQS model, the following partition of the γ matrix is assumed:

$$\gamma = \begin{pmatrix} \gamma_0 & E \end{pmatrix}$$

where γ_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 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 γ 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 ν Vector and the Associated Model Restrictions

The ν 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 ν vector is assumed:

$$\nu = \begin{pmatrix} \nu_0 \\ 0 \end{pmatrix}$$

where ν_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 Φ matrix

The Φ 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 Φ matrix is assumed:

$$\Phi = \begin{pmatrix} \Phi_{11} & \Phi'_{21} \\ \Phi_{21} & \Phi_{22} \end{pmatrix}$$

where Φ_{11} is an $(n_i - n_d) \times (n_i - n_d)$ covariance matrix for the independent manifest variables and latent factors, Φ_{22} is an $n_d \times n_d$ covariance matrix for the errors, and Φ_{21} is an $n_d \times (n_i - n_d)$ covariance matrix for the errors with other independent variables in the LINEQS model. Because Φ is symmetric, Φ_{11} and Φ_{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 Φ_{21} is a null matrix. However, Φ_{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:

Matrix	Name	Description	Dimensions
Model Matrices			
α	_EQSALPHA_	intercepts of dependent variables	$n_d \times 1$
β	_EQSBETA_	effects of dependent (column) variables on dependent (row) variables	$n_d \times n_d$
γ	_EQSGAMMA_	effects of independent (column) variables on dependent (row) variables	$n_d \times n_i$
ν	_EQSNU_	means of independent variables	$n_i \times 1$
Φ	_EQSPHI_	covariance matrix of independent variables	$n_i \times n_i$
Submatrices			
γ_0	_EQSGAMMA_SUB_	effects of independent variables, excluding errors, on dependent variables	$n_d \times (n_i - n_d)$
ν_0	_EQSNU_SUB_	means of independent variables, excluding errors	$(n_i - n_d) \times 1$
Φ_{11}	_EQSPHI11_	covariance matrix of independent variables, excluding errors	$(n_i - n_d)\times(n_i - n_d)$
Φ_{21}	_EQSPHI21_	covariances of errors with other independent variables	$n_d \times (n_i - n_d)$
Φ_{22}	_EQSPHI22_	covariance matrix of errors	$n_d \times n_d$

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 α vector and the β and γ matrices in the LINEQS statement without using any matrix language.

For example:

$$Y = b_0 + b_1 X_1 + b_2 F_2 + E_1$$

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:

```
   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:

```
   lineqs Y =  b1 X1 + b2 F2 + E1;
```

See the LINEQS statement on page 6758 for the details in syntax.

Because of the LINEQS model restrictions (see the section "Partitions of Some LINEQS Model Matrices and Their Restrictions" on page 6834), 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. This limitation, however, can be resolved by reparameterization in some cases. See Example 88.2.
- 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 STD statement).
- The regression coefficient or effect associated with an error term must be fixed at one (1.0). This is done automatically by omitting any fixed constants or parameters associated with the error terms. Inserting a parameter or a fixed value other than one immediately before an error term is not allowed.

Mean, Variance, and Covariance Parameter Specification

In addition to the intercept and effect parameters 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, STD, 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 6833 for mathematical details.

Default Parameters in the LINEQS Model

Model-restricted values in the LINEQS model include the zero direct effects of any variables on themselves (that is, the diagonal of β matrix contains zeros only), the predetermined effects of error variables (that is, each element of the E submatrix in γ is always either one or zero) and the predetermined means of error variables (that is, ν elements pertaining to error means are always zero). These fixed values are always enforced and cannot be specified differently. All other elements or locations in the LINEQS model matrices can be specified as parameters (free, constrained, or fixed) in the LINEQS and the subsidiary model specification statements, as described in the previous section.

If a location (or an element) in a LINEQS model matrix is not model-restricted and you do not specify a parameter for it, a default parameter will be assumed for this location. There are two types of default parameters. One is automatic free parameters and the other is fixed zeros.

Automatic Free Parameters

In the LINEQS model, automatic free parameters are generated for the variances of all independent variables (manifest variables, latent factors, and latent errors), the covariances among all manifest independent variables (but not latent factors nor errors), and the means of all manifest independent variables if the mean structures are modeled. The name of each automatic free parameter is generated with the _Add prefix and appended with a unique integer.

In terms of LINEQS model matrices, automatic parameters are applied to all diagonal elements of the Φ_{11} submatrix, to all off-diagonal elements of the Φ_{11} submatrix pertaining to the manifest independent variable in rows and in columns, to the diagonal elements of the Φ_{22} submatrix, and to the elements in the ν_0 subvector pertaining to the manifest independent variables in rows.

Default Fixed Zeros

Unspecified locations (elements) of LINEQS model matrices that are neither model-restricted nor automatic free parameters will be fixed at zeros by default.

Rationale of the Default Parameters in the LINEQS Model

To explain the automatic free parameters in the LINEQS model, note that manifest independent variables are explanatory variables in your model. Their means, variances, and covariances are not functions of other parameters in the model. Rather, together with other parameters, they explain the means, variances, and covariances of the dependent variables. Therefore, means, variances, and covariances of manifest independent variables should be saturated as free parameters unless you are also testing a theoretical pattern of these parameters. Otherwise, failing to do this might add unnecessary constraints to your model.

The same argument applies to the latent independent variables. However, because latent variable means, variances, and covariances are not observed, the treatment is different. Only the variances of latent independent variables (factors or errors) are automatic free parameters. The means and covariances among latent independent variables are fixed zeros by default.

You can override the default parameters by manual specification in the STD, COV, and MEAN statements. Manual specification of these parameters becomes necessary when you are testing theoretical covariance or mean patterns or when these parameters are constrained with other parameters in your model. For example, when you test a fixed covariance matrix pattern, you have to manually specify the fixed values for the variances of, and covariances among, exogenous manifest variables in the STD and COV statements.

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 TCALIS 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 it is noted that you can also specify these so-called LISMOD models by other general modeling languages supported in PROC TCALIS.

The following descriptions of LISMOD models are basically the same as that 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.

Structural Equation Model

The structural equation model for latent factors is:

$$\eta = \alpha + \beta\eta + \Gamma\xi + \zeta$$

where:
η is a random vector of n_η endogenous latent factors
ξ is a random vector of n_ξ exogenous latent factors
ζ is a random vector of errors
α is a vector of intercepts
β is a matrix of regression coefficients of η variables on other η variables
Γ is a matrix of regression coefficients of η on ξ

There are some assumptions in the structural equation model. To prevent a random variable in η from regressing directly on itself, the diagonal elements of β are assumed to be zeros. Also, $(I - \beta)^{-1}$ is assumed to be nonsingular, and ζ is uncorrelated with ξ.

The covariance matrix of ζ is denoted by Ψ and its expected value is a null vector. The covariance matrix of ξ is denoted by Φ and its expected value is denoted by κ.

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 will be 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
ϵ is a random vector of errors for y
ν_y is a vector of intercepts for y
Λ_y is a matrix of regression coefficients of y on η

It is assumed that ϵ is uncorrelated with either η or ζ. The covariance matrix of ϵ is denoted by Θ_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
δ is a random vector of errors for x
ν_x is a vector of intercepts for x
Λ_x is a matrix of regression coefficients of x on ξ

It is assumed that δ is uncorrelated with ξ, ϵ, or ζ. The covariance matrix of δ is denoted by Θ_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:

$$\Sigma = \begin{pmatrix} \Lambda_y (I-\beta)^{-1}(\Gamma\Phi\Gamma' + \Psi)(I-\beta')^{-1}\Lambda_y' + \Theta_y & \Lambda_y (I-\beta)^{-1}\Gamma\Phi\Lambda_x' \\ \Lambda_x \Phi\Gamma'(I-\beta')^{-1}\Lambda_y' & \Lambda_x\Phi\Lambda_x' + \Theta_x \end{pmatrix}$$

The mean structures of the manifest variables $(y', x')'$ are expressed as:

$$\mu = \begin{pmatrix} \nu_y + \Lambda_y (I-\beta)^{-1}(\alpha + \Gamma\kappa) \\ \nu_x + \Lambda_x \kappa \end{pmatrix}$$

Model Matrices in the LISMOD Model

The parameters of the LISMOD model are elements in the model matrices, which are summarized as follows:

Matrix	Name	Description	Dimensions	Row Variables	Column Variables
α	_ALPHA_	intercepts for η	$n_\eta \times 1$	η (ETAVAR=)	N/A
β	_BETA_	effects of η on η	$n_\eta \times n_\eta$	η (ETAVAR=)	η (ETAVAR=)
Γ	_GAMMA_	effects of ξ on η	$n_\eta \times n_\xi$	η (ETAVAR=)	ξ (XIVAR=)
Ψ	_PSI_	error covariance matrix for η	$n_\eta \times n_\eta$	η (ETAVAR=)	η (ETAVAR=)
Φ	_PHI_	covariance matrix for ξ	$n_\xi \times n_\xi$	ξ (XIVAR=)	ξ (XIVAR=)
κ	_KAPPA_	mean vector for ξ	$n_\xi \times 1$	ξ (XIVAR=)	N/A
ν_y	_NUY_	intercepts for y	$n_y \times 1$	y (YVAR=)	N/A
Λ_y	_LAMBDAY_	effects of η on y	$n_y \times n_\eta$	y (YVAR=)	η (ETAVAR=)
Θ_y	_THETAY_	error covariance matrix for y	$n_y \times n_y$	y (YVAR=)	y (YVAR=)
ν_x	_NUX_	intercepts for x	$n_x \times 1$	x (XVAR=)	N/A
Λ_x	_LAMBDAX_	effects of ξ on x	$n_x \times n_\xi$	x (XVAR=)	ξ (XIVAR=)
Θ_x	_THETAX_	error covariance matrix for x	$n_x \times n_x$	x (XVAR=)	x (XVAR=)

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 6843 for details. In the 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 are 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 is characterized by 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, η-variables, and the ξ-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_η, and n_ξ. The variables on the YVAR= and XVAR= lists are manifest variables and therefore must be present in the analyzed data set. The variables on 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. Once 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 on the lists are referred to by the row and column variables of the model matrices.

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 only be specified in one MATRIX statement and each MATRIX statement is used for specifying one model matrix.

An Example

In the section "LISMOD Model" on page 6701, the LISMOD modeling language is used to specify the model described in the section "A Structural Equation Example" on page 6696. In the LISMOD statement, you define four lists of variables, as shown in the following statement:

```
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 η-variables, and SES is the only ξ-variable in the model. Manifest variables that are indicators of endogenous latent factors in η are specified in the YVAR= list. In this case, they are the Anomie and Powerless variables, measured in two different time points. Manifest variables that are indicators of exogenous latent factors in ξ 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_ (Λ_x) matrix with two nonzero entries:

```
matrix _LAMBDAX_ [1,1] = 1.0,
                 [2,1] = 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 initial value is specified there.

Another example is shown as follows:

```
   matrix _THETAY_  [1,1] = theta1,
                    [2,2] = theta2,
                    [3,3] = theta1,
                    [4,4] = theta2,
                    [3,1] = theta5,
                    [4,2] = theta5;
```

In this MATRIX statement, the error variances and covariances (that is, the Θ_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_, or _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 TCALIS will analyze 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 will 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= (an XVAR=) list but without an ETAVAR= (a XIVAR=) list, it is assumed that the set of y-variables (x-variables) is equivalent to the η-variables (ξ-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 η (or ξ) are perfectly measured by the manifest variables y (or x). That is, in such a perfect measurement model,

Λ_y (Λ_x) is an identity matrix and Θ_y (Θ_x) and ν_y (ν_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 Λ_y (Λ_x), Θ_y (Θ_x) and ν_y (ν_x) are non-existent 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 will affect the default or automatic parameterization. See the section "Default Parameters in the LISMOD Model" on page 6846 for more details.

By using these three principles, the models and submodels that PROC TCALIS analyzes are summarized in the following table, followed by detailed descriptions of these models and submodels.

Presence of Lists		Description	Model Equations	Non-fixed Model Matrices
Presence of Both x and y Variables				
1	YVAR=, ETAVAR=, XVAR=, XIVAR=	full model	$y = \nu_y + \Lambda_y \eta + \epsilon$ $x = \nu_x + \Lambda_x \xi + \delta$ $\eta = \alpha + \beta\eta + \Gamma\xi + \zeta$	ν_y,Λ_y,Θ_y ν_x,Λ_x,Θ_x,κ,Φ α ,β,Γ,Ψ
2	YVAR=, XVAR=, XIVAR=	full model with $y \equiv \eta$	$x = \nu_x + \Lambda_x \xi + \delta$ $y = \alpha + \beta y + \Gamma\xi + \zeta$	ν_x,Λ_x,Θ_x,κ,Φ α,β,Γ,Ψ
3	YVAR=, ETAVAR=, XVAR=	full model with $x \equiv \xi$	$y = \nu_y + \Lambda_y \eta + \epsilon$ $\eta = \alpha + \beta\eta + \Gamma x + \zeta$	ν_y,Λ_y,Θ_y α ,β,Γ,Ψ,κ,Φ
4	YVAR=, XVAR=	regression ($y \equiv \eta$) ($x \equiv \xi$)	$y = \alpha + \beta y + \Gamma x + \zeta$, or $(I - \beta)^{-1} y = \alpha + \Gamma x + \zeta$	α,β,Γ,Ψ,κ,Φ
Presence of x Variables and Absence of y Variables				
5	XVAR=, XIVAR=	factor model for x	$x = \nu_x + \Lambda_x \xi + \delta$	ν_x,Λ_x,Θ_x,κ,Φ
6	XVAR=	x-structures ($x \equiv \xi$)		κ,Φ
Presence of y Variables and Absence of x Variables				
7	YVAR=, ETAVAR=	factor model for y	$y = \nu_y + \Lambda_y \eta + \epsilon$ $\eta = \alpha + \beta\eta + \zeta$	ν_y,Λ_y,Θ_y α,β,Ψ
8	YVAR=	y-structures ($y \equiv \eta$)	$y = \alpha + \beta y + \zeta$, or $(I - \beta)^{-1} y = \alpha + \zeta$	α,β,Ψ
9	YVAR=, ETAVAR=, XIVAR=	second-order factor model	$y = \nu_y + \Lambda_y \eta + \epsilon$ $\eta = \alpha + \beta\eta + \Gamma\xi + \zeta$	ν_y,Λ_y,Θ_y α,β,Γ,Ψ,κ,Φ
10	YVAR=, XIVAR=	factor model ($y \equiv \eta$)	$y = \alpha + \beta y + \Gamma\xi + \zeta$, or $(I - \beta)^{-1} y = \alpha + \Gamma\xi + \zeta$	α,β,Γ,Ψ,κ,Φ

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. Model 1 is in fact 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 η ($y \equiv \eta$). 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 ξ ($x \equiv \xi$) 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 η ($y \equiv \eta$) and x is equivalent to ξ ($x \equiv \xi$). 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 β 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 Φ and κ 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 ξ.

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 Λ_x representing the factor loading matrix, Θ_x the error covariance matrix, and Φ the factor covariance matrix. If mean structures are modeled, κ represents the factor means and ν_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 κ and Φ, 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 6869 for more details.

Note that because η-variables cannot exist in the absence of y-variables (see one of the three aforementioned principles in deriving submodels), adding the ETAVAR= list alone to these two submodels does not generate new submodels that can be analyzed by PROC TCALIS.

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 β is null, model 7 functions the same as model 5. It becomes a factor model for y-variables, with Λ_y representing the factor loading matrix, Θ_y the error covariance matrix, Ψ the factor covariance matrix, α the factor means, and ν_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 β is null, the mean structures of y are specified in α and the covariance structures are specified in Ψ. 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 in the LISMOD Model" on page 6846 for details about the default parameterization.

Model 9 represents a modified version of second-order factor model for y. It would be a standard second-order factor model when β is null. This is the TYPE 3A submodel in LISREL VI. With β being null, η represents the first-order factors and ξ represents the second-order factors. The first- and second- order factor loading matrices are Λ_y and Γ, respectively.

Model 10 is another form of factor model when β is null, with factors represented by ξ and manifest variables represented by y. However, if β 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 (or submodel), you can specify fixed values or parameters for the elements in the matrix. See the MATRIX statement on page 6776 for the syntax of specification. All other unspecified elements in the model matrices will be set by default. There are two types of default parameters: one is automatic free parameters; the other is fixed zeros.

Automatic Free Parameters

Automatic additions of free parameters are done to ensure the model is properly parameterized. Each automatic parameters is prefixed with _Add and appended with an unique integer. The following rules are used to add parameters in the LISMOD model matrices:

- _THETAX_, _THETAY_, _PSI_, and _PHI_ matrices: If a matrix in this group is defined in a LISMOD submodel or full model, its diagonal elements are automatically represented by free parameters unless they are specified explicitly in the MATRIXstatements. This ensures that these covariance matrices will have nonzero variances.

- _PHI_ matrix: For LISMOD submodels with XVAR= list specified but XIVAR= unspecified, all the off-diagonal elements of the _PHI_ are automatic free parameters. In this case, because ξ and x are equivalent, its covariance matrix Φ is the covariance matrix of **exogenous**

manifest variables. Because exogenous covariances among manifest variables are not predicted as functions of any other parameters in the LISMOD model, they should be saturated by free parameters in the model unless there are theoretical reasons not to do so (for example, when you are testing a fixed pattern of these covariances). Failure to treat covariances among mainfest variables as free parameters is equivalent to fixing these values to constants, which leads to an overly restrictive model.

- _KAPPA_ matrix: For LISMOD submodels with XVAR= list specified but XIVAR= unspecified, all the elements of the _KAPPA_ vector are automatic free parameters if the mean structures are fitted. The reason for this is the same as the previous rule for adding covariance parameters among exogenous manifest variables. Exogenous manifest variables serve as explanatory variables in your model, and their means are not predicted by relationships with other variables. Unless you are also testing hypothesized mean values for the exogenous manifest variables, in general you should allow these means to be free parameters in the LISMOD model.

These three rules for automatic free parameters are in line with the treatments in the LINEQS, RAM, and PATH models. See, for example, the section "Default Parameters in the LINEQS Model" on page 6838 and the section "Rationale of the Default Parameters in the LINEQS Model" on page 6838 for details about the treatments of automatic free parameters.

Default Fixed Zeros

Matrix elements that are unspecified and are not automatic free parameters will be fixed at zeros by default.

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, although it is noted that you can also specify these so-called RAM models by other general modeling languages supported in PROC TCALIS.

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, exogenous/endogenous and latent/manifest distinctions 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 is 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 supported by PROC TCALIS. 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.

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 to show the partitions of the model matrices and certain mathematical properties. See the section "Partitions of the RAM Model Matrices and Some Restrictions" on page 6850 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 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 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 associated with exogenous row and column variables represents covariance between the two exogenous variables. An off-diagonal element in P 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 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 defined in the RAM model matrix A. Elements in W represent either intercepts or means. An element in W associated with an exogenous row variable represents the mean of the variable. An element in W 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 the 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:

$$\Sigma_a = (I - A)^{-1} P (I - A)^{-1\prime}$$

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 Σ_a and μ_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 Σ_a or μ_a. With each row of G selecting a distinct manifest variable, the structured covariance matrix of all manifest variables is expressed as the following:

$$\Sigma = G \Sigma_a G'$$

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 β is an $n_d \times n_d$ matrix for paths or effects from (column) endogenous variables to (row) endogenous variables and γ is an $n_d \times n_i$ matrix for paths or effects from (column) exogenous variables to (row) endogenous variables.

As shown in the matrix partitions, there are four submatrices. The two 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 β are also zeros. This restriction prevents a direct path from any endogenous variable to itself. There are no restrictions on the pattern of γ.

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 application, 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 TCALIS.

Partition of the W Vector

The W vector is partitioned as:

$$W = \begin{pmatrix} \alpha \\ \nu \end{pmatrix}$$

where α is an $n_d \times 1$ vector for intercepts of the endogenous variables and ν 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:

Matrix	Name	Description	Dimensions
Model Matrices			
A	_A_ or _RAMA_	effects of column variables on row variables, or paths from the column variables to the row variables	$n_a \times n_a$
P	_P_ or _RAMP_	(partial) variances and covariances	$n_a \times n_a$
W	_W_ or _RAMW_	intercepts and means	$n_a \times 1$
Submatrices			
β	_RAMBETA_	effects of endogenous variables on endogenous variables	$n_d \times n_d$
γ	_RAMGAMMA_	effects of exogenous variables on endogenous variables	$n_d \times n_i$
P_{11}	_RAMP11_	error variances and covariances for endogenous variables	$n_d \times n_d$
P_{21}	_RAMP21_	covariances between exogenous variables and error terms for endogenous variables	$n_d \times n_i$
P_{22}	_RAMP22_	variances and covariances for exogenous variables	$n_i \times n_i$
α	_RAMALPHA_	intercepts for endogenous variables	$n_d \times 1$
ν	_RAMNU_	means for exogenous variables	$n_i \times 1$

Specification of 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. This is not the case in PROC TCALIS. You no longer need to specify the RAM model parameters as matrix entries. Instead, you specify the model by using the variable names directly. Representing the RAM model matrices is done internally by PROC TCALIS.

To specify the RAM model parameters, you use the RAM statement to list each parameter location. There are three types of parameter listings, which correspond to the elements in the three model matrices. The three types of parameter listings are described in the following.

(1) Specification of Effects or Paths in Model Matrix A

If in your model there is a path from V2 to V1 and the associated effect parameter is named parm1 with 0.5 as the starting value, you can use one of the following list-entries of the RAM statement to specify this parameter location:

```
 _A_       V1     V2   parm1   (0.5),
 path      V1 <-  V2   parm1   (0.5),
 path      V2 ->  V1   parm1   (0.5),
```

(2) Specification of (Partial) Variances and (Partial) Covariances in Model Matrix P

If V2 is an exogenous variable in the RAM model and you want to specify its variance as a parameter named parm2 with 10 as the starting value, you can use one of the following list-entries of the RAM statement:

```
_P_       V2     V2   parm2   (10.),
pvar      V2          parm2   (10.),
```

If V1 is an endogenous variable in the RAM model and you want to specify its partial variance or error variance as a parameter named parm3 with 5 as the starting value, you can use one of the following list-entries of the RAM statement:

```
_P_       V1     V1   parm3   (5.),
pvar      V1          parm3   (5.),
```

Notice that this partial variance specification has the same format as that for the variance parameter of an exogenous variable.

If you want to specify (partial) covariance between two variables V3 and V4 as a parameter named parm4 with 5 as the starting value, you can use one of the following list-entries of the RAM statement:

```
    _P_       V3     V4   parm4   (5.),
    pcov      V3     V4   parm4   (5.),
```

(3) Specification of Means and Intercepts in Model Matrix _W_

Means and intercepts are specified when the mean structures of the model are of interest. If V5 is an exogenous variable and you want to specify its mean as a parameter named parm5 with 11 as the starting value, you can use one of the following list-entries of the RAM statement:

```
_W_       V5    parm5   (11.),
mean      V5    parm5   (11.),
```

If V6 is an endogenous variable and you want to specify its intercept as a parameter named parm6 with 7 as the starting value, you can use one of the following list-entries of the RAM statement:

```
_W_           V6    parm6  (7.),
intercept     V6    parm6  (7.),
```

Note that this specification bears the same format as that for the mean parameter of an exogenous variable.

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 TCALIS to compute it. For example, you can just provide the parameter locations and parameter names as shown in the following specifications:

```
_A_         V1     V2   parm1,
intercept   V6          parm6,
```

Specifying Fixed Parameter Values

If you want to specify a fixed parameter value, you do not need to provide the parameter name. Instead, you just provide the fixed value without parentheses at the end of the list-entry. The following shows examples for specifying fixed parameters:

```
_PCOV_      V3   V4     5.,
mean        V5          11.,
```

A Complete RAM Model Specification Example

Essentially, you can specify all parameters and their locations of the RAM model in the RAM statement. No subsidiary model specification statements are needed in the RAM model—that is, all information regarding the model is specified in the RAM statement. Each parameter location specification is a list-entry in the RAM statement, and these list-entries are separated by commas. For example,

```
   ram
      path      V1 <-  V2   parm1  (.5),
      path      V1 <-  V3   parm2      ,
      pvar      V1          errv   (1.),
      pvar      V2          parm3  (10.),
      pvar      V3          parm4  (10.),
      pcov      V3     V2   parm5  (5.);
```

Notice that although the order of the list-entries is not critical to the model specification, it is the same order that is used in output displays. See the RAM statement on page 6797 for more details about the syntax.

Default Parameters in the RAM Model

The effect of any variable on itself is restricted to zero in the RAM model. In other words, it is invalid to specify:

```
ram
   path     V1  <-  V1  parm;
```

In matrix terms, you cannot have any nonzero diagonal element in the β submatrix. All other elements in the RAM model matrices can be specified as free or fixed parameters.

If an element in a model matrix is not subject to model restrictions and is not specified explicitly, a default parameter will be applied to the location. There are two types of default parameters: one is automatic free parameter; and the other is fixed zero.

Automatic Free Parameters

The set of automatic parameters in the RAM model is essentially derived in the same way as those in the LINEQS model. That is, in the LINEQS model the set of automatic parameters includes the variances of and covariances among manifest exogenous variables, the means of manifest exogenous variables when the mean structures are modeled, and the variances of all exogenous latent variables (factors and errors). All these automatic free parameters are applied similarly to the RAM model.

The only difficulty about this generalization is the definition of error variances in the RAM model. Unlike the LINEQS model, error terms in the RAM model are not named explicitly. The question is what the error variances in the LINEQS model correspond to in the RAM model. As described previously, error variances in the RAM model are recast as the partial variances of the corresponding endogenous variables. As a result, the set of automatic parameters in the RAM model includes the variances or partial variances of all variables—that is, variances when the variables are exogenous, partial variances when the variables are endogenous.

As in the LINEQS model, the reason for automatic parameter generation is to safeguard a proper RAM model specification. See the section "Rationale of the Default Parameters in the LINEQS Model" on page 6838 for a more detailed discussion.

In terms of the RAM model matrices, the set of automatic parameters in the RAM model includes:

- all P matrix diagonal elements, which are either variances of exogenous variables (manifest or latent) or partial variances of endogenous variables (manifest or latent)
- those P matrix off-diagonal elements with their rows and columns corresponding to manifest exogenous variables
- those W matrix elements with their rows corresponding to manifest exogenous variables, provided that the mean structures are modeled

Each of these automatic mean, intercept, variance, partial variance, covariance, and partial covariance parameters is named with the prefix _Add, and appended with an unique integer.

Default Fixed Zeros

All unspecified parameter locations that are neither model-restricted nor automatic free parameters in the RAM model matrices are fixed zeros by default.

Relating the RAM Model to the PATH Model

The mathematical model for the RAM and the PATH modeling languages is the same. As a result, the two modeling languages are also quite similar in their syntax. See the section "Relating the PATH Model to the RAM Model" on page 6868 for details.

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 outputting 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 measured. Each latent variable in the FACTOR model can either be 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 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 summarized as follows:

Matrix F $(p \times n)$: Factor Loading Matrix

The rows of F represent the p manifest variables, while the columns represent the n factors. Each row of F contains the factor loadings of a variable on all factors in the model.

Matrix P $(n \times n)$: Factor Covariance Matrix

The P matrix is a symmetric matrix for the variances of and covariances among the n factors.

Matrix U $(p \times p)$: Error Covariance Matrix

The U matrix represents an $p \times p$ 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 U are fixed zeros in the FACTOR model.

Vector a $(p \times 1)$: Intercepts

If the mean structures are analyzed, vector a represents the intercepts of the manifest variables.

Vector v $(n \times 1)$: Factor Means

If the mean structures are analyzed, vector v represents the means of the factors.

Matrix Representation of the FACTOR Model

Let y be a $p \times 1$ vector of manifest variables, ξ be an $n \times 1$ vector of latent factors, and e be a $p \times 1$ vector of errors. The factor model is written as:

$$y = a + F\xi + e$$

With the model matrix definitions in the previous section, the covariance matrix Σ $(p \times p)$ of manifest variables is structured as:

$$\Sigma = FPF' + U$$

The mean vector μ $(p \times p)$ of manifest variables is structured as:

$$\mu = a + Fv$$

Exploratory Factor Analysis Models

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.

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.

Nonetheless, in terms of initial model fitting, exploratory factor analysis is not different from confirmatory factor analysis. In PROC TCALIS, 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 $p \times (n-1)/2$ 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.
- The mean structures are not modeled. That is, no specification in a or v vector is allowed.

For example, for an analysis with nine variables and three factors, the relevant model matrices of an exploratory FACTOR model have the following patterns:

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

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and

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

where * denotes free parameters in the model matrices.

It is emphasized that most of these patterns are put on the initial factor solutions only. If an initial factor solution is rotated afterward, only the pattern of the error covariance matrix U remains unchanged. That is, it is still a diagonal matrix after rotation. In general, rotating a factor solution will eliminate 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 will 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 will not be an identity matrix and the off-diagonal elements will not be zeros.

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:

$$F = \begin{pmatrix} 1 & 0 & 0 \\ * & 0 & 0 \\ * & 0 & 0 \\ 0 & 1 & 0 \\ 0 & * & 0 \\ 0 & * & 0 \\ 0 & 0 & 1 \\ 0 & 0 & * \\ 0 & 0 & * \end{pmatrix}$$

$$P = \begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \end{pmatrix}$$

and

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

where * denotes free parameters in the model matrices.

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:

Matrix	Name	Description	Dimensions
F	_FACTLOAD_	factor loading matrix	$p \times n$
P	_FACTFCOR_	factor covariance matrix	$n \times n$
U	_FACTERRV_	error covariance matrix	$p \times p$
a	_FACTINTE_	intercepts	$p \times 1$
v	_FACTMEAN_	factor means	$n \times 1$

Specification of the Exploratory Factor Model

Because all initial model matrices of exploratory FACTOR models are predefined in PROC TCALIS, 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, N=.
- options for factor rotation: GAMMA=, NORM=, RCONVERGE=, RITER=, ROTATE=, TAU=.

For example, the following statement requests that three factors are extracted, followed by a varimax rotation of the initial factor solution:

```
factor n=3 rotate=varimax;
```

See the FACTOR statement on page 6744 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:

```
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 parm1 and parm2, respectively. Initial estimates can be set in parentheses after the free parameters. For example, both parm1 and 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 TCALIS can determine appropriate initial estimates for these parameters. See the descriptions about *CFA_spec* in the FACTOR statement on page 6744 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 6795 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 6794 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 6780 for more details about the syntax.

Default Parameters in the FACTOR Model

In the initial exploratory FACTOR model, all fixed and free parameters of the model are prescribed or set by default. These prescribed or default 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. See the section "Exploratory Factor Analysis Models" on page 6857 for more details about the patterns of the exploratory FACTOR model.

In the confirmatory FACTOR model, the error covariance matrix U is restricted to be a diagonal matrix. In other words, covariances of errors are restricted to zeros. Other than that, you can specify any other parameter locations in the all model matrices. These includes all the parameter locations in the factor loading matrix F, the factor covariance matrix P, the diagonal of the error covariance matrix U, the factor mean vector v, and the intercept vector a. If any of these parameter locations are not specified, default parameters are applied. There are two types of default parameters: one is automatic free parameter, and the other is fixed zero.

Automatic Free Parameters

The set of automatic free parameters in the confirmatory FACTOR model are:

- the variances of factors; that is, the diagonal elements of the factor covariance matrix P
- the error variances for the manifest variables; that is, the diagonal elements of the error covariance matrix U

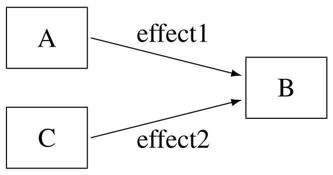
Each automatic free parameter is named with prefix _Add and appended with a unique integer. The reason for automatic parameter generation is to safeguard a proper FACTOR model specification. See the section "Rationale of the Default Parameters in the LINEQS Model" on page 6838 for a more detailed explanation.

Default Fixed Zeros

All unspecified parameter locations that are neither model-imposed specifications nor automatic free parameters in the confirmatory FACTOR model matrices are fixed zeros by default.

The PATH Model

The PATH modeling language is supported in PROC TCALIS 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, given the following simple path diagram:



you can use the PATH statement to specify the paths easily:

```
path    A -> B   effect1,
        C -> B   effect2;
```

In the first entry of the PATH statement, the path A –> B is specified. The associated path coefficient or effect is named effect1. Similarly, in the second entry, the C –> B path is specified with effect2 as the associated effect parameter. In addition to the path coefficients or 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 6696 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 <– or –>) 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 arrow or it serves as an outcome variable in a path at least once. 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 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. Any manifest variable in the PATH model can be endogenous or exogenous. This 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. If error variables are not named, how can one specify the error variance parameters? In the PATH model, error variances are expressed equivalently as partial variances for 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 required 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 is 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 any specific prefix for the latent factor names in the PATH model.

As a general naming convention, you should not use Intercept as either a manifest or latent variable name.

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, in your model there is a path from v2 to v1 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:

```
   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 6791 for more details about the syntax.

(2) Specification of Variances and Partial (Error) Variances

If v2 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 v1 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 TCALIS), you are specifying the variance parameter of the variable. Otherwise, you are specifying the partial or error variance for an endogenous variable.

If you have more than one variance or partial variance parameters 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 v1–v3 are parm1, parm2, and parm3, respectively. Only parm1 is given an initial value at 0.5. The initial values for other parameters are generated by PROC TCALIS.

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 6795 for more details about the syntax.

(3) Specification of Covariances and Partial Covariances

If you want to specify (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 variables 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 covariances.

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.

Because the PCOV statement can only be used 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 6794 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 TCALIS 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 only be used 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 6780 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 TCALIS compute it. For example, you can just provide 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, 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.;
   pvar
        F1 = 1.;
```

A Complete PATH Model Specification Example

To show a more complete PATH model specification, the RAM model example in the section "A Complete RAM Model Specification Example" on page 6853 is translated into the PATH model specification in the following statements:

```
path      v1 <-  v2   parm1  (.5),
          v1 <-  v3   parm2      ;
   pvar   v1 = errv   (1.),
          v2 = parm3  (10.),
          v3 = parm4  (10.);
   pcov   v3 v2 =  parm5  (5.);
```

The PATH model specification is very much the same as the RAM model specification. All paths specified in the RAM model are translated into specification in the PATH statement. All other specification are translated into the PVAR, PCOV, and MEAN statements.

Default Parameters in the PATH Model

The treatment of restricted and default parameters in the PATH model is essentially the same as that of the RAM model (see the section "Default Parameters in the RAM Model" on page 6854).

The PATH model does not allow the specification of the effect of any variable on itself. In other words, it is invalid to specify the following:

```
path      v1  <-  v1  parm;
```

The coefficient for such a path is always zero, meaning that the path should not exist in any PATH model. Other than this restriction, any other parameters supported by the PATH model can be specified in the PATH, PVAR, PCOV, and the MEAN statements. If a parameter location in the PATH model is not specified, a default parameter will be applied to that location. There are two types of default parameters: one is automatic free parameters, and the other is fixed zeros.

Automatic Free Parameters

The set of automatic free parameters in the PATH model is derived essentially in the same way as in the RAM or LINEQS model. That is, automatic free parameters of the PATH model include:

- the variances or partial (or error) variances of all variables, manifest or latent. Consequently, all possible variance or partial variance parameters that can be specified in the PVAR statement are automatic free parameters unless explicitly specified otherwise.
- the means of exogenous manifest variables when the mean structures are modeled. That is, all mean parameters pertaining to exogenous manifest variables that can be specified in the MEAN statement are automatic free parameters unless explicitly specified otherwise.
- the covariances among all exogenous manifest latent variables. That is, all covariance parameters pertaining to all possible pairs of exogenous manifest variables that can be specified in the PCOV statement are automatic free parameters unless explicitly specified otherwise.

The reason for automatic parameter generation is to safeguard a proper PATH model specification. See the section "Rationale of the Default Parameters in the LINEQS Model" on page 6838 for a more detailed explanation.

An automatic parameter name is generated by PROC TCALIS for each of these automatic mean, intercept, variance, partial variance, covariance, and partial covariance parameters. Each automatic parameter name is prefixed with _Add and appended with a unique integer.

Default Fixed Zeros

All unspecified parameter locations that are neither set by model-restricted values nor generated with automatic free parameters in the PATH model are fixed zeros by default.

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 6848 and the section "Summary of Matrices and Submatrices in the RAM Model" on page 6851 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 that of the RAM model. See the section "The RAM Model" on page 6847 for derivations of the model structures.

Because the mathematical model behind the PATH and the RAM modeling languages are essentially the same, it is no wonder that the PATH and the RAM syntax for model specification resemble to each other. That is, all path specifications in the PATH statement translate into the PATH *list-entries* of the RAM statement. All PVAR, PCOV, and MEAN specifications of the PATH model translate into the PVAR, PCOV, and MEAN (or INTERCEPT) *list-entries*, respectively, in the RAM statement.

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 TCALIS 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 following 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 on 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 specified using the MSTRUCT modeling language are called MSTRUCT models.

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:

Matrix	Name	Description	Dimensions
Σ	_COV_ or _MSTRUCTCOV_	structured covariance matrix	$p \times p$
μ	_MEAN_ or _MSTRUCTMEAN_	structured mean vector	$p \times 1$

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 analyzed in your MSTRUCT model by this statement:

```
mstruct VAR= v1 v2 v3 v4 v5;
```

See the MSTRUCT statement on page 6784 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 θ. The dimension of θ depends on your hypothesized model. In the preceding example, θ contains two parameters in ϕ and τ. You can use the MATRIX statement to specify these parameters in the _COV_ matrix:

```
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 short-hand for specifying phi five times, respectively 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,] 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 6776 for details about the syntax.

Default Parameters in the MSTRUCT Model

You can specify either fixed values or parameters (with or without initial values) for the elements in the _COV_ and _MEAN_ model matrices. If some elements are not specified, default parameters are applied. There are two types of default parameters: one is automatic free parameters, and the other is fixed zeros. They are applied in different situations.

Automatic Free Parameters

If no MATRIX statements are used to specify parameters in your model matrices, automatic free parameters are generated for all elements in the _COV_ matrix. For example, if you do not use the MATRIX statement to specify any element in the _COV_ matrix, free parameters are generated for all variances and covariances in the _COV_ matrix in the model. Similarly, free parameters are generated for the _MEAN_ vector when the mean structures are invoked by the MEANSTR option in the PROC TCALIS or MODEL statement. Each automatic free parameter is named with a prefix _Add and appended with a unique integer.

Default Fixed Zeros

If you specify **at least one** parameter (fixed or free) for an element in a model matrix using the MATRIX statement, no automatic free parameters will be generated for that matrix. All unspecified locations in the _COV_ and _MEAN_ model matrices are fixed zeros by default.

How and Why the Default Parameters Are Treated Differently in the MSTRUCT Model

Notice that the default parameter treatment in the MSTRUCT model is different from other types of models. For example, in the LINEQS model variances of and covariances among exogenous manifest variables are all automatic free parameters. These automatic free parameters are generated by PROC TCALIS. In the MSTRUCT model, however, automatic free parameters are either generated for all elements in a model matrix (when nothing is specified for the matrix or when the MATRIX statement is not used for the MSTRUCT model) or not generated at all (when at least one element in the _COV_ matrix is specified).

The reason is that there is no distinction between exogenous (independent) and endogenous (dependent) variables in the MSTRUCT model. Automatic free parameters for variances of, and covariances among, exogenous manifest variables are thus irrelevant.

Because of the particular 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 some diagonal elements, the default zero variances would lead to a nonpositive definite _COV_ model matrix, making the model fitting problematic.

Naming Variables and Parameters

Follow these rules when you name your variables:

- 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:

- 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 “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, . . . etc.
- 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.
- Automatic parameter names are generated by PROC TCALIS for automatic free parameters in the model. The purpose of adding these parameters is to safeguard ill-defined models. Automatic parameter names start with the prefix _Add and appended with unique integers. 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 generated names by PROC TCALIS. The confusions might lead to unintended constraints to your model if the parameter names you use match those generated by PROC TCALIS exactly.

- 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, in PROC TCALIS parameter names and variable names 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 supported by PROC TCALIS. 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:

```
proc tcalis;
   path
      a -> b  c;
   run;
```

PROC TCALIS 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:

```
proc tcalis;
   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 TCALIS in terms of model specification and fitting, it would create unnecessary confusions 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 TCALIS 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.

```
bounds   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 TCALIS 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 Statement

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.

```
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 TCALIS a nonredundant linear equality constraint on independent parameters effectively reduces the number of parameters by one.

NLINCON Statement

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:

```
nlincon
    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 TCALIS procedure. PROC TCALIS 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:**

  $$q \geq a \qquad \text{or} \qquad q \leq b$$

  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:

  $$q = a + x^2 \qquad \text{or} \qquad q = b - x^2$$

  in which the new parameter x is unconstrained.

  For example, inequality constraint $q \geq 7$ can be accomplished by the following statements:

  ```
  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 TCALIS, you can use the following statements:

  ```
  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:**

$$q_1 \le q_2, \quad q_1 \le q_3, \quad \ldots, \quad q_1 \le q_k$$

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 $q_1 \le q_2$, $q_1 \le q_3$, and $q_1 \le q_4$ simultaneously, you can use the following statements:

```
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:**

$$q_1 \le q_2 \le q_3 \le \ldots \le q_k$$

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 $q_1 \le q_2 \le q_3 \le q_4$ simultaneously, you can use the following statements:

```
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 $x_1, x_2, \ldots, x_k$:

  $$\begin{aligned} q_i &= x_i / b_i \qquad (i < k) \\ q_k &= (a - \sum_j^{k-1} x_j) / b_k \end{aligned}$$

  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:

  ```
  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.

Refer to 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 TCALIS 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 TCALIS 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:

```
proc tcalis data=dixon method=gls nobs=180 print mod;
   var    x1-x7;
   lineqs y = e;
   std    e = var;
run;
proc tcalis data=dixon method=gls nobs=180 print mod;
   var    x1-x7;
   lineqs y = g1 x1 + e;
   std    e = var;
run;
proc tcalis data=dixon method=gls nobs=180 print mod;
   var    x1-x7;
   lineqs y = g1 x1 + g6 x6 + e;
   std    e = var;
run;
```

In the first analysis, no independent manifest variables are included in the regression equation for dependent variable y. However, x1–x7 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, x1 is already a predictor in the regression equation, while x2–x7 are treated as potential predictors in the LM tests. In the last analysis, x1 and x6 are predictors in the regression equation, while other x-variables are treated as potential predictors in the LM tests.

Estimation Criteria

The following five estimation methods are available in PROC TCALIS:

- unweighted least squares (ULS)
- generalized least squares (GLS)
- normal-theory maximum likelihood (ML)
- weighted least squares (WLS, ADF)
- diagonally weighted least squares (DWLS)

Default weight matrices $\mathbf{W}$ are computed for GLS, WLS, and DWLS estimation. You can also provide your own weight matrices by using an INWGT= data set.

PROC TCALIS does not implement all estimation methods in the field. As mentioned in the section "Overview: TCALIS Procedure" on page 6673, 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 TCALIS; however, PLS estimates are less efficient in general. When the statistical assumptions of PROC TCALIS 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 TCALIS. For example, in a PROC TCALIS 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 TCALIS.

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, $\mathbf{S}$ the sample $p \times p$ covariance matrix for a sample with size N, $\bar{\boldsymbol{x}}$ the $p \times 1$ vector of sample means, $\boldsymbol{\Sigma}$ the fitted covariance matrix, and $\boldsymbol{\mu}$ the vector of fitted means, the discrepancy function for unweighted least squares (ULS) estimation is:

$$F_{ULS} = 0.5Tr[(\mathbf{S} - \boldsymbol{\Sigma})^2] + (\bar{\boldsymbol{x}} - \boldsymbol{\mu})'(\bar{\boldsymbol{x}} - \boldsymbol{\mu})$$

The discrepancy function for generalized least squares estimation (GLS) is:

$$F_{GLS} = 0.5Tr[(\mathbf{W}^{-1}(\mathbf{S} - \boldsymbol{\Sigma}))^2] + (\bar{\boldsymbol{x}} - \boldsymbol{\mu})'\mathbf{W}^{-1}(\bar{\boldsymbol{x}} - \boldsymbol{\mu})$$

By default, $\mathbf{W} = \mathbf{S}$ 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:

$$F_{ML} = Tr(\mathbf{S}\boldsymbol{\Sigma}^{-1}) - p + ln(|\boldsymbol{\Sigma}|) - ln(|\mathbf{S}|) + (\bar{\boldsymbol{x}} - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\bar{\boldsymbol{x}} - \boldsymbol{\mu})$$

In each of the discrepancy functions, $\mathbf{S}$ and $\bar{\boldsymbol{x}}$ are considered to be given and $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ are functions of model parameter vector $\boldsymbol{\Theta}$. That is:

$$F = F(\boldsymbol{\Sigma}(\boldsymbol{\Theta}), \boldsymbol{\mu}(\boldsymbol{\Theta}); \mathbf{S}, \bar{\boldsymbol{x}})$$

Estimating $\boldsymbol{\Theta}$ by using a particular estimation method amounts to choosing a vector $\boldsymbol{\theta}$ 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:

$$F_{ULS} = 0.5Tr[(\mathbf{S} - \boldsymbol{\Sigma})^2]$$

$$F_{GLS} = 0.5Tr[(\mathbf{W}^{-1}(\mathbf{S} - \boldsymbol{\Sigma}))^2]$$

$$F_{ML} = Tr(\mathbf{S}\boldsymbol{\Sigma}^{-1}) - p + ln(|\boldsymbol{\Sigma}|) - ln(|\mathbf{S}|)$$

If, instead of being a covariance matrix, $\mathbf{S}$ is a correlation matrix in the discrepancy functions, $\boldsymbol{\Sigma}$ would naturally be interpreted as the fitted correlation matrix. Although whether $\mathbf{S}$ 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 $\boldsymbol{\Sigma}$ 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 TCALIS 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 TCALIS results for correlation analysis useful.

PROC TCALIS is primarily developed for the analysis of covariance structures, and hence COVARIANCE is the default option in the procedure. Depending on the nature of research, you can add the mean structures in the analysis. However, for the analysis of correlation structures, you can not add the mean structures for modeling in PROC TCALIS.

WLS and ADF Discrepancy Functions

Another important discrepancy function to consider is the weighted least squares (WLS) function. Let $u = (s, \bar{\boldsymbol{x}})$ be a $p(p+3)/2$ vector containing all nonredundant elements in the sample covariance matrix $\mathbf{S}$ and sample mean vector $\bar{\boldsymbol{x}}$, with $s = vecs(\mathbf{S})$ representing the vector of the

$p(p+1)/2$ lower triangle elements of the symmetric matrix $\mathbf{S}$, 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 $\mathbf{\Sigma}$ and the fitted mean vector $\boldsymbol{\mu}$, with $\sigma = vecs(\mathbf{\Sigma})$ representing the vector of the $p(p+1)/2$ lower triangle elements of the symmetric matrix $\mathbf{\Sigma}$.

The WLS discrepancy function is:

$$F_{WLS} = (u - \eta)'\mathbf{W}^{-1}(u - \eta)$$

where $\mathbf{W}$ is a positive definite symmetric weight matrix with $(p(p+3)/2)$ rows and columns. Because η is a function of model parameter vector $\mathbf{\Theta}$ under the structural model, you can write the WLS function as:

$$F_{WLS} = (u - \eta(\mathbf{\Theta}))'\mathbf{W}^{-1}(u - \eta(\mathbf{\Theta}))$$

Suppose that u converges to $\eta_o = (\sigma_o, \mu_o)$ with increasing sample size, where σ_o and μ_o denote the population covariance matrix and mean vector, respectively. By default, the WLS weight matrix $\mathbf{W}$ in PROC TCALIS is computed from the raw data as a consistent estimate of the asymptotic covariance matrix $\mathbf{\Gamma}$ of $\sqrt{N}(u - \eta_o)$, with $\mathbf{\Gamma}$ partitioned as

$$\Gamma = \begin{pmatrix} \mathbf{\Gamma}_{ss} & \mathbf{\Gamma}'_{\bar{x}s} \\ \mathbf{\Gamma}_{\bar{x}s} & \mathbf{\Gamma}_{\bar{x}\bar{x}} \end{pmatrix}$$

where $\mathbf{\Gamma}_{ss}$ denotes the $(p(p+1)/2) \times (p(p+1)/2)$ asymptotic covariance matrix for $\sqrt{N}(s - \sigma_o)$, $\mathbf{\Gamma}_{\bar{x}\bar{x}}$ denotes the $p \times p$ asymptotic covariance matrix for $\sqrt{N}(\bar{x} - \mu_o)$, and $\mathbf{\Gamma}_{\bar{x}s}$ 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 $\mathbf{W}$ as a consistent estimate of $\mathbf{\Gamma}$, define a similar partition of the weight matrix $\mathbf{W}$ as:

$$\mathbf{W} = \begin{pmatrix} \mathbf{W}_{ss} & \mathbf{W}'_{\bar{x}s} \\ \mathbf{W}_{\bar{x}s} & \mathbf{W}_{\bar{x}\bar{x}} \end{pmatrix}$$

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:

$$\mathbf{t}_{ij} = \frac{1}{N}\sum_{r=1}^{N}(x_{ri} - \bar{x}_i)(x_{rj} - \bar{x}_j)$$

and the sample fourth-order central moment for variables i, j, k, and l as:

$$\mathbf{t}_{ij,kl} = \frac{1}{N}\sum_{r=1}^{N}(x_{ri} - \bar{x}_i)(x_{rj} - \bar{x}_j)(x_{rk} - \bar{x}_k)(x_{rl} - \bar{x}_l)$$

The submatrices in $\mathbf{W}$ are computed by:

$$[\mathbf{W}_{ss}]_{ij,kl} = \mathbf{t}_{ij,kl} - \mathbf{t}_{ij}\mathbf{t}_{kl}$$

$$[\mathbf{W}_{\bar{x}s}]_{i,kl} = \frac{1}{N}\sum_{r=1}^{N}(x_{ri} - \bar{x}_i)(x_{rk} - \bar{x}_k)(x_{rl} - \bar{x}_l)$$

$$[\mathbf{W}_{\bar{x}\bar{x}}]_{ij} = \mathbf{t}_{ij}$$

Assuming the existence of finite eighth-order moments, this default weight matrix **W** is a consistent but biased estimator of the asymptotic covariance matrix $\mathbf{\Gamma}$.

By using the ASYCOV option, you can use Browne's (1984, formula (3.8)) unbiased estimator of $\mathbf{\Gamma}_{ss}$ as:

$$\begin{aligned}[\mathbf{W}_{ss}]_{ij,kl} &= \frac{N(N-1)}{(N-2)(N-3)}(\mathbf{t}_{ij,kl} - \mathbf{t}_{ij}\mathbf{t}_{kl}) \\ &\quad - \frac{N}{(N-2)(N-3)}(\mathbf{t}_{ik}\mathbf{t}_{jl} + \mathbf{t}_{il}\mathbf{t}_{jk} - \frac{2}{N-1}\mathbf{t}_{ij}\mathbf{t}_{kl})\end{aligned}$$

There is no guarantee that $\mathbf{W}_{ss}$ 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 χ^2 test statistic (Browne 1982, 1984).

With the default weight matrix **W** computed by PROC TCALIS, the WLS estimation is also called as the asymptotically distribution-free (ADF) method. In fact, as options in PROC TCALIS, 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 η 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:

$$F_{WLS} = (s-\sigma)'\mathbf{W}_{ss}^{-1}(s-\sigma)$$

If **S** is a correlation rather than a covariance matrix, the default setting of the $\mathbf{W}_{ss}$ is a consistent estimator of the asymptotic covariance matrix $\mathbf{\Gamma}_{ss}$ of $\sqrt{N}(s-\sigma_o)$ (Browne and Shapiro 1986; DeLeeuw 1983), with s and σ_o representing vectors of sample and population correlations, respectively. Elementwise, $\mathbf{W}_{ss}$ is expressed as:

$$\begin{aligned}[\mathbf{W}_{ss}]_{ij,kl} &= r_{ij,kl} - \frac{1}{2}r_{ij}(r_{ii,kl} + r_{jj,kl}) - \frac{1}{2}r_{kl}(r_{kk,ij} + r_{ll,ij}) \\ &\quad + \frac{1}{4}r_{ij}r_{kl}(r_{ii,kk} + r_{ii,ll} + r_{jj,kk} + r_{jj,ll})\end{aligned}$$

where

$$r_{ij} = \frac{\mathbf{t}_{ij}}{\sqrt{\mathbf{t}_{ii}\mathbf{t}_{jj}}}$$

and

$$r_{ij,kl} = \frac{\mathbf{t}_{ij,kl}}{\sqrt{\mathbf{t}_{ii}\mathbf{t}_{jj}\mathbf{t}_{kk}\mathbf{t}_{ll}}}$$

The asymptotic variances of the diagonal elements of a correlation matrix are 0. That is,

$$[\mathbf{W}_{ss}]_{ii,ii} = 0$$

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:

$$\begin{aligned} F_{WLS} &= \sum_{i=2}^{n}\sum_{j=1}^{i-1}\sum_{k=2}^{n}\sum_{l=1}^{k-1}[\mathbf{W}_{ss}]^{ij,kl}([\mathbf{S}]_{ij} - [\mathbf{\Sigma}]_{ij})([\mathbf{S}]_{kl} - [\mathbf{\Sigma}]_{kl}) \\ &\quad + r\sum_{i}^{n}([\mathbf{S}]_{ii} - [\mathbf{\Sigma}]_{ii})^2 \end{aligned}$$

where r is the penalty weight specified by the WPENALTY=r option and the $[\mathbf{W}_{ss}]^{ij,kl}$ 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 $\mathbf{W}_{ss}$.

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 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:

$$\begin{aligned} F_{DWLS} &= (u-\eta)'[\text{diag}(\mathbf{W})]^{-1}(u-\eta) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{i}[\mathbf{W}_{ss}]^{-1}_{ij,ij}([\mathbf{S}]_{ij} - [\mathbf{\Sigma}]_{ij})^2 + \sum_{i=1}^{n}[\mathbf{W}_{\bar{x}\bar{x}}]^{-1}_{ii}(\bar{x}_i - \boldsymbol{\mu}_i)^2 \end{aligned}$$

When only the covariance structures are modeled, the discrepancy function becomes:

$$F_{DWLS} = \sum_{i=1}^{n}\sum_{j=1}^{i}[\mathbf{W}_{ss}]^{-1}_{ij,ij}([\mathbf{S}]_{ij} - [\mathbf{\Sigma}]_{ij})^2$$

For correlation models, the discrepancy function is:

$$F_{DWLS} = \sum_{i=2}^{n}\sum_{j=1}^{i-1}[\mathbf{W}_{ss}]^{-1}_{ij,ij}([\mathbf{S}]_{ij} - [\mathbf{\Sigma}]_{ij})^2 + r\sum_{i=1}^{n}([\mathbf{S}]_{ii} - [\mathbf{\Sigma}]_{ii})^2$$

where r is the penalty weight specified by the 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 $\mathbf{W}$ by using an INWGT= data set. The TCALIS 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, ..., N_k are the sample sizes for the groups. The overall discrepancy function $F(\mathbf{\Theta})$ is expressed as a weighted sum of individual discrepancy functions F_i's for the groups:

$$F(\mathbf{\Theta}) = \sum_{i=1}^{k} t_i F_i(\mathbf{\Theta})$$

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 TCALIS, 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. However, you can fit covariance and mean structures to a group but fit covariance structures only to another group.

Relationships among Estimation Criteria

If only the covariance or correlation structures are considered, the five estimation functions, F_{ULS}, F_{GLS}, F_{ML}, F_{WLS}, and F_{DWLS}, belong to the following two groups:

- The functions F_{ULS}, F_{GLS}, and F_{ML} take into account all n^2 elements of the symmetric residual matrix $\mathbf{S} - \mathbf{\Sigma}$. 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 $\mathbf{S} - \mathbf{\Sigma}$. This means that the off-diagonal residuals contribute to the discrepancy function F only once.

The F_{DWLS} function used in PROC TCALIS differs from that used by the LISREL 7 program. Formula (1.25) of the LISREL 7 manual (Jöreskog and Sörbom 1988, 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 $\mathbf{S} - \mathbf{\Sigma}$.

- Relationship between DWLS and WLS:
  PROC TCALIS: The F_{DWLS} and F_{WLS} discrepancy functions deliver the same results for the special case that the weight matrix $\mathbf{W} = \mathbf{W}_{ss}$ 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 $\mathbf{W} = \mathbf{W}_{ss}$ used by DWLS estimation is an identity matrix.
  PROC TCALIS: To obtain the same results with F_{DWLS} and F_{ULS} estimation, set the diagonal weight matrix $\mathbf{W} = \mathbf{W}_{ss}$ used in DWLS estimation to:

  $$[\mathbf{W}_{ss}]_{ik,ik} = \begin{cases} 1. & \text{if } i = k \\ 0.5 & \text{otherwise} \qquad (k \le i) \end{cases}$$

  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.

Gradient, Hessian, Information Matrix, and Approximate Standard Errors

For a single-sample setting with a discrepancy function $F = F(\mathbf{\Sigma}(\mathbf{\Theta}), \boldsymbol{\mu}(\mathbf{\Theta}); \mathbf{S}, \bar{\boldsymbol{x}})$, the gradient is defined as the first partial derivatives of the discrepancy function with respect to the model parameters $\mathbf{\Theta}$:

$$g(\mathbf{\Theta}) = \frac{\partial}{\partial \mathbf{\Theta}} F(\mathbf{\Theta})$$

The Hessian is defined as the second partial derivatives of the discrepancy function with respect to the model parameters $\mathbf{\Theta}$:

$$H(\mathbf{\Theta}) = \frac{\partial^2}{\partial \mathbf{\Theta} \partial \mathbf{\Theta}'} F(\mathbf{\Theta})$$

Suppose that the mean and covariance structures fit perfectly with $\mathbf{\Theta} = \mathbf{\Theta}_o$ in the population. The information matrix is defined as:

$$I(\mathbf{\Theta}_o) = \frac{1}{2}\mathcal{E}(H(\mathbf{\Theta}_o))$$

where the expectation $\mathcal{E}(\cdot)$ is taken over the sampling space of $\mathbf{S}, \bar{\mathbf{x}}$.

The information matrix plays a significant role in statistical theory. Under certain regularity conditions, the inverse of the information matrix $I^{-1}(\mathbf{\Theta}_o)$ is the asymptotic covariance matrix for $\sqrt{N}(\hat{\mathbf{\Theta}} - \mathbf{\Theta}_o)$, where N denotes the sample size and $\hat{\mathbf{\Theta}}$ is an estimator.

In practice, $\mathbf{\Theta}_o$ is never known and can only be estimated. The information matrix is therefore estimated by the so-called empirical information matrix:

$$I(\hat{\mathbf{\Theta}}) = \frac{1}{2}H(\hat{\mathbf{\Theta}})$$

which is evaluated at the values of the sample estimates $\hat{\mathbf{\Theta}}$. Notice that this empirical information matrix, rather than the unknown $I(\mathbf{\Theta}_o)$, is the "information matrix" displayed in PROC TCALIS output.

Taking the inverse of the empirical information matrix with sample size adjustment, PROC TCALIS approximates the estimated covariance matrix of $\hat{\mathbf{\Theta}}$ by:

$$((N-1)I(\hat{\mathbf{\Theta}}))^{-1} = ((N-1)\frac{1}{2}H(\hat{\mathbf{\Theta}}))^{-1} = \frac{2}{N-1}H^{-1}(\hat{\mathbf{\Theta}})$$

Approximate standard errors for $\hat{\mathbf{\Theta}}$ 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 TCALIS 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 6885, the overall discrepancy function for multiple-group analysis is defined. The same notation is applied here. To begin with, the overall discrepancy function $F(\mathbf{\Theta})$ is expressed as a weighted sum of individual discrepancy functions F_i's for the groups as follows:

$$F(\mathbf{\Theta}) = \sum_{i=1}^{k} t_i F_i(\mathbf{\Theta})$$

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 $g(\mathbf{\Theta})$ and the Hessian $H(\mathbf{\Theta})$ are now defined as weighted sum of individual functions. That is,

$$g(\mathbf{\Theta}) = \sum_{i=1}^{k} t_i g_i(\mathbf{\Theta}) = \sum_{i=1}^{k} t_i \frac{\partial}{\partial \mathbf{\Theta}} F_i(\mathbf{\Theta})$$

and

$$H(\mathbf{\Theta}) = \sum_{i=1}^{k} t_i H_i(\mathbf{\Theta}) = \sum_{i=1}^{k} t_i \frac{\partial^2}{\partial \mathbf{\Theta} \partial \mathbf{\Theta}'} F_i(\mathbf{\Theta})$$

Suppose that the mean and covariance structures fit perfectly with $\mathbf{\Theta} = \mathbf{\Theta}_o$ in the population. If each t_i converges to a fixed constant τ_i ($\tau_i > 0$) with increasing total sample size, the information matrix can be written as:

$$I(\mathbf{\Theta}_o) = \frac{1}{2} \sum_{i=1}^{k} \tau_i \mathcal{E}(H_i(\mathbf{\Theta}_o))$$

To approximate this information matrix, an empirical counterpart is used:

$$I(\hat{\mathbf{\Theta}}) = \frac{1}{2} \sum_{i=1}^{k} t_i H_i(\hat{\mathbf{\Theta}})$$

which is evaluated at the values of the sample estimates $\hat{\mathbf{\Theta}}$. Again, this empirical information matrix, rather than the unknown $I(\mathbf{\Theta}_o)$, is the "information matrix" output in PROC TCALIS results.

Taking the inverse of the empirical information matrix with sample size adjustment, PROC TCALIS approximates the estimated covariance matrix of $\hat{\mathbf{\Theta}}$ in multiple-group analysis by:

$$((N-k)I(\hat{\mathbf{\Theta}}))^{-1} = ((N-k)\frac{1}{2}H(\hat{\mathbf{\Theta}}))^{-1} = \frac{2}{N-k} \sum_{i=1}^{k} t_i H_i^{-1}(\hat{\mathbf{\Theta}})$$

Approximate standard errors for $\hat{\mathbf{\Theta}}$ 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.

1. The inversion (usually of a normalized matrix $\mathbf{D}^{-1}\mathbf{H}\mathbf{D}^{-1}$) 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 TCALIS 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 TCALIS.

Counting the Degrees of Freedom

When fitting covariance and mean structure models, the population moments are hypothesized to be functions of model parameters $\boldsymbol{\Theta}$. 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 larger 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 $p_1, p_2, \ldots, p_k$ mainfest 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

$\delta_1, \delta_2, \cdots, \delta_k$ as zero-one indicators of the mean structures for the groups. If δ_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:

$$q = \sum_{i=1}^{k} (p_i(p_i + 1)/2 + \delta_i p_i)$$

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 PARMS statement, or it is generated automatically by PROC TCALIS 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 TCALIS 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:

```
std    E1-E3 = vare1 vare2 vare3;
```

You define three variance parameter locations with three independent parameters vare1, vare2, and vare3. However, in the following specification:

```
std    E1-E3 = vare vare vare;
```

you still have three variance parameter locations to define, but the number of independent parameter is only one, which is the parameter named 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:

```
bounds  3 <= parm01 <= 3;
lincon  3 * parm02 + 2 * parm03 = 12;
```

In the BOUNDS statement, parm01 is constrained to a fixed number 3, and in the LINCON statement, parm02 and 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:

```
bounds  0 <= var01;
lincon  3 * beta1 + 2 * beta2 >= 7;
```

The optimal solution occurs at the boundary point so that you observe in the final solution the following two equalities:

```
var01 = 0,
3 * beta1 + 2 * beta2 = 7
```

These two active constraints reduce the number of independent parameters of your original model. As a result, PROC TCALIS 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. Refer to 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 TCALIS 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 TCALIS applies mainly to models with covariance structures with or without mean structures. When you model correlation structures, the degrees of freedom calculation in PROC TCALIS 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 TCALIS does not do this automatically, you can use the DFREDUCE=*i* 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 TCALIS to specify the effective number of observations in the input data set only.

Assessment of Fit

In PROC TCALIS, 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
- $\mathbf{\Theta}$ for the t-vector of parameters, $\hat{\mathbf{\Theta}}$ for the estimated parameters
- $\mathbf{S} = (s_{ij})$ for the $p \times p$ input covariance or correlation matrix
- $\bar{\mathbf{x}} = (\bar{x}_i)$ for the p-vector of sample means
- $\hat{\mathbf{\Sigma}} = \mathbf{\Sigma}(\hat{\mathbf{\Theta}}) = (\hat{\sigma}_{ij})$ for the predicted covariance or correlation matrix
- $\hat{\boldsymbol{\mu}} = (\hat{\mu}_i)$ for the predicted mean vector

- δ for indicating the modeling of the mean structures
- $\mathbf{W}$ 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, $\mathbf{S}_r$, $\bar{\mathbf{x}}_r$, $\hat{\mathbf{\Sigma}}_r$, $\hat{\boldsymbol{\mu}}_r$, δ_r, and $\mathbf{W}_r$ is used for multiple-group situations.

Residuals

Residuals indicate how well each entry or element in the mean or covariance matrix is fitted. Large residuals indicate bad fit.

PROC TCALIS computes four types of residuals and writes them to the OUTSTAT= data set when requested.

- **raw residuals**

  $$s_{ij} - \hat{\sigma}_{ij}, \quad \bar{x}_i - \hat{\mu}_i$$

  for the covariance and mean residuals, respectively. The raw residuals are displayed whenever the PALL, PRINT, or RESIDUAL option is specified.

- **variance standardized residuals**

  $$\frac{s_{ij} - \hat{\sigma}_{ij}}{\sqrt{s_{ii}s_{jj}}}, \quad \frac{\bar{x}_i - \hat{\mu}_i}{\sqrt{s_{ii}}}$$

  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**

  $$\frac{s_{ij} - \hat{\sigma}_{ij}}{\sqrt{v_{ij,ij}}}, \quad \frac{\bar{x}_i - \hat{\mu}_i}{\sqrt{u_{ii}}}$$

  for the covariance and mean residuals, respectively; with

  $$v_{ij,ij} = (\hat{\mathbf{\Gamma}}_1 - \mathbf{J}_1 \hat{\mathrm{Cov}}(\hat{\mathbf{\Theta}}) \mathbf{J}_1')_{ij,ij}$$

$$u_{ii} = (\hat{\mathbf{\Gamma}}_2 - \mathbf{J}_2 \hat{\mathrm{Cov}}(\hat{\mathbf{\Theta}})\mathbf{J}_2')_{ii}$$

where $\hat{\mathbf{\Gamma}}_1$ is the $p^2 \times p^2$ estimated asymptotic covariance matrix of sample covariances, $\hat{\mathbf{\Gamma}}_2$ is the $p \times p$ estimated asymptotic covariance matrix of sample means, $\mathbf{J}_1$ is the $p^2 \times t$ Jacobian matrix $d\mathbf{\Sigma}/d\mathbf{\Theta}$, $\mathbf{J}_2$ is the $p \times t$ Jacobian matrix $d\boldsymbol{\mu}/d\mathbf{\Theta}$, and $\hat{Cov}(\hat{\mathbf{\Theta}})$ 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 $\hat{\mathbf{\Gamma}}_1$ and $\hat{\mathbf{\Gamma}}_2$. 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.

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 $\hat{\mathbf{\Gamma}}_1$.

- **normalized residuals**

$$\frac{s_{ij} - \hat{\sigma}_{ij}}{\sqrt{(\hat{\mathbf{\Gamma}}_1)_{ij,ij}}}, \quad \frac{\bar{x}_i - \hat{\mu}_i}{\sqrt{(\hat{\mathbf{\Gamma}}_2)_{ii}}}$$

for the covariance and mean residuals, respectively; with $\hat{\mathbf{\Gamma}}_1$ as the $p^2 \times p^2$ estimated asymptotic covariance matrix of sample covariances; and $\hat{\mathbf{\Gamma}}_2$ as the $p \times p$ estimated asymptotic covariance matrix of sample means.

Diagonal elements of $\hat{\mathbf{\Gamma}}_1$ and $\hat{\mathbf{\Gamma}}_2$ are defined for the following methods:

- **GLS**: $(\hat{\mathbf{\Gamma}}_1)_{ij,ij} = \frac{1}{(N-1)}(s_{ii}s_{jj} + s_{ij}^2)$ and $(\hat{\mathbf{\Gamma}}_2)_{ii} = \frac{1}{(N-1)}s_{ii}$
- **ML**: $(\hat{\mathbf{\Gamma}}_1)_{ij,ij} = \frac{1}{(N-1)}(\hat{\sigma}_{ii}\hat{\sigma}_{jj} + \hat{\sigma}_{ij}^2)$ and $(\hat{\mathbf{\Gamma}}_2)_{ii} = \frac{1}{(N-1)}\hat{\sigma}_{ii}$
- **WLS**: $(\hat{\mathbf{\Gamma}}_1)_{ij,ij} = \frac{1}{(N-1)}W_{ij,ij}$ and $(\hat{\mathbf{\Gamma}}_2)_{ii} = \frac{1}{(N-1)}s_{ii}$

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 $\hat{\mathbf{\Gamma}}_1$.

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 $\mathbf{\Gamma}_1$ 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 $\mathbf{\Gamma}_1$ and $\mathbf{\Gamma}_2$), the normalized residuals cannot be larger than the asymptotically standardized residuals (which use the diagonal of the form $\mathbf{\Gamma} - \mathbf{J}\hat{\text{Cov}}(\hat{\mathbf{\Theta}})\mathbf{J}'$).

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 TCALIS 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 TCALIS:

- 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 6880 for definitions of various discrepancy functions available in PROC TCALIS. 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 $a_r = \frac{(N_j - 1)}{(N - k)}$ 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 $\mathbf{\Theta}$ during estimation.

  The minimized function value of F will be denoted as f_{min}, which is always positive, with small values indicating good fit.

- **χ^2 test statistic**
  For the ML, GLS, and the WLS estimation, the overall χ^2 measure for testing model fit is:

  $$\chi^2 = (N - k) * f_{min}$$

  where f_{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_{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 χ^2 test is valid only if the observations are independent and identically distributed, the analysis is based on the nonstandardized sample covariance matrix $\mathbf{S}$, 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 TCALIS, the notation "Prob > Chi-Square" means "the probability of obtaining a greater χ^2 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 χ^2 value (Browne 1982)**
  If the variables are p-variate elliptic rather than normal and have significant amounts of multivariate kurtosis (leptokurtic or platykurtic), the χ^2 value can be adjusted to:

  $$\chi^2_{ell} = \frac{\chi^2}{\eta_2}$$

  where η_2 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:

  $$Z = \frac{\sqrt[3]{\frac{\chi^2}{d}} - (1 - \frac{2}{9d})}{\sqrt{\frac{2}{9d}}}$$

  where d is the degrees of freedom of the model. Refer to McArdle (1988) and Bishop, Fienberg, and Holland (1977, p. 527) for an application of the Z-test.

- **critical N index (Hoelter 1983)**

  $$\text{CN} = \frac{\chi^2_{crit}}{f_{min}} + 1$$

  where χ^2_{crit} is the critical chi-square value for the given d degrees of freedom and probability $\alpha = 0.05$. Refer to Bollen (1989b, p. 277). 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.

- **root mean square residual (RMR)**
  For a single-group analysis, the RMR is the root of the mean of the squared residuals:

  $$\text{RMR} = \sqrt{\frac{2}{p(p+1+2\delta)}[\sum_i^p \sum_j^i (s_{ij} - \hat{\sigma}_{ij})^2 + \delta \sum_i^p (\bar{x}_i - \hat{\mu}_i)^2]}$$

  For multiple-group analysis, PROC TCALIS computes the root mean square residual RMR_r for each group first. To obtain an overall RMR measure for the analysis, individual RMR_r's are weighted by the group weights $a_r = \frac{N_r - 1}{N-k}$. That is,

  $$\text{overall RMR} = \sqrt{\sum_{r=1}^{k} a_r \text{RMR}_r^2}$$

- **standardized root mean square residual (SRMR)**
  For a single-group analysis, the SRMR is the root of the mean of the squared standardized residuals:

  $$\text{SRMR} = \sqrt{\frac{2}{p(p+1+2\delta)}[\sum_i^p \sum_j^i \frac{(s_{ij} - \hat{\sigma}_{ij})^2}{s_{ii}s_{jj}} + \delta \sum_i^p \frac{(\bar{x}_i - \hat{\mu}_i)^2}{s_{ii}}]}$$

  Similar to the calculation of the overall RMR, an overall measure of SRMR in a multiple-group analysis is a weighted average of the individual SRMR's. That is, with $a_r = \frac{N_r - 1}{N-k}$

  $$\text{overall SRMR} = \sqrt{\sum_{r=1}^{k} a_r \text{SRMR}_r^2}$$

- **goodness-of-fit index (GFI)**
  For a single-group analysis, the goodness-of-fit index for the ULS, GLS, and ML estimation methods is:

  $$GFI = 1 - \frac{Tr((\mathbf{W}^{-1}(\mathbf{S} - \hat{\mathbf{\Sigma}}))^2) + \delta(\bar{\mathbf{x}} - \hat{\boldsymbol{\mu}})'\mathbf{W}^{-1}(\bar{\mathbf{x}} - \hat{\boldsymbol{\mu}})}{Tr((\mathbf{W}^{-1}\mathbf{S})^2) + \delta\bar{\mathbf{x}}'\mathbf{W}^{-1}\bar{\mathbf{x}}}$$

  with $\mathbf{W} = I$ for ULS, $\mathbf{W} = \mathbf{S}$ for GLS, and $\mathbf{W} = \hat{\mathbf{\Sigma}}$. For WLS and DWLS estimation,

  $$GFI = 1 - \frac{(u - \hat{\eta})'\mathbf{W}^{-1}(u - \hat{\eta})}{u'\mathbf{W}^{-1}u}$$

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 vecs($\mathbf{S}$) in the covariance matrix and all the means. That is,

$$u = (\text{vecs}'(\mathbf{S}), \bar{\boldsymbol{x}}')', \quad \hat{\eta} = (\text{vecs}'(\hat{\boldsymbol{\Sigma}}), \hat{\boldsymbol{\mu}}')'$$

and the symmetric weight matrix $\mathbf{W}$ 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 vecs($\mathbf{S}$) in the covariance matrix only. That is,

$$u = \text{vecs}(\mathbf{S}), \quad \hat{\eta} = \text{vecs}(\hat{\boldsymbol{\Sigma}})$$

and the symmetric weight matrix $\mathbf{W}$ is of dimension $p \times (p+1)/2$. In addition, for the DWLS estimation, $\mathbf{W}$ is a diagonal matrix.

For a constant weight matrix $\mathbf{W}$, 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 larger 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,

$$\text{overall GFI} = \sum_{r=1}^{k} a_r \text{GFI}_r$$

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

  $$c = \sum_{r=1}^{k} \frac{p_k(p_k + 1 + 2\delta_k)}{2}$$

  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 larger than 1. For more information, refer to 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:

$$\text{PGFI} = \frac{d_{min}}{d_0}\text{GFI}$$

  where d_{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 6901 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 Steiger and Lind (1980) (see also Steiger 1998) root mean square error approximation (RMSEA) coefficient is:

$$\epsilon = \sqrt{k}\sqrt{\max(\frac{f_{min}}{d_{min}} - \frac{1}{(N-k)}, 0)}$$

  The lower and upper limits of the $(1-\alpha)\%$-confidence interval are computed using the cumulative distribution function of the noncentral chi-squared distribution $\Phi(x|\lambda, d)$. With $x = (N-k)f_{min}$, λ_L satisfying $\Phi(x|\lambda_L, d_{min}) = 1-\frac{\alpha}{2}$, and λ_U satisfying $\Phi(x|\lambda_U, d_{min}) = \frac{\alpha}{2}$:

$$(\epsilon_{\alpha_L}; \epsilon_{\alpha_U}) = (\sqrt{k}\sqrt{\frac{\lambda_L}{(N-k)d_{min}}}; \sqrt{k}\sqrt{\frac{\lambda_U}{(N-k)d_{min}}})$$

  Refer to Browne and Du Toit (1992) for more details. The size of the confidence interval can be set by the option ALPHARMS=α, $0 \le \alpha \le 1$. 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 χ^2 test hypothesis $H_0\colon \epsilon = 0$ is replaced by the null hypothesis of close fit $H_0\colon \epsilon \le 0.05$ and the exceedance probability P is computed as:

$$P = 1 - \Phi(x|\lambda^*, d_{min})$$

  where $x = (N-k)f_{min}$ and $\lambda^* = 0.05^2(N-k)d_{min}/k$. The null hypothesis of close fit is rejected if P is smaller than a pre-specified 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. For other cases, ECVI is not defined in PROC TCALIS. 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-squared distribution,

$$(c_L; c_U) = (\frac{\lambda_L + p(p+1)/2 + t}{(N-1)}; \frac{\lambda_U + p(p+1)/2 + t}{(N-1)})$$

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:

$$(c_L^*; c_U^*) = (\frac{\lambda_L^* + p(p+1)/2 + t}{N-p-2}; \frac{\lambda_U^* + p(p+1)/2 + t}{N-p-2})$$

where $x = (N-p-2)f_{min}$, $\Phi(x|\lambda_U^*, d_{min}) = \frac{\alpha}{2}$ and $\Phi(x|\lambda_L^*, d_{min}) = 1 - \frac{\alpha}{2}$. Refer to Browne and Cudeck (1993). The size of the confidence interval can be set by the option ALPHAECV=α, $0 \le \alpha \le 1$. 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.

  $$\text{AIC} = \chi^2 - 2d_{min}$$

- **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 χ^2 test.

  $$\text{CAIC} = \chi^2 - (ln(N) + 1)d_{min}$$

- **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 χ^2 test.

  $$\text{SBC} = \chi^2 - ln(N)d_{min}$$

- **McDonald's measure of centrality (McDonald and Hartmann 1992)**

  $$\text{CENT} = \exp(-\frac{(\chi^2 - d_{min})}{2N})$$

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_{min} and d_{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)**

$$\text{CFI} = 1 - \frac{\max((N-k)f_{min} - d_{min}, 0)}{\max((N-k)f_0 - d_0, 0)}$$

- **Bentler-Bonett normed fit index (NFI) (Bentler and Bonett 1980)**

$$\Delta = \frac{f_0 - f_{min}}{f_0}$$

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)**

$$\rho = \frac{f_0/d_0 - f_{min}/d_{min}}{f_0/d_0 - 1/(N-k)}$$

Refer to Tucker and Lewis (1973).

- **normed index ρ_1 (Bollen 1986)**

$$\rho_1 = \frac{f_0/d_0 - f_{min}/d_{min}}{f_0/d_0}$$

ρ_1 is always less than or equal to 1; $\rho_1 < 0$ is unlikely in practice. Refer to the discussion in Bollen (1989a).

- **nonnormed index Δ_2 (Bollen 1989a)**

$$\Delta_2 = \frac{f_0 - f_{min}}{f_0 - \frac{d_{min}}{(N-k)}}$$

is a modification of Bentler and Bonett's Δ that uses d and "lessens the dependence" on N. Refer to the discussion in Bollen (1989b). Δ_2 is identical to Mulaik et al.'s (1989) IFI2 index.

- **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,

  $$\text{PNFI} = \frac{d_{min}}{d_0} \frac{(f_0 - f_{min})}{f_0}$$

  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 TCALIS 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 χ^2 value according to the groups. This eliminates the possibility of breaking down those fit indices that are functions of degrees of freedom or χ^2 for group comparison purposes. Bearing this fact in mind, PROC TCALIS 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 $a_r = \frac{(N_j - 1)}{(N-k)}$ 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 large 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:

$$g = \frac{p_r(p_r + 1 + 2\delta_r)}{2}$$

  where p_r is the number of variables and δ_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:

$$\text{RMR}_r = \sqrt{\frac{1}{g}[\sum_i^{p_r}\sum_j^{i}([\mathbf{S}_r]_{ij} - [\hat{\mathbf{\Sigma}}_r]_{ij})^2 + \delta_r \sum_i^{p_r}([\bar{\mathbf{x}}_r]_i - [\hat{\boldsymbol{\mu}}_r]_i)^2]}$$

- **standardized root mean square residual (SRMR)**
  For the r-th group, the standardized root mean square residual is:

$$\text{SRMR} = \sqrt{\frac{1}{g}[\sum_i^{p_r}\sum_j^{i}\frac{([\mathbf{S}_r]_{ij} - [\hat{\mathbf{\Sigma}}_r]_{ij})^2}{[\mathbf{S}_r]_{ii}[\mathbf{S}_r]_{jj}} + \delta_r \sum_i^{p_r}\frac{([\bar{\mathbf{x}}_r]_i - [\hat{\boldsymbol{\mu}}_r]_i)^2}{[\mathbf{S}_r]_{ii}}]}$$

- **goodness-of-fit index (GFI)**
  For the ULS, GLS, and ML estimation, the goodness-of-fit index (GFI) for the r-th group is:

$$GFI = 1 - \frac{Tr((\mathbf{W}_r^{-1}(\mathbf{S}_r - \hat{\mathbf{\Sigma}}_r))^2) + \delta_r(\bar{\mathbf{x}}_r - \hat{\boldsymbol{\mu}}_r)'\mathbf{W}_r^{-1}(\bar{\mathbf{x}}_r - \hat{\boldsymbol{\mu}}_r)}{Tr((\mathbf{W}_r^{-1}\mathbf{S}_r)^2) + \delta_r\bar{\mathbf{x}}_r'\mathbf{W}_r^{-1}\bar{\mathbf{x}}_r}$$

  with $\mathbf{W}_r = I$ for ULS, $\mathbf{W}_r = \mathbf{S}_r$ for GLS, and $\mathbf{W}_r = \hat{\mathbf{\Sigma}}_r$. For the WLS and DWLS estimation,

$$GFI = 1 - \frac{(u_r - \hat{\eta}_r)'\mathbf{W}_r^{-1}(u_r - \hat{\eta}_r)}{u_r'\mathbf{W}_r^{-1}u_r}$$

  where u_r is the vector of observed moments and $\hat{\eta}_r$ 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 $\hat{\eta}_r$ contain all the nonredundant elements vecs($\mathbf{S}_r$) in the covariance matrix and all the means, and $\mathbf{W}_r$ is the weight matrix for covariances and means. When the mean structures are not modeled, u_r, $\hat{\eta}_r$, and $\mathbf{W}_r$ 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.

- **Bentler-Bonnett normed fit index (NFI)**
  For the r-th group, the Bentler-Bonnett 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 Δ_r, the better is the fit for the group. Basically, the formula here is the same as the overall Bentler-Bonnet 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**

$$R^2 = 1 - \frac{\widehat{\text{Evar}}(y)}{\widehat{\text{Var}}(y)}$$

  where y denotes an endogenous variable, $\widehat{\text{Var}}(y)$ denotes its variance, and $\widehat{\text{Evar}}(y)$ denotes its error (or unsystematic) variance. The variance and error variance are estimated under the model.

- **total determination of all equations**

$$\text{DETAE} = 1 - \frac{|\widehat{\text{Ecov}}(\mathbf{y}, \boldsymbol{\eta})|}{|\widehat{\text{Cov}}(\mathbf{y}, \boldsymbol{\eta})|}$$

  where the $\mathbf{y}$ vector denotes all manifest dependent variables, the $\boldsymbol{\eta}$ vector denotes all latent dependent variables, $\widehat{\text{Cov}}(\mathbf{y}, \boldsymbol{\eta})$ denotes the covariance matrix of $\mathbf{y}$ and $\boldsymbol{\eta}$, and $\widehat{\text{Ecov}}(\mathbf{y}, \boldsymbol{\eta})$ denotes the error covariance matrix of $\mathbf{y}$ and $\boldsymbol{\eta}$. The covariance matrices are estimated under the model.

- **total determination of latent equations**

$$\text{DETSE} = 1 - \frac{|\widehat{\text{Ecov}}(\boldsymbol{\eta})|}{|\widehat{\text{Cov}}(\boldsymbol{\eta})|}$$

  where the $\boldsymbol{\eta}$ vector denotes all latent dependent variables, $\widehat{\text{Cov}}(\boldsymbol{\eta})$ denotes the covariance matrix of $\boldsymbol{\eta}$, and $\widehat{\text{Ecov}}(\boldsymbol{\eta})$ denotes the error covariance matrix of $\boldsymbol{\eta}$. The covariance matrices are estimated under the model.

- **total determination of the manifest equations**

$$\text{DETMV} = 1 - \frac{|\widehat{\text{Ecov}}(\mathbf{y})|}{|\widehat{\text{Cov}}(\mathbf{y})|}$$

where the $\mathbf{y}$ vector denotes all manifest dependent variables, $\widehat{\text{Cov}}(\mathbf{y})$ denotes the covariance matrix of $\mathbf{y}$, $\widehat{\text{Ecov}}(\mathbf{y})$ denotes the error covariance matrix of $\mathbf{y}$, and $|A|$ denotes the determinant of matrix A. All the covariance matrices in the formula are estimated under the model.

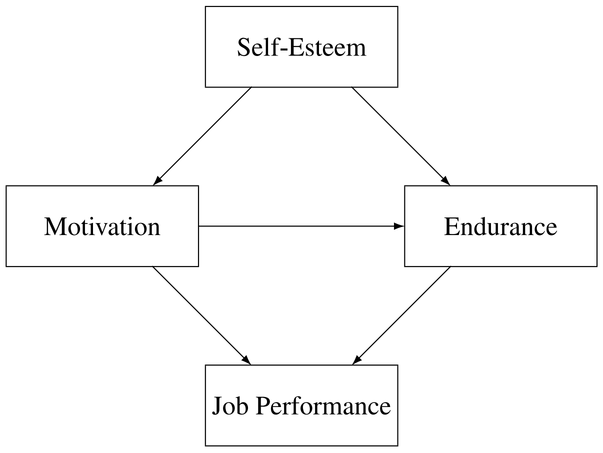
You can also use the DETERM statement to request the computations of determination coefficients for any subsets of dependent variables.

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 88.4 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 TCALIS statement and the EFFPART statement are designed to address these questions. By using the TOTEFF option or the EFFPART statement, PROC TCALIS 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:

$$\mathbf{D} = \begin{pmatrix} \beta & \gamma \\ 0 & 0 \end{pmatrix}$$

where β is an $(n_u \times n_u)$ matrix for direct effects of dependent variables on dependent variables and γ 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 β 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:

$$\mathbf{T} = \begin{pmatrix} (I - \beta)^{-1} - I & (I - \beta)^{-1}\gamma \\ 0 & 0 \end{pmatrix}$$

Finally, the $(n_u + n_w) \times (n_u + n_w)$ matrix **M** of **indirect** effects of column variables on the row variables is computed by the difference between **T** and **D** as:

$$\mathbf{M} = \begin{pmatrix} (I-\beta)^{-1} - I - \beta & (I-\beta)^{-1}\gamma - \gamma \\ 0 & 0 \end{pmatrix}$$

In PROC TCALIS, any subsets of **D**, **T**, and **M** 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 6743 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 $\boldsymbol{\beta}$ 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 $\boldsymbol{\beta}$ matrix. A stability coefficient less than one would ensure that all eigenvalues, complex or real, of the $\boldsymbol{\beta}$ 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 TCALIS 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 TCALIS provides standardized solutions routinely. In standardizing a solution, parameters are classified into five groups:

- **path coefficients, regression coefficients, or direct effects**
  With each parameter α in this group, there is an associated outcome variable and a predictor variable. Denote the predicted variance of the outcome variable by σ_o^2 and the variance of the predictor variable by σ_p^2, the standardized parameter α^* is:

  $$\alpha^* = \alpha \frac{\sigma_p}{\sigma_o}$$

- **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 σ_{ij} be the covariance between variables i and j. In this notation, σ_{ii} is the variance of variable i. The standardized covariance σ^*_{ij} is:

  $$\sigma^*_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$$

  When $i = j$, σ^*_{ii} takes the value of 1 for all i. Also, σ^*_{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 θ_{ij} so that θ_{ii} 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 σ_{ii}. In the standardized solution, the error covariance θ_{ij} is rescaled as:

  $$\theta^*_{ij} = \frac{\theta_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$$

  Notice that when $i = j$, θ^*_{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, θ^*_{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 6905, 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 6895 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 88.10 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 6895 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 TCALIS statement. To customize the LM tests by setting specific regions of parameters, you can use the LMTESTS statements. PROC TCALIS 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 χ^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 biggest model improvements (that is, largest χ^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 χ^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 χ^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 χ^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 χ^2 value is set to 0. Besides the multivariate χ^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, refer to 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

If the DATA= data set contains raw data (rather than a covariance or correlation matrix), observations with missing values for any variables in the analysis are omitted from the computations. 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.

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 TCALIS 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 TCALIS 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

$$\gamma_{1(j)} = \frac{N \sum_i^N (z_{ij} - \bar{z}_j)^3}{\sqrt{N[\sum_i^N (z_{ij} - \bar{z}_j)^2]^3}}$$

- **corrected univariate skewness for variable** z_j

$$\gamma_{1(j)} = \frac{N}{(N-1)(N-2)} \frac{\sum_i^N (z_{ij} - \bar{z}_j)^3}{\sigma_j^3}$$

- **uncorrected univariate kurtosis for variable** z_j

$$\gamma_{2(j)} = \frac{N \sum_i^N (z_{ij} - \bar{z}_j)^4}{[\sum_i^N (z_{ij} - \bar{z}_j)^2]^2} - 3$$

- **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**

$$\gamma_2 = \frac{1}{N} \sum_i^N [(z_i - \bar{z})' S^{-1} (z_i - \bar{z})]^2 - p(p+2)$$

- **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

  $$\gamma^*_{2(j)} = \begin{cases} \gamma_{2(j)} \quad , & \text{if} \quad \gamma_{2(j)} > \frac{-6}{p+2} \\ \frac{-6}{p+2} \quad , & \text{otherwise} \end{cases}$$

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 γ_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 κ_1, κ_2, and κ_3 should not be smaller than the following lower bound (Bentler 1985):

$$\hat{\kappa} \geq \frac{-2}{p+2}$$

PROC TCALIS displays a message if κ_1, κ_2, or κ_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 TCALIS procedure computes two more measures of multivariate kurtosis.

- **multivariate mean kappa**

  $$\kappa_4 = \frac{1}{m} \sum_i^p \sum_j^i \sum_k^j \sum_l^k \hat{\kappa}_{ij,kl} - 1$$

  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 γ_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 χ^2 value. In general, the parameter estimates are more stable against violation of the normal distribution assumption. For more details, refer to 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 TCALIS 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 TCALIS will use a combination of good strategic methods to compute initial estimates for your model.

These initial estimation methods are used in PROC TCALIS:

- 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 TCALIS displays the methods used to obtain initial estimates in the output.

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 TCALIS 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 TCALIS can find the correct solution. All optimization techniques in PROC TCALIS 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 TCALIS 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 TCALIS 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 TCALIS, 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.

OMETHOD=	Optimization Technique
LEVMAR	Levenberg-Marquardt method
TRUREG	trust-region method
NEWRAP	Newton-Raphson method with line search
NRRIDG	Newton-Raphson method with ridging
QUANEW	quasi-Newton methods (DBFGS, DDFP, BFGS, DFP)
DBLDOG	double-dogleg method (DBFGS, DDFP)
CONGRA	conjugate gradient methods (PB, FR, PR, CD)

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.

	Derivatives		Constraints		
OMETHOD=	First Order	Second Order	Boundary	Linear	Nonlinear
LEVMAR	x	x	x	x	-
TRUREG	x	x	x	x	-
NEWRAP	x	x	x	x	-
NRRIDG	x	x	x	x	-
QUANEW	x	-	x	x	x
DBLDOG	x	-	x	x	-
CONGRA	x	-	x	x	-

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 60 or 100 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 (1978a, 1978b, 1982a, 1982b) Variable Metric Constrained WatchDog (VMCWD) algorithm. 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 TCALIS 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 μ with the original update of Powell (1978a, 1978b), 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 μ is performed in the same way as Powell (1982a, 1982b) 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 $\mathbf{Z}_k'\mathbf{G}\mathbf{Z}_k$ and the QT decomposition are updated simultaneously when the active set changes. Refer to Gill et al. (1984) for more information.
- The line-search strategy is very similar to that of Powell (1982a, 1982b). 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 α of the first iterations.
- The watchdog strategy is similar to that of Powell (1982a, 1982b). 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:

$$(\nabla c_i) = (\frac{\partial c_i}{\partial x_j}), \quad i = 1, \dots, nc, j = 1, \dots, n$$

where nc 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 α 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 ρ (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 ρ 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 ρ means that the radius of the trust region does not need to be changed. And a value close to or larger 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 ρ (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 α (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 Δ 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 ABSGCONV= 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 α 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,

$$\begin{aligned} \text{lfgmax} &= \nabla_x L(x^{(k)}, \lambda^{(k)}) \\ &= \nabla_x f(x^{(k)}) - \sum_{i=1}^{m} \lambda_i^{(k)} \nabla_x c_i(x^{(k)}) \end{aligned}$$

For the double dogleg technique, the iteration history also includes the following information:

- the parameter λ 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 α 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 6723. The line-search methods LIS=1, LIS=2, and LIS=3 satisfy the left-hand-side and right-hand-side Goldstein conditions (Fletcher 1987). When derivatives are available, the line-search methods LIS=6, LIS=7, and LIS=8 try to satisfy the right-hand-side Goldstein condition; if derivatives are not available, these line-search algorithms use only function calls.

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 TCALIS provides options that restrict the step length or trust region radius, especially during the first main iterations.

The inner product $g's$ 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 α. 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 larger than r times the final step length α_- of the previous iteration. The default is $r = 2$.

- Not using the DAMPSTEP option:

  $$\alpha_1^{(0)} = \begin{cases} step & \text{if } 0.1 \le step \le 10 \\ 10 & \text{if } step > 10 \\ 0.1 & \text{if } step < 0.1 \end{cases}$$

  with

  $$step = \begin{cases} df/|g's| & \text{if } |g's| \ge \epsilon \max(100df, 1) \\ 1 & \text{otherwise} \end{cases}$$

  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 $0 < c \le 4$ depending on the ρ ratio. Reducing the radius corresponds to increasing the ridge parameter λ 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 TCALIS 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 TCALIS 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 DEMPHAS= 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 $\mathbf{\Theta}$ in the structural model

$$\mathbf{\Sigma} = \mathbf{\Sigma}(\mathbf{\Theta})$$

is said to be identified in a parameter space G, if

$$\mathbf{\Sigma}(\mathbf{\Theta}) = \mathbf{\Sigma}(\tilde{\mathbf{\Theta}}), \quad \tilde{\mathbf{\Theta}} \in G$$

implies $\mathbf{\Theta} = \tilde{\mathbf{\Theta}}$. 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 TCALIS.

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 TCALIS 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 6744 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 TCALIS 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 *VAL=EIGVAL(U)* to compute the vector *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:

$$R_i^2 = 1 - \frac{\widehat{var(\zeta_i)}}{\widehat{var(\eta_i)}}$$

When $\widehat{var(\zeta_i)} > \widehat{var(\eta_i)}$, R_i^2 is negative. This might indicate poor model fit or R^2 is an inappropriate measure for the model. For the latter case, for example, negative R^2 might be due to cyclical (nonrecursive) paths in the model so that the R^2 interpretation is not appropriate.

Displayed Output

The output of PROC TCALIS 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:

  – 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-, ξ-, and η- 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:

  – 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 6911 for more information.

- 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.

- 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 6915.

- 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 6895. Note that for ULS and DWLS estimation methods, many of those fit indices that are based on model fit χ^2 are not displayed. See the section "Overall Model Fit Indices" on page 6895 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 6902 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.

- 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" on page 6893 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 the ODS Graphics by the 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 estimation results and the standardized results. Except for ULS or DWLS estimates, the approximate standard errors and *t* values are also displayed. This output is displayed by default or if you specify the PESTIM option or at least the PSHORT option.

  Depending on the modeling language, the sets of output vary, as shown in the following:

  - 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, 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 6892 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's (1971, p.109) factor scores regression matrix,

  $$\mathbf{FSR} = \hat{\mathbf{\Sigma}}_{yx}\hat{\mathbf{\Sigma}}_{xx}^{-1}$$

  where $\hat{\mathbf{\Sigma}}_{xx}$ is the $p \times p$ predicted covariance matrix among manifest variables and $\hat{\mathbf{\Sigma}}_{yx}$ 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 χ^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 χ^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 6909 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 LMTESTS statement. These matrices are predefined in PROC TCALIS, or you can specify them in the LMTESTS 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 LMTESTS 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 6909 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 6909 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 6909 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 larger 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 6909 for more details.

ODS Table Names

PROC TCALIS 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 TCALIS procedure. The conditions for these ODS tables to display vary a lot. For convenience in presentation, the ODS tables for the PROC TCALIS procedure are organized in the following categories:

- ODS tables for descriptive statistics 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 TCALIS 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 TCALIS statement. See the section "Global Display Options" on page 6716 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 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 TCALIS or GROUP statement. If the specialized option is set in the PROC TCALIS 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 TCALIS statement to print these tables. Either a specialized or a global printing option is sufficient to print the tables.

Table Names for Descriptive Statistics

ODS Table Name	**Description**	**Specialized Option**	**Global Display Option**
ContKurtosis	Contributions to kurtosis from observations	KURTOSIS	PRINT
InCorr	Input correlation matrix	PCORR	PALL
InCorrDet	Determinant of the input correlation matrix	PCORR	PALL
InCov	Input covariance matrix	PCORR	PALL
InCovDet	Determinant of the input covariance matrix	PCORR	PALL
InMean	Input mean vector	PCORR	PALL
Kurtosis	Kurtosis, with raw data input	KURTOSIS	PRINT
PredCorr	Predicted correlation matrix	PCORR	PALL
PredCorrDet	Determinant of the predicted correlation matrix	PCORR	PALL
PredCov	Predicted covariance matrix	PCORR	PALL
PredCovDet	Determinant of the predicted covariance matrix	PCORR	PALL
PredMean	Predicted mean vector	PCORR	PALL
SimpleStatistics	Simple statistics, with raw data input	SIMPLE	default
Weights	Weight matrix	PWEIGHT	PALL
WeightsDet	Determinant of the weight matrix	PWEIGHT	PALL

Table Names for Residual Displays

ODS Table Name	Description	Specialized Option	Global Display Option
AsymStdRes	Asymptotically standardized residual matrix	RESIDUAL=ASYSTAND[1]	PALL
AveAsymStdRes	Average of absolute asymptotically standardized residual values	RESIDUAL=ASYSTAND[1]	PALL
AveNormRes	Average of absolute normalized residual values	RESIDUAL=NORM[1]	PALL
AveRawRes	Average of absolute raw residual values	RESIDUAL	PALL
AveVarStdRes	Average of absolute variance standardized residual values	RESIDUAL=VARSTAND[1]	PALL
DistAsymStdRes	Distribution of asymptotically standardized residuals	RESIDUAL=ASYSTAND[1]	PALL
DistNormRes	Distribution of normalized residuals	RESIDUAL=NORM[1]	PALL
DistRawRes	Distribution of raw residuals	RESIDUAL	PALL
DistVarStdRes	Distribution of variance standardized residuals	RESIDUAL=VARSTAND[1]	PALL
NormRes	Normalized residual matrix	RESIDUAL=NORM[1]	PALL
RawRes	Raw residual matrix	RESIDUAL[2]	PALL
RankAsymStdRes	Rank order of asymptotically standardized residuals	RESIDUAL=ASYSTAND[1]	PALL
RankNormRes	Rank order of normalized residuals	RESIDUAL=NORM[1]	PALL
RankRawRes	Rank order of raw residuals	RESIDUAL	PALL
RankVarStdRes	Rank order of variance standardized residuals	RESIDUAL=VARSTAND[1]	PALL
VarStdRes	Variance standardized residual matrix	RESIDUAL=VARSTAND[1]	PALL

1. In effect, the RESIDUAL= option specifies the RESIDUAL option and the type of residuals requested after the equal sign. For example, if you set RESIDUAL=ASYSTAND, asymptotically standardized residuals are requested, in addition to the printing of the tables enabled by the RESIDUAL option. In some cases, the RESIDUAL= option cannot be honored due to the specific estimation method or data type used. When this occurs, PROC TCALIS will determine the appropriate sets of normalized or standardized residuals to display. A warning message with an explanation will be printed.
2. Raw residuals are also printed for correlation analysis even if RESIDUAL or PALL is not specified.

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:

ODS Table Name	**Description**	**Global Display Option**	**Additional Specification Required**
AddParms	Estimates for additional parameters	PSHORT	PARAMETERS statement
AddParmsInit	Initial values for additional parameters	PSHORT	PARAMETERS statement
Fit	Fit summary	PSUMMARY	
GroupFit	Fit comparison among groups	PSUMMARY	Multiple groups
ModelingInfo	General modeling information	PSHORT	
ModelSummary	Summary of models and their labels and types	PSHORT	Multiple models[1]
ParmFunc	Parametric function testing	PSHORT	TESTFUNC statement
SimTest	Simultaneous tests of parametric functions	PSHORT	SIMTEST statement

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 TCALIS: 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 TCALIS 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 TCALIS 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:

Type of ODS Tables	**Table Name Suffix**	**Specialized Option**	**Global Display Option**
Initial specification	Init	PINITIAL	PSHORT
Unstandardized solutions	(none)	PESTIM	PSHORT
Standardized solutions	Std	PESTIM, and NOSTAND not used	PSHORT

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 TCALIS or MODEL statement. If you want matrix output for all models, set this option in the PROC TCALIS statement. If you want matrix output for a specific model, set this option in the associated MODEL statement only.

ODS Table Name	**Description**	**Additional Option**
FACTORCov	Estimated factor covariances	
FACTORCovInit	Initial factor covariances	
FACTORCovStd	Factor correlations	
FACTORErrVar	Estimated error variances	
FACTORErrVarInit	Initial error variances	
FACTORErrVarStd	Standardized results for error variances	
FACTORIntercepts	Estimated intercepts	
FACTORInterceptsInit	Initial intercepts	
FACTORLoadings	Estimated factor loadings	
FACTORLoadingsInit	Initial factor loadings	
FACTORLoadingsStd	Standardized factor loadings	
FACTORMeans	Estimated factor means	
FACTORMeansInit	Initial factor means	
FACTORRotCov	Estimated rotated factor covariances	
FACTORRotCovStd	Rotated factor correlations	
FACTORRotErrVar	Error variances in rotated solution	
FACTORRotErrVarStd	Standardized results for error variances in rotated solution	
FACTORRotLoadings	Rotated factor loadings	
FACTORRotLoadingsStd	Standardized rotated factor loadings	
FACTORRotMat	Rotation matrix	
FACTORScoresRegCoef	Factor scores regression coefficients	
FACTORVariables	Variables in the analysis	
LINEQSAlpha	Estimated intercept vector	PRIMAT
LINEQSAlphaInit	Initial intercept vector	PRIMAT
LINEQSBeta	Estimated _EQSBETA_ matrix	PRIMAT
LINEQSBetaInit	Initial _EQSBETA_ matrix	PRIMAT
LINEQSBetaStd	Standardized results for _EQSBETA_ matrix	PRIMAT
LINEQSCovExog	Estimated covariances among exogenous variables	
LINEQSCovExogInit	Initial covariances among exogenous variables	
LINEQSCovExogStd	Standardized results for covariances among exogenous variables	
LINEQSEq	Estimated equations	
LINEQSEqInit	Initial equations	
LINEQSEqStd	Standardized equations	
LINEQSGamma	Estimated _EQSGAMMA_ matrix	PRIMAT
LINEQSGammaInit	Initial _EQSGAMMA_ matrix	PRIMAT

ODS Table Name	Description	Additional Option
LINEQSGammaStd	Standardized results for _EQSGAMMA_ matrix	PRIMAT
LINEQSMeans	Estimated means for exogenous variables	
LINEQSMeansInit	Initial means for exogenous variables	
LINEQSNu	Estimated mean vector	PRIMAT
LINEQSNuInit	Initial mean vector	PRIMAT
LINEQSPhi	Estimated _EQSPHI_ matrix	PRIMAT
LINEQSPhiInit	Initial _EQSPHI_ matrix	PRIMAT
LINEQSPhiStd	Standardized results for _EQSPHI_ matrix	PRIMAT
LINEQSVarExog	Estimated variances of exogenous variables	
LINEQSVarExogInit	Initial variances of exogenous variables	
LINEQSVarExogStd	Standardized results for variances of exogenous variables	
LINEQSVariables	Exogenous and endogenous variables	
LISMODAlpha	Estimated _ALPHA_ vector	
LISMODAlphaInit	Initial _ALPHA_ vector	
LISMODBeta	Estimated _BETA_ matrix	
LISMODBetaInit	Initial _BETA_ matrix	
LISMODBetaStd	Standardized _BETA_ matrix	
LISMODGamma	Estimated _GAMMA_ matrix	
LISMODGammaInit	Initial _GAMMA_ matrix	
LISMODGammaStd	Standardized _GAMMA_ matrix	
LISMODKappa	Estimated _KAPPA_ vector	
LISMODKappaInit	Initial _KAPPA_ vector	
LISMODLambdaX	Estimated _LAMBDAX_ matrix	
LISMODLambdaXInit	Initial _LAMBDAX_ matrix	
LISMODLambdaXStd	Standardized _LAMBDAX_ matrix	
LISMODLambdaY	Estimated _LAMBDAY_ matrix	
LISMODLambdaYInit	Initial _LAMBDAY_ matrix	
LISMODLambdaYStd	Standardized _LAMBDAY_ matrix	
LISMODNuX	Estimated _NUX_ vector	
LISMODNuXInit	Initial _NUX_ vector	
LISMODNuY	Estimated _NUY_ vector	
LISMODNuYInit	Initial _NUY_ vector	
LISMODPhi	Estimated _PHI_ matrix	
LISMODPhiInit	Initial _PHI_ matrix	
LISMODPhiStd	Standardized _PHI_ matrix	
LISMODPsi	Estimated _PSI_ matrix	
LISMODPsiInit	Initial _PSI_ matrix	
LISMODPsiStd	Standardized _PSI_ matrix	
LISMODThetaX	Estimated _THETAX_ matrix	
LISMODThetaXInit	Initial _THETAX_ matrix	
LISMODThetaXStd	Standardized _THETAX_ matrix	
LISMODThetaY	Estimated _THETAY_ matrix	
LISMODThetaYInit	Initial _THETAY_ matrix	
LISMODThetaYStd	Standardized _THETAY_ matrix	

ODS Table Name	Description	Additional Option
LISMODVariables	Variables in the model	
MSTRUCTCov	Estimated _COV_ matrix	
MSTRUCTCovInit	Initial _COV_ matrix	
MSTRUCTCovStd	Standardized _COV_ matrix	
MSTRUCTMean	Estimated _MEAN_ vector	
MSTRUCTMeanInit	Initial _MEAN_ vector	
MSTRUCTVariables	Variables in the model	
PATHCovErrors	Estimated error covariances	
PATHCovErrorsInit	Initial error covariances	
PATHCovErrorsStd	Standardized error covariances	
PATHCovVarErr	Estimated covariances bewteen exogenous variables and errors	
PATHCovVarErrInit	Initial covariances bewteen exogenous variables and errors	
PATHCovVarErrStd	Standardized results for covariances bewteen exogenous variables and errors	
PATHCovVars	Estimated covariances among exogenous variables	
PATHCovVarsInit	Initial covariances among exogenous variables	
PATHCovVarsStd	Standardized results for covariances among exogenous variables	
PATHList	Estimated path list	
PATHListInit	Initial path list	
PATHListStd	Standardized path list	
PATHMeansIntercepts	Estimated intercepts	
PATHMeansInterceptsInit	Initial intercepts	
PATHVariables	Exogenous and endogenous variables	
PATHVarParms	Estimated variances or error variances	
PATHVarParmsInit	Initial variances or error variances	
PATHVarParmsStd	Standardized results for variances or error variances	
RAMAMat	Estimated _A_ matrix	PRIMAT
RAMAMatInit	Initial _A_ matrix	PRIMAT
RAMAMatStd	Standardized results of _A_ matrix	PRIMAT
RAMList	List of RAM estimates	
RAMListInit	List of initial RAM estimates	
RAMListStd	Standardized results for RAM estimates	
RAMPMat	Estimated _P_ matrix	PRIMAT
RAMPMatInit	Initial _P_ matrix	PRIMAT
RAMPMatStd	Standardized results of _P_ matrix	PRIMAT
RAMVariables	Exogenous and endogenous variables	
RAMWVec	Estimated mean and intercept vector	PRIMAT
RAMWVecInit	Initial mean and intercept vector	PRIMAT

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 TCALIS or MODEL statement. If the specialized options are used in the PROC TCALIS 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 TCALIS 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.

ODS Table Name	Description	Specialized Option or Statement	Global Display Option
Determination	Coefficients of determination	PDETERM DETERM[4]	default
DirectEffects	Direct effects	TOTEFF[1]	PRINT
DirectEffectsStd	Standardized direct effects	TOTEFF[1,3]	PRINT
EffectsOf	Effects of the listed variables	EFFPART[2]	PRINT
EffectsOn	Effects on the listed variables	EFFPART[2]	PRINT
IndirectEffects	Indirect effects	TOTEFF[1]	PRINT
IndirectEffectsStd	Standardized indirect effects	TOTEFF[1,3]	PRINT
LatentScoresRegCoef	Latent variable scores regression coefficients	PLATCOV	PRINT
PredCovLatent	Predicted covariances among latent variables	PLATCOV	PRINT
PredCovLatMan	Predicted covariances between latent and manifest variables	PLATCOV	PRINT
PredMeanLatent	Predicted means of latent variables	PLATCOV	PRINT
SqMultCorr	Squared multiple correlations	PESTIM	PSHORT
Stability	Stability coefficient of reciprocal causation	PDETERM, DETERM[4]	default
StdEffectsOf	Standardized effects of the listed variables	EFFPART[2,3]	PSHORT
StdEffectsOn	Standardized effects on the listed variables	EFFPART[2,3]	PSHORT
TotalEffects	Total effects	TOTEFF[1]	PRINT
TotalEffectsStd	Standardized total effects	TOTEFF[1,3]	PRINT

1. This refers to the TOTEFF or EFFPART option in the PROC TCALIS or MODEL statement.
2. This refers to the EFFPART statement specifications.
3. NOSTAND option must not be specified in the MODEL or PROC TCALIS statement.
4. PDETERM is an option specified in the PROC TCALIS 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 TCALIS or MODEL statement. When this option is set in the PROC TCALIS 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 TCALIS statement to print these ODS tables.

If the NOMOD option is set in the PROC TCALIS 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 LM indices in separate regions of parameter space are the default printing format. You can turn off this default printing by the NODEFAULT option in the LMTESTS statement for models. If you want to print matrices of LM test indices rather than the rankings of LM test indices, you can use the NORANK or MAXRANK=0 option in the LMTESTS statement.

These ODS tables are specific to the types of modeling languages used. This is noted in the last column of the following table.

ODS Table Name	Description	Model
LMRankCov	Covariances among variables	MSTRUCT
LMRankCovErr	Covariances among errors	LINEQS
LMRankCovErrofVar	Covariances among errors of variables	RAM or PATH
LMRankCovExog	Covariances among existing exogenous variables	LINEQS, PATH, or RAM
LMRankCovFactors	Covariance among factors	FACTOR
LMRankErrorVar	Error variances	FACTOR
LMRankFactMeans	Factor means	FACTOR
LMRankIntercepts	Intercepts	FACTOR, LINEQS, PATH, or RAM
LMRankLoadings	Factor loadings	FACTOR
LMRankMeans	Means of existing variables	LINEQS, MSTRUCT, PATH, or RAM
LMRankPaths	All possible paths in the model	PATH or RAM

ODS Table Name	Description	Model
LMRankPathsFromEndo	Paths from existing endogenous variables	LINEQS
LMRankPathsFromExog	Paths from existing exogenous variables	LINEQS
LMRankPathsNewEndo	Paths to existing exogenous variables	LINEQS
LMRankLisAlpha	LISMOD _ALPHA_	LISMOD
LMRankLisBeta	LISMOD _BETA_	LISMOD
LMRankLisGamma	LISMOD _GAMMA_	LISMOD
LMRankLisKappa	LISMOD _KAPPA_	LISMOD
LMRankLisLambdaX	LISMOD _LAMBDAX_	LISMOD
LMRankLisLambdaY	LISMOD _LAMBDAY_	LISMOD
LMRankLisNuX	LISMOD _NUX_	LISMOD
LMRankLisNuY	LISMOD _NUY_	LISMOD
LMRankLisPhi	LISMOD _PHI_	LISMOD
LMRankLisPsi	LISMOD _PSI_	LISMOD
LMRankLisThetaX	LISMOD _THETAX_	LISMOD
LMRankLisThetaY	LISMOD _THETAY_	LISMOD
LMRankMySet	Customized sets of new parameters defined in LMTESTS statements	any model

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. This option itself is not sufficient for **all** the matrices of LM tests to print. Only those matrices associated with the LM test indices that are selected (by default or by customized sets) in the LMTESTS statement will be printed.

These ODS tables are specific to the types of modeling languages used in the model. This is noted in the last column of the following table.

ODS Table Name	Description	Model
LMEqsAlpha	_EQSALPHA_ vector	LINEQS
LMEqsBeta	_EQSBETA_ matrix	LINEQS
LMEqsGammaSub	_EQSGAMMA_ matrix, excluding entries with error variables in columns	LINEQS
LMEqsNewDep	New rows for expanding _EQSBETA_ and _EQSGAMMA_ matrices	LINEQS
LMEqsNuSub	_EQSNU_ vector, excluding fixed zero means for error variables	LINEQS
LMEqsPhi	_EQSPHI_ matrix	LINEQS
LMEqsPhi11	Upper left partition (exogenous variances and covariances) of the _EQSPHI_ matrix	LINEQS
LMEqsPhi21	Lower left partition (error variances and covariances) of the _EQSPHI_ matrix	LINEQS
LMEqsPhi22	Lower right partition (error variances and covariances) of the _EQSPHI_ matrix	LINEQS

ODS Table Name	Description	Model
LMFactErrv	Vector of error variances	FACTOR
LMFactFcov	Factor covariance matrix	FACTOR
LMFactInte	Intercept vector	FACTOR
LMFactLoad	Factor loading matrix	FACTOR
LMFactMean	Factor mean vector	FACTOR
LMLisAlpha	LISMOD _ALPHA_ vector	LISMOD
LMLisBeta	LISMOD _BETA_ matrix	LISMOD
LMLisGamma	LISMOD _GAMMA_ matrix	LISMOD
LMLisKappa	LISMOD _KAPPA_ vector	LISMOD
LMLisLambdaX	LISMOD _LAMBDAX_ matrix	LISMOD
LMLisLambdaY	LISMOD _LAMBDAY_ matrix	LISMOD
LMLisNuX	LISMOD _NUX_ vector	LISMOD
LMLisNuY	LISMOD _NUY_ vector	LISMOD
LMLisPhi	LISMOD _PHI_ matrix	LISMOD
LMLisPsi	LISMOD _PSI_ matrix	LISMOD
LMLisThetaX	LISMOD _THETAX_ matrix	LISMOD
LMLisThetaY	LISMOD _THETAY_ matrix	LISMOD
LMMstructCov	Covariance matrix	MSTRUCT
LMMstructMean	Mean vector	MSTRUCT
LMRamA	_RAMA_ matrix	RAM
LMRamALower	Lower partitions of the _RAMA_ matrix	RAM
LMRamAUpper	Upper partitions of the _RAMA_ matrix	RAM
LMRamAlpha	_RAMALPHA_ matrix	RAM
LMRamBeta	Upper left partition of the _RAMA_ matrix	RAM
LMRamGamma	Upper right partition of the _RAMA_ matrix	RAM
LMRamNu	_RAMNU_ matrix	RAM
LMRamP	_RAMP_ matrix	RAM
LMRamP11	Upper left partition of the _RAMP_ matrix	RAM
LMRamP21	Lower left partition of the _RAMP_ matrix	RAM
LMRamP22	Lower right partition of the _RAMP_ matrix	RAM
LMRamW	_RAMW_ matrix	RAM

ODS Table Names for Other Modification Indices

The following table shows the ODS tables for the remaining modification indices.

ODS Table Name	Description	Additional Requirement
LagrangeBoundary	LM tests for active boundary constraints	Presence of active boundary constraints
LagrangeDepParmEquality	LM tests for equality constraints in dependent parameters	Presence of equality constraints in dependent parameters
LagrangeEquality	LM tests for equality constraints	Presence of equality constraints in independent parameters
WaldTest	Wald tests for testing existing parameters equaling zeros	At least one insignificant parameter value

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 TCALIS statement: PSHORT, 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 TCALIS or the NLOPTIONS statement. Some restrictions might apply. Additional options and restrictions are noted in the last column.

ODS Table Name	Description	Additional Option Required or Restriction
CovParm	Covariances of parameters	PCOVES[1] or PALL[2], restriction[3]
ConvergenceStatus	Convergence status	
DependParmsResults	Final dependent parameter estimates	Restriction[4]
DependParmsStart	Initial dependent parameter estimates	Restriction[4]
Information	Information matrix	PCOVES[1] or PALL[2], restriction[3]
InitEstMethod	Initial estimation methods	
InputOptions	Optimization options	PALL[2]
IterHist	Iteration history	
IterStart	Iteration start	
IterStop	Iteration stop	
Lagrange	First and second order Lagrange multipliers	PALL[2]
LinCon	Linear constraints	PALL[2], restriction[5]
LinConDel	Deleted constraints	PALL[2], restriction[5]
LinConSol	Linear constraints evaluated at solution	PALL[2], restriction[5]
LinDep	Linear dependencies of parameter estimates	Restriction[6]
ParameterEstimatesResults	Final estimates	
ParameterEstimatesStart	Initial estimates	
ProblemDescription	Problem description	
ProjGrad	Projected gradient	PALL[2]

1. PCOVES option is specified in the PROC TCALIS 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.
6. Existence of parameter dependencies during optimization, but not due to model specification.

ODS Graphics

To request graphics with PROC TCALIS, you must first enable ODS Graphics by specifying the `ods graphics on` statement. See Chapter 21, "Statistical Graphics Using ODS," for more information. In the following table, ODS graph names and the options to display the graphs are listed.

ODS Graph Name	**Plot Description**	**PLOTS= Option**
AsymStdResidualHistogram	Asymptotically standardized residuals	PLOTS=RESIDUALS and RESIDUAL=ASYMSTD, METHOD= is not ULS or DWLS
NormResidualHistogram	Normalized residuals	PLOTS=RESIDUALS and RESIDUAL=NORM
RawResidualHistogram	Raw residuals	PLOTS=RESIDUALS
VarStdResidualHistogram	Variance standardized residuals	PLOTS=RESIDUALS and RESIDUAL=VARSTD

Examples: TCALIS Procedure

Example 88.1: 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). The path diagram of this model is displayed in Figure 88.2. The same model is reanalyzed here by PROC TCALIS. 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.

```
title "Stability of Alienation";
title2 "Data Matrix of WHEATON, MUTHEN, ALWIN & SUMMERS (1977)";
data Wheaton(TYPE=COV);
   _type_ = 'cov';
   input _name_ $ 1-11 Anomie67 Powerless67 Anomie71 Powerless71
                       Education SEI;
   label Anomie67='Anomie (1967)' Powerless67='Powerlessness (1967)'
         Anomie71='Anomie (1971)' Powerless71='Powerlessness (1971)'
         Education='Education'    SEI='Occupational Status Index';
   datalines;
Anomie67      11.834     .        .        .        .       .
Powerless67    6.947    9.364     .        .        .       .
Anomie71       6.819    5.091   12.532     .        .       .
Powerless71    4.783    5.028    7.495    9.986     .       .
Education     -3.839   -3.889   -3.841   -3.625   9.610     .
SEI          -21.899  -18.831  -21.748  -18.775  35.522  450.288
;

ods graphics on;

proc tcalis nobs=932 data=Wheaton plots=residuals;
   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;

ods graphics off;
```

Since no METHOD= option has been used, maximum likelihood estimates are computed by default. In this example, no global display options are specified. This means that PROC TCALIS prints the default set of results.

PROC TCALIS can produce a high-quality residual histogram that is useful for showing the distribution of residuals. To request the residual histogram, you must first enable ODS Graphics by specifying the **ods graphics on** statement, as shown in the preceding statements before the PROC TCALIS statement. Then, the residual histogram is requested by the **plots=residuals** option in the PROC TCALIS statement.

Output 88.1.1 displays the modeling information and variables in the analysis.

Output 88.1.1 Model Specification and Variables

```
                          Stability of Alienation
                          PATH Model Specification

                            The TCALIS Procedure
            Covariance Structure Analysis: Model and Initial Values

                            Modeling Information

                      Data Set            WORK.WHEATON
                      N Obs               932
                      Model Type          PATH


                           Variables in the Model

    Endogenous    Manifest    Anomie67  Anomie71  Education  Powerless67
                              Powerless71  SEI
                  Latent      Alien67  Alien71
    Exogenous     Manifest
                  Latent      SES

                     Number of Endogenous Variables = 8
                     Number of Exogenous Variables  = 1
```

Output 88.1.1 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 88.1.2 echoes the initial specification of the PATH model.

Output 88.1.2 Initial Estimates

```
                         Initial Estimates for PATH List

             ------------Path------------    Parameter      Estimate

             Anomie67       <-    Alien67                     1.00000
             Powerless67    <-    Alien67                     0.83300
             Anomie71       <-    Alien71                     1.00000
             Powerless71    <-    Alien71                     0.83300
             Education      <-    SES                         1.00000
             SEI            <-    SES        lambda                 .
             Alien67        <-    SES        gamma1                 .
             Alien71        <-    SES        gamma2                 .
             Alien71        <-    Alien67    beta                   .

                    Initial Estimates for Variance Parameters

               Variance
               Type         Variable       Parameter      Estimate

               Error        Anomie67       theta1                .
                            Powerless67    theta2                .
                            Anomie71       theta1                .
                            Powerless71    theta2                .
                            Education      theta3                .
                            SEI            theta4                .
                            Alien67        psi1                  .
                            Alien71        psi2                  .
               Exogenous    SES            phi                   .

                 Initial Estimates for Covariances Among Errors

              Error of       Error of       Parameter      Estimate

              Anomie67       Anomie71       theta5                .
              Powerless67    Powerless71    theta5                .
```

In Output 88.1.2, numerical values for estimates are the initial values you input in the model specification. If the associated parameter name for a numerical estimate is blank, it means 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 with the fixed values given. For numerical estimates with parameter names given, the numerical values serve as initial values, which would be changed during the estimation. In Output 88.1.2, you actually do not have this kind of specification. All free parameters specified in the model are with missing initial values, denoted by '.'. For example, lambda, gamma1, theta1, and psi1, among others, are free parameters without initial values given. The initial values of these parameters will be generated by PROC TCALIS automatically.

You can examine this output to ensure that the desired model is being analyzed. PROC TCALIS 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 88.1.3.

Output 88.1.3 Descriptive Statistics

```
                               Simple Statistics

                     Variable                              Mean        Std Dev

      Anomie67       Anomie (1967)                            0        3.44006
      Powerless67    Powerlessness (1967)                     0        3.06007
      Anomie71       Anomie (1971)                            0        3.54006
      Powerless71    Powerlessness (1971)                     0        3.16006
      Education      Education                                0        3.10000
      SEI            Occupational Status Index                0       21.21999
```

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. They could be provided by users or automatically generated by PROC TCALIS. As shown in Output 88.1.2, you did not provide any initial estimates for all free parameters. PROC TCALIS 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 88.1.4.

Output 88.1.4 Optimization Starting Point

```
                          Initial Estimation Methods

                         1    Instrumental Variables Method
                         2    McDonald Method
                         3    Two-Stage Least Squares

                              Optimization Start
                              Parameter Estimates

                      N    Parameter      Estimate      Gradient

                      1    lambda          4.99508      -0.00206
                      2    gamma1         -0.62322      -0.04069
                      3    gamma2         -0.20437      -0.03816
                      4    beta            0.66589       0.03789
                      5    theta1          3.51433      -0.00409
                      6    theta2          3.65991       0.01182
                      7    theta3          2.49860      -0.00578
                      8    theta4        272.85274     0.0000194
                      9    psi1            5.57764      -0.00217
                     10    psi2            3.79636      -0.00935
                     11    phi             7.11140       0.00108
                     12    theta5          0.45298      -0.06463

                    Value of Objective Function = 0.0365979443
```

In this example, instrumental Variable Method, method by McDonald and Hartmann, and 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 88.1.5 displays the optimization information, including technical details, iteration history and convergence status.

Output 88.1.5 Optimization

```
              Parameter Estimates                                  12
              Functions (Observations)                             21

                                Optimization Start

  Active Constraints                     0    Objective Function    0.0365979443
  Max Abs Gradient Element    0.0646338767    Radius                           1

                                                                         Actual
                                                       Max Abs             Over
          Rest     Func       Act     Objective  Obj Fun Gradient          Pred
  Iter    arts    Calls       Con      Function   Change  Element  Lambda  Change

     1       0        4         0       0.01453   0.0221  0.00142       0   1.013
     2       0        6         0       0.01448 0.000046 0.000249       0   1.001
     3       0        8         0       0.01448 1.007E-7 4.717E-6       0   1.006

                               Optimization Results

 Iterations                             3   Function Calls                     11
 Jacobian Calls                         5   Active Constraints                  0
 Objective Function          0.0144844814   Max Abs Gradient Element  4.7172823E-6
 Lambda                                 0   Actual Over Pred Change   1.0060912391
 Radius                       0.001390392

 Convergence criterion (ABSGCONV=0.00001) satisfied.
```

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 TCALIS issues an error message.

The fit summary statistics are displayed in Output 88.1.6. By default, PROC TCALIS displays all available fit indices and modeling information.

Output 88.1.6 Fit Summary

```
                                    Fit Summary

       Modeling Info        N Observations                                932
                            N Variables                                     6
                            N Moments                                      21
                            N Parameters                                   12
                            N Active Constraints                            0
                            Independence Model Chi-Square           2131.4327
                            Independence Model Chi-Square DF               15
       Absolute Index       Fit Function                               0.0145
                            Chi-Square                                13.4851
                            Chi-Square DF                                   9
                            Pr > Chi-Square                            0.1419
                            Z-Test of Wilson & Hilferty                1.0754
                            Hoelter Critical N                           1170
                            Root Mean Square Residual (RMSR)           0.2281
                            Standardized RMSR (SRMSR)                  0.0150
                            Goodness of Fit Index (GFI)                0.9953
       Parsimony Index      Adjusted GFI (AGFI)                        0.9890
                            Parsimonious GFI                           0.5972
                            RMSEA Estimate                             0.0231
                            RMSEA Lower 90% Confidence Limit                .
                            RMSEA Upper 90% Confidence Limit           0.0470
                            Probability of Close Fit                   0.9705
                            ECVI Estimate                              0.0405
                            ECVI Lower 90% Confidence Limit                 .
                            ECVI Upper 90% Confidence Limit            0.0556
                            Akaike Information Criterion              -4.5149
                            Bozdogan CAIC                            -57.0509
                            Schwarz Bayesian Criterion               -48.0509
                            McDonald Centrality                        0.9976
       Incremental Index    Bentler Comparative Fit Index              0.9979
                            Bentler-Bonett NFI                         0.9937
                            Bentler-Bonett Non-normed Index            0.9965
                            Bollen Normed Index Rho1                   0.9895
                            Bollen Non-normed Index Delta2             0.9979
                            James et al. Parsimonious NFI              0.5962
```

First, the fit summary table starts with some basic modeling information, as shown in Output 88.1.6. 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 are interpreted without referring to any baseline model and without adjusting model parsimony. 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 larger than the conventional 0.05 value. From a statistical hypothesis testing point of view, this model cannot be rejected. The Z-test of Wilson and Hilferty is also insignificant at $\alpha = .05$, which echoes the result of the chi-square test. Other absolute indices can also be consulted. Although it seems that there are no clear conventional levels for these indices to indicate an acceptable model fit, you can always use these indices to compare the relative fit between different 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 improve the model fit indicated by these indices. Models with large numbers of parameters are penalized. There is no uniform way to interpret all these indices. However, for the relatively well-known RMSEA estimate, by convention values under 0.05 indicate good fit. The RMSEA value for this example is 0.0231, and so this is a very good 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 comparison with a baseline model, usually the independence model where all mainfest variables are assumed to be uncorrelated. The independence 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 independence model is 2131.43, with 15 degrees of freedom. The incremental indices show how well the hypothesized model improves over the independence 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 larger 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 do 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 RMSR, 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.

PROC TCALIS can perform a detailed residual analysis. Large residuals might indicate misspecification of the model. In Output 88.1.7, raw residuals are reported and ranked.

Output 88.1.7 Raw Residuals and Ranking

```
                                Raw Residual Matrix

                                                Anomie67   Powerless67      Anomie71

Anomie67      Anomie (1967)                     -0.06997       0.03642      -0.01116
Powerless67   Powerlessness (1967)               0.03642       0.01261       0.15600
Anomie71      Anomie (1971)                     -0.01116       0.15600      -0.08381
Powerless71   Powerlessness (1971)              -0.15200       0.01135      -0.00854
Education     Education                          0.32892      -0.41712       0.22464
SEI           Occupational Status Index          0.47786      -0.19108       0.07976

                                Raw Residual Matrix

                                             Powerless71     Education           SEI

Anomie67      Anomie (1967)                     -0.15200       0.32892       0.47786
Powerless67   Powerlessness (1967)               0.01135      -0.41712      -0.19108
Anomie71      Anomie (1971)                     -0.00854       0.22464       0.07976
Powerless71   Powerlessness (1971)               0.14067      -0.23832      -0.59248
Education     Education                         -0.23832             0    4.03148E-6
SEI           Occupational Status Index         -0.59248    4.03148E-6     0.0000221

           Average Absolute Residual                           0.153940
           Average Off-diagonal Absolute Residual              0.195044

                   Rank Order of the 10 Largest Raw Residuals

                    Var1            Var2            Residual

                    SEI             Powerless71     -0.59248
                    SEI             Anomie67         0.47786
                    Education       Powerless67     -0.41712
                    Education       Anomie67         0.32892
                    Education       Powerless71     -0.23832
                    Education       Anomie71         0.22464
                    SEI             Powerless67     -0.19108
                    Anomie71        Powerless67      0.15600
                    Powerless71     Anomie67        -0.15200
                    Powerless71     Powerless71      0.14067
```

Because of the differential scaling of the variables, it is usually more useful to examine the standardized residuals instead. In Output 88.1.8, for example, the table for the 10 largest asymptotically standardized residuals is displayed.

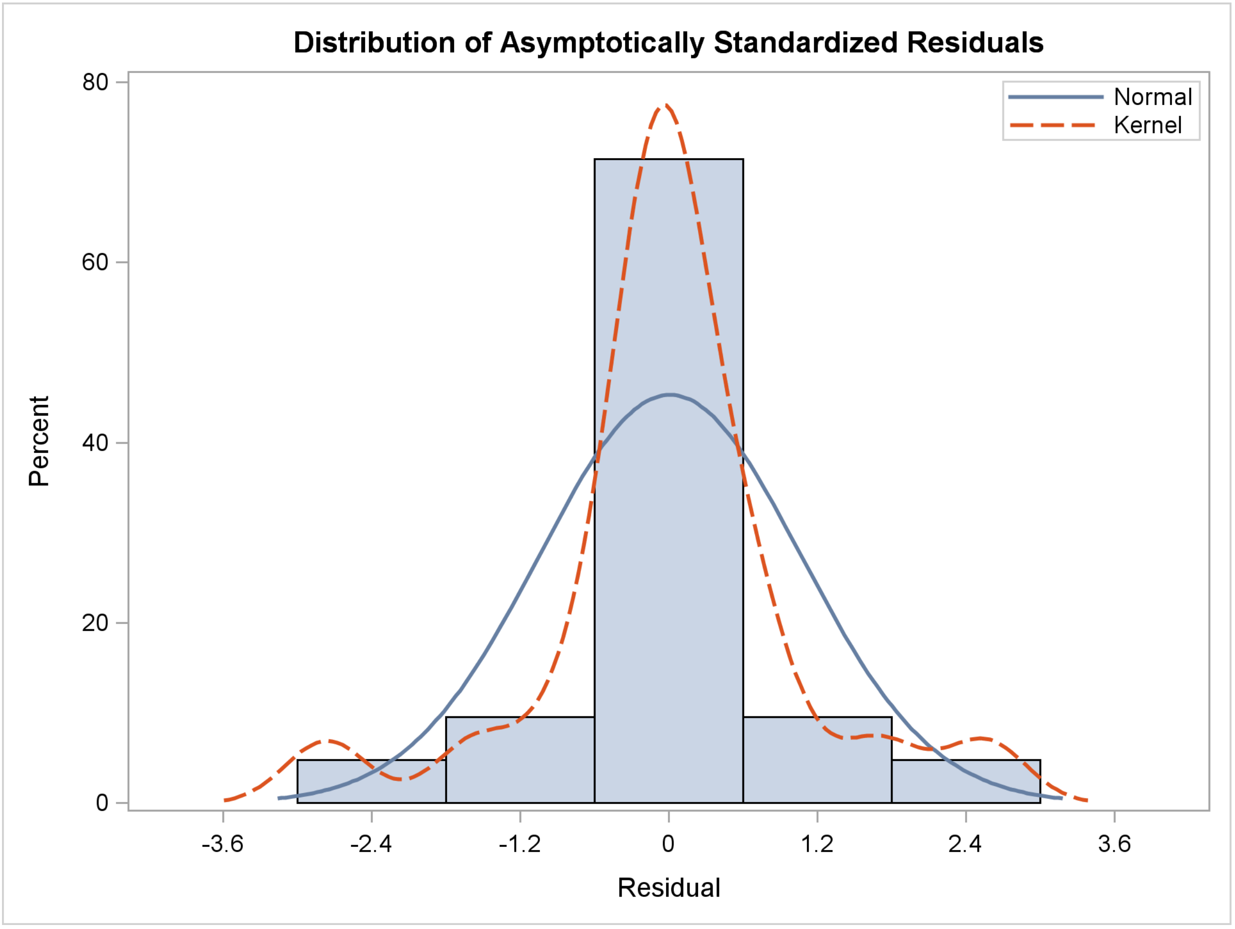
Output 88.1.8 Asymptotically Standardized Residuals and Ranking

```
                 Asymptotically Standardized Residual Matrix

                                          Anomie67   Powerless67      Anomie71

Anomie67      Anomie (1967)               -0.30882       0.52686      -0.05619
Powerless67   Powerlessness (1967)         0.52686       0.05464       0.87613
Anomie71      Anomie (1971)               -0.05619       0.87613      -0.35460
Powerless71   Powerlessness (1971)        -0.86507       0.05735      -0.12169
Education     Education                    2.55338      -2.76371       1.69781
SEI           Occupational Status Index    0.46484      -0.17015       0.07009

                 Asymptotically Standardized Residual Matrix

                                       Powerless71     Education           SEI

Anomie67      Anomie (1967)               -0.86507       2.55338       0.46484
Powerless67   Powerlessness (1967)         0.05735      -2.76371      -0.17015
Anomie71      Anomie (1971)               -0.12169       1.69781       0.07009
Powerless71   Powerlessness (1971)         0.58521      -1.55750      -0.49608
Education     Education                   -1.55750             0             0
SEI           Occupational Status Index   -0.49608             0             0

            Average Standardized Residual                  0.646672
            Average Off-diagonal Standardized Residual     0.818456

        Rank Order of the 10 Largest Asymptotically Standardized Residuals

                    Var1           Var2             Residual

                    Education      Powerless67      -2.76371
                    Education      Anomie67          2.55338
                    Education      Anomie71          1.69781
                    Education      Powerless71      -1.55750
                    Anomie71       Powerless67       0.87613
                    Powerless71    Anomie67         -0.86507
                    Powerless71    Powerless71       0.58521
                    Powerless67    Anomie67          0.52686
                    SEI            Powerless71      -0.49608
                    SEI            Anomie67          0.46484
```

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 88.1.9, which also shows the normal and kernel approximations.

Output 88.1.9 Distribution of Asymptotically Standardized Residuals



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.

Estimation results are shown in Output 88.1.10.

Output 88.1.10 Estimation Results

```
                              PATH List

                                                        Standard
-----------Path-----------   Parameter     Estimate        Error      t Value

Anomie67       <-   Alien67                  1.00000
Powerless67    <-   Alien67                  0.83300
Anomie71       <-   Alien71                  1.00000
Powerless71    <-   Alien71                  0.83300
Education      <-   SES                      1.00000
SEI            <-   SES      lambda          5.36883      0.43371     12.37880
Alien67        <-   SES      gamma1         -0.62994      0.05634    -11.18092
Alien71        <-   SES      gamma2         -0.24086      0.05489     -4.38836
Alien71        <-   Alien67  beta            0.59312      0.04678     12.67884
```

Output 88.1.10 *continued*

```
                              Variance Parameters

Variance                                                        Standard
Type          Variable       Parameter       Estimate           Error        t Value

Error         Anomie67       theta1           3.60796          0.20092      17.95717
              Powerless67    theta2           3.59488          0.16448      21.85563
              Anomie71       theta1           3.60796          0.20092      17.95717
              Powerless71    theta2           3.59488          0.16448      21.85563
              Education      theta3           2.99366          0.49861       6.00398
              SEI            theta4         259.57639         18.31151      14.17559
              Alien67        psi1             5.67046          0.42301      13.40500
              Alien71        psi2             4.51479          0.33532      13.46394
Exogenous     SES            phi              6.61634          0.63914      10.35190

                            Covariances Among Errors

                                                                Standard
  Error of      Error of      Parameter       Estimate          Error        t Value

  Anomie67      Anomie71      theta5           0.90580         0.12167       7.44472
  Powerless67   Powerless71   theta5           0.90580         0.12167       7.44472
```

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 88.1.10 are significantly different from zero, supporting the presence of these parameters in the model.

Squared multiple correlations are shown in Output 88.1.11.

Output 88.1.11 Squared Multiple Correlations

```
                         Squared Multiple Correlations

                                Error          Total
               Variable       Variance       Variance     R-Square

               Anomie67        3.60796       11.90397       0.6969
               Anomie71        3.60796       12.61581       0.7140
               Education       2.99366        9.61000       0.6885
               Powerless67     3.59488        9.35139       0.6156
               Powerless71     3.59488        9.84533       0.6349
               SEI           259.57639      450.28798       0.4235
               Alien67         5.67046        8.29601       0.3165
               Alien71         4.51479        9.00786       0.4988
```

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 88.1.10, the path coefficient for path Education <– SES is fixed at one, while the path coefficient for path SEI <– SES is 5.369. It would be very simple-minded to conclude that the effect of SES on SEI is larger 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 TCALIS standardizes your results in Output 88.1.12.

Output 88.1.12 Standardized Results

```
                        Standardized Results for PATH List

                                                             Standard
-----------Path-----------     Parameter      Estimate          Error       t Value

Anomie67       <-    Alien67                    0.83481        0.01093      76.35313
Powerless67    <-    Alien67                    0.78459        0.01163      67.47756
Anomie71       <-    Alien71                    0.84499        0.01031      81.97956
Powerless71    <-    Alien71                    0.79678        0.01107      71.96263
Education      <-    SES                        0.82975        0.03172      26.15990
SEI            <-    SES       lambda           0.65079        0.03019      21.55331
Alien67        <-    SES       gamma1          -0.56257        0.03456     -16.27961
Alien71        <-    SES       gamma2          -0.20642        0.04483      -4.60430
Alien71        <-    Alien67   beta             0.56920        0.04066      14.00001
```

Output 88.1.12 *continued*

```
                 Standardized Results for Variance Parameters

Variance                                                 Standard
Type         Variable       Parameter      Estimate         Error        t Value

Error        Anomie67       theta1          0.30309       0.01825       16.60309
             Powerless67    theta2          0.38442       0.01825       21.06948
             Anomie71       theta1          0.28599       0.01742       16.41782
             Powerless71    theta2          0.36514       0.01764       20.69424
             Education      theta3          0.31152       0.05264        5.91822
             SEI            theta4          0.57647       0.03930       14.66804
             Alien67        psi1            0.68352       0.03888       17.57968
             Alien71        psi2            0.50121       0.03321       15.08974
Exogenous    SES            phi             1.00000

               Standardized Results for Covariances Among Errors

                                                         Standard
  Error of      Error of      Parameter      Estimate       Error       t Value

  Anomie67      Anomie71      theta5          0.07391     0.01013       7.29574
  Powerless67   Powerless71   theta5          0.09440     0.01274       7.40922
```

Now, the standardized path coefficient for path Education <– SES is 0.830, while the standardized path coefficient for path SEI <– SES is 0.651. So the standardized effect of SES on SEI is actually smaller than that of SES on Education.

Furthermore, in PROC TCALIS 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 TCALIS 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 TCALIS 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 TCALIS are rescaled so as to keep the mathematical consistency of the model.

Essentially, in PROC TCALIS only variances of mainfest 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 88.1.12, 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 6907 for the logic of the standardization scheme adopted by PROC TCALIS.

In appearance, the standardized solution is like a correlational analysis on the standardized manifest variables with standardized exogenous latent factors. Unfortunately, this statement is oversimplified, 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 TCALIS provides the standardized solution so that users can interpret the standardized estimates whenever they find them appropriate.

Example 88.2: 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 TCALIS 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:

$$Q_t(demand) = \alpha_1 + \beta_1 P_t + \gamma_1 D_t$$

$$Q_t(supply) = \alpha_2 + \beta_2 P_t + \gamma_2 F_t + \gamma_3 Y_t$$

for $t = 1, \ldots, 20$.

To analyze this model in PROC TCALIS, 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 TCALIS:

```
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
;
proc tcalis data=food pshort nostand;
   lineqs
      Q = alpha1 Intercept + beta1  P  + gamma1 D + E1,
      P = theta1 Intercept + theta2 Q  + theta3 F + theta4 Y + E2;
   std
      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 STD 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 TCALIS 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 TCALIS might produce a large amount of output, using these printing options make your output more concise and readable. Whenever appropriate, it is recommended that you consider using these printing options.

The estimated equations are shown in Output 88.2.1.

Output 88.2.1 Linear Equations

```
                             Linear Equations

     Q        =    93.6193*Intercept + -0.2295*P         +  0.3100*D
     Std Err        7.5748 alpha1       0.0923 beta1        0.0448 gamma1
     t Value       12.3592             -2.4856              6.9186

                  +  1.0000 E1



                             Linear Equations

     P        =     -218.9*Intercept +  4.2140*Q         + -0.9305*F
     Std Err         137.7 theta1       1.7540 theta2       0.3960 theta3
     t Value       -1.5897              2.4025             -2.3500

                  + -1.5579*Y         +  1.0000 E2
                     0.6650 theta4
                    -2.3429
```

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 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 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 88.2.2.

Output 88.2.2 Variance, Covariance, and Mean Parameters

```
                 Estimates for Variances of Exogenous Variables

Variable                                                  Standard
Type            Variable     Parameter       Estimate        Error        t Value

Error           E1           eps1             3.51274      1.20204        2.92233
                E2           eps2           105.06749     83.89446        1.25238
Observed        D            _Add1          139.96029     45.40911        3.08221
                F            _Add2          161.51355     52.40192        3.08221
                Y            _Add3           35.00000     11.35550        3.08221

                      Covariances Among Exogenous Variables

                                                   Standard
     Var1      Var2      Parameter       Estimate        Error        t Value

     E1        E2        eps3           -18.87270      8.77951       -2.14963
     F         D         _Add4           74.02539     38.44699        1.92539
     Y         D         _Add5           22.99211     16.90102        1.36040
     Y         F         _Add6          -21.58158     17.94544       -1.20262

                                 Mean Parameters

 Variable                                                Standard
 Type           Variable     Parameter       Estimate        Error        t Value

 Observed       D            _Add7           97.53500      2.71410       35.93643
                F            _Add8           96.62500      2.91560       33.14071
                Y            _Add9           10.50000      1.35724        7.73628
```

Parameters with a name prefix _Add are added automatically by PROC TCALIS. These parameters are added as free parameters to complete the model specification. In PROC TCALIS, variances and covariances among the set of exogenous manifest variables must be parameters. You either specify them explicitly or let the TCALIS 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 88.2.3.

Output 88.2.3 Squared Multiple Correlations

```
                    Squared Multiple Correlations

                          Error          Total
         Variable      Variance       Variance      R-Square

         Q              3.51274       14.11128        0.7511
         P            105.06749       35.11850       -1.9918
```

For endogenous variable P, the R-square is -1.9918, which is obviously an invalid value. In fact, because there are correlated errors (between E1 and E2) 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.

If you are interested in estimating the parameters in the original econometric model (that is, α_2, β_2, γ_2, and γ_3), the previous reparameterized LINEQS model does not serve your purpose well enough. However, using the relations between these original parameters with the θ parameters in the reparameterized LINEQS model, you can set up some "super-parameters" in the LINEQS model as follows:

```
   proc tcalis data=Food pshort nostand;
      lineqs
         Q = alpha1 Intercept + beta1  P  + gamma1 D + E1,
         P = theta1 Intercept + theta2 Q  + theta3 F + theta4 Y + E2;
      std
         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, theta1, theta2, theta3, and theta4 are no longer independent parameters in the model, as they are in the previous reparameterized model. Instead, alpha2, beta2, gamma2, and gamma3 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 88.2.4.

Output 88.2.4 Additional Parameters

```
                          Additional Parameters

                                                  Standard
   Type            Parameter       Estimate          Error        t Value

   Independent     alpha2          51.94452       11.70002        4.43969
                   beta2            0.23731        0.09877        2.40262
                   gamma2           0.22082        0.04161        5.30695
                   gamma3           0.36971        0.07060        5.23649
```

You can now interpret the results in terms of the original econometric parameterization. As shown in Output 88.2.4, 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.

Example 88.3: A Direct Covariance Structures Model

In the section "Direct Covariance Structures Analysis" on page 6704, 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:

$$\Sigma = \begin{pmatrix} \phi_1 & \theta_1 & \theta_2 & \theta_1 \\ \theta_1 & \phi_2 & \theta_l & \theta_3 \\ \theta_2 & \theta_1 & \phi_1 & \theta_1 \\ \theta_1 & \theta_3 & \theta_1 & \phi_2 \end{pmatrix}$$

where:

ϕ_1:	variance of anomie
ϕ_2:	variance of powerlessness
θ_1:	covariance between anomie and powerlessness
θ_2:	covariance between anomie measures
θ_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 88.1), structural equations (see Example 88.2), 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 ϕ_1, and both Powerless67 and Powerless71 have the same variance parameter ϕ_2. These two parameters, ϕ_1 and ϕ_2, are hypothesized on the diagonal of the covariance matrix Σ. In the same structured covariance matrix, θ_1 represents the covariance between Anomie and Powerless, without regard to the time of measurement. The θ_2 parameter represents the covariance between the Anomie measures, or the reliability of the Anomie measure. Similarly, the θ_2 parameter represents the covariance between the Powerless measures, or the reliability of the Anomie measure.

As explained in the section "Direct Covariance Structures Analysis" on page 6704, you can use the MSTRUCT modeling language to specify the hypothesized covariance structures directly, as shown in the following statements:

```
proc tcalis 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 TCALIS 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 TCALIS determine all the initial estimates.

In the PROC TCALIS 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.

The entire printed output is shown in Output 88.3.1.

Output 88.3.1 Testing Direct Covariance Structures

```
                              Fit Summary

                    Chi-Square              221.5798
                    Chi-Square DF                  5
                    Pr > Chi-Square           0.0000
```

As you can see, the displayed output is very precise. It contains only a fit summary table with three statistics. The p-value for the mode 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 with more parameters is needed to better account for the variances and covariances of the variables.

All fit indices are saved in the savefit data set. To view it, you can use the following statement:

```
proc print data=savefit;
run;
```

All indices, their types and values are shown in Output 88.3.2.

Output 88.3.2 Saved Fit Indices

```
                     Analysis of Direct Covariance Structures
                      Testing Model by the MSTRUCT Language

                                                                    Fit
Obs      _TYPE_        FitIndex                                   Value    PrintChar

  1    ModelInfo      N Observations                            932.00          932
  2    ModelInfo      N Variables                                 4.00            4
  3    ModelInfo      N Moments                                  10.00           10
  4    ModelInfo      N Parameters                                5.00            5
  5    ModelInfo      N Active Constraints                        0.00            0
  6    ModelInfo      Independence Model Chi-Square            1563.94    1563.9442
  7    ModelInfo      Independence Model Chi-Square DF            6.00            6
  8    Absolute       Fit Function                                0.24       0.2380
  9    Absolute       Chi-Square                                221.58     221.5798
 10    Absolute       Chi-Square DF                               5.00            5
 11    Absolute       Pr > Chi-Square                             0.00       0.0000
 12    Absolute       Elliptic Corrected Chi-Square                 .             .
 13    Absolute       Pr > Elliptic Corr. Chi-Square                .             .
 14    Absolute       Z-Test of Wilson & Hilferty                12.25      12.2533
 15    Absolute       Hoelter Critical N                         48.00           48
 16    Absolute       Root Mean Square Residual (RMSR)            0.76       0.7649
 17    Absolute       Standardized RMSR (SRMSR)                   0.07       0.0701
 18    Absolute       Goodness of Fit Index (GFI)                 0.90       0.9036
 19    Parsimony      Adjusted GFI (AGFI)                         0.81       0.8071
 20    Parsimony      Parsimonious GFI                            0.75       0.7530
 21    Parsimony      RMSEA Estimate                              0.22       0.2157
 22    Parsimony      RMSEA Lower 90% Confidence Limit            0.19       0.1920
 23    Parsimony      RMSEA Upper 90% Confidence Limit            0.24       0.2404
 24    Parsimony      Probability of Close Fit                    0.00       0.0000
 25    Parsimony      ECVI Estimate                               0.25       0.2488
 26    Parsimony      ECVI Lower 90% Confidence Limit             0.20       0.2003
 27    Parsimony      ECVI Upper 90% Confidence Limit             0.31       0.3053
 28    Parsimony      Akaike Information Criterion              211.58     211.5798
 29    Parsimony      Bozdogan CAIC                             182.39     182.3932
 30    Parsimony      Schwarz Bayesian Criterion                187.39     187.3932
 31    Parsimony      McDonald Centrality                         0.89       0.8903
 32    Incremental    Bentler Comparative Fit Index               0.86       0.8610
 33    Incremental    Bentler-Bonett NFI                          0.86       0.8583
 34    Incremental    Bentler-Bonett Non-normed Index             0.83       0.8332
 35    Incremental    Bollen Normed Index Rho1                    0.83       0.8300
 36    Incremental    Bollen Non-normed Index Delta2              0.86       0.8611
 37    Incremental    James et al. Parsimonious NFI               0.72       0.7153
```

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:

```
proc tcalis nobs=932 data=Wheaton psummary;
   fitindex on(only)=[chisq df probchi] outfit=savefit;
   mstruct
      var = Anomie67 Powerless67 Anomie71 Powerless71;
   matrix _COV_ [1,1] = phi1 phi2 phi1 phi2;
                [2, ] = theta1,
                [3, ] = theta2 theta1,
                [4, ] = theta1 theta3 theta1;
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 6776 for more details about this kind of shorthand notation.

Example 88.4: 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 and the PROC TCALIS specification for the analysis:

```
title "Confirmatory Factor Analysis Using the FACTOR Modeling Language";
title2 "Cognitive Data";
data cognitive1(type=cov);
   _type_='cov';
   input _name_ $ reading1 reading2 reading3 math1 math2 math3
         writing1 writing2 writing3;
   datalines;
reading1 83.024    .       .      .      .      .      .      .      .
reading2 50.924 108.243    .      .      .      .      .      .      .
reading3 62.205  72.050 99.341    .      .      .      .      .      .
math1    22.522  22.474 25.731 82.214    .      .      .      .      .
math2    14.157  22.487 18.334 64.423 96.125    .      .      .      .
math3    22.252  20.645 23.214 49.287 58.177 88.625    .      .      .
writing1 33.433  42.474 41.731 25.318 14.254 27.370 90.734    .      .
writing2 24.147  20.487 18.034 22.106 26.105 22.346 53.891 96.543    .
writing3 13.340  20.645 23.314 19.387 28.177 38.635 55.347 52.999 98.445
;
```

```
proc tcalis data=cognitive1 nobs=64 modification;
   factor
      Read_Factor   -> reading1-reading3  = load1-load3,
      Math_Factor   -> math1-math3        = load4-load6,
      Write_Factor  -> writing1-writing3  = load7-load9;
   pvar
      Read_Factor Math_Factor Write_Factor = 3 * 1.,
      reading1-reading3 math1-math3 writing1-writing3 = errvar1-errvar9;
run;
```

In the PROC TCALIS statement, the number of observations is specified with the NOBS= option. The MODIFICATION option in the same statement requests model modification indices be computed.

The FACTOR modeling language is the most handy tool for specifying confirmatory factor models. The FACTOR statement is used to invoke this 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 –>. Then you specify the observed variables that have nonzero loadings on the factor. Next, an equal sign is used to signify the specification of the loading parameters that follow. The loading parameters can be names (parameters without initial estimates), numbers (fixed values), or names followed by parenthesized numbers (parameter with initial values). 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, yielding a total of nine loading parameters named load1–load9. No initial estimates are specified for them. They are computed by PROC TCALIS.

In the PVAR statement, you specify the variances of the factors and the error variances for the observed variables. The factor variances in this model are fixed at ones for identification purposes. The error variances for the observed variables are free parameters without initial estimates, named errvar1–errvar9, respectively.

The covariances of the factors are not specified in the model, meaning that the factors are uncorrelated by default. Certainly, this might not be reasonable. But for illustration purposes, this uncorrelated factor model is fitted. With the MODIFICATION option in the PROC TCALIS statement, LM (Lagrange Multiplier) tests are conducted. The results of LM tests can suggest the inclusion of additional parameters for a better model fit. If the uncorrelated factor model is indeed unreasonable, it is shown in the results of the LM tests.

In Output 88.4.1, the initial model specification is echoed in matrix form. The observed variables and factors are also displayed.

Output 88.4.1 Uncorrelated Factor Model Specification

```
                          Variables in the Model

        Variables    reading1  reading2  reading3  math1  math2  math3
                     writing1  writing2  writing3
        Factors      Read_Factor  Math_Factor  Write_Factor

                         Number of Variables = 9
                         Number of Factors   = 3

                      Initial Factor Loading Matrix

                        Read_Factor        Math_Factor       Write_Factor

          reading1                .                  0                  0
                            [load1]

          reading2                .                  0                  0
                            [load2]

          reading3                .                  0                  0
                            [load3]

          math1                   0                  .                  0
                                               [load4]

          math2                   0                  .                  0
                                               [load5]

          math3                   0                  .                  0
                                               [load6]

          writing1                0                  0                  .
                                                                  [load7]

          writing2                0                  0                  .
                                                                  [load8]

          writing3                0                  0                  .
                                                                  [load9]

                       Initial Factor Covariance Matrix

                          Read_Factor        Math_Factor       Write_Factor

        Read_Factor            1.0000                  0                  0
        Math_Factor                 0             1.0000                  0
        Write_Factor                0                  0             1.0000
```

Output 88.4.1 *continued*

```
                              Initial Error Variances

                        Variable     Parameter       Estimate

                        reading1     errvar1                .
                        reading2     errvar2                .
                        reading3     errvar3                .
                        math1        errvar4                .
                        math2        errvar5                .
                        math3        errvar6                .
                        writing1     errvar7                .
                        writing2     errvar8                .
                        writing3     errvar9                .
```

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. No initial estimates were specified, as indicated by the missing values '.'.

In Output 88.4.2, initial estimates are generated by the instrumental variable method and the McDonald method.

Output 88.4.2 Optimization of the Uncorrelated Factor Model: Initial Estimates

```
                             Initial Estimation Methods

                            1    Instrumental Variables Method
                            2    McDonald Method
```

Output 88.4.2 *continued*

```
                         Optimization Start
                         Parameter Estimates

                N    Parameter      Estimate      Gradient

                1    load1           7.15372       0.00851
                2    load2           7.80225      -0.00170
                3    load3           8.70856      -0.00602
                4    load4           7.68637       0.00272
                5    load5           8.01765      -0.01096
                6    load6           7.05012       0.00932
                7    load7           8.76776    -0.0009955
                8    load8           5.96161      -0.01335
                9    load9           7.23168       0.01665
               10    errvar1        31.84831      -0.00179
               11    errvar2        47.36790     0.0003461
               12    errvar3        23.50199       0.00257
               13    errvar4        23.13374    -0.0008384
               14    errvar5        31.84224       0.00280
               15    errvar6        38.92075      -0.00167
               16    errvar7        13.86035      -0.00579
               17    errvar8        61.00217       0.00115
               18    errvar9        46.14784      -0.00300

            Value of Objective Function = 0.9103815918
```

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 88.4.3.

Output 88.4.3 Optimization of the Uncorrelated Factor Model: Iteration Summary

```
                                                                  Actual
                                                  Max Abs           Over
         Rest    Func      Act   Objective  Obj Fun Gradient          Pred
Iter     arts   Calls      Con    Function   Change  Element  Lambda  Change

   1        0       4        0     0.78792   0.1225  0.00175       0   0.932
   2        0       6        0     0.78373  0.00419 0.000037       0   1.051
   3        0       8        0     0.78373 5.087E-7 3.715E-9       0   1.001

                          Optimization Results

Iterations                          3  Function Calls                    11
Jacobian Calls                      5  Active Constraints                 0
Objective Function        0.783733415  Max Abs Gradient Element  3.7146571E-9
Lambda                              0  Actual Over Pred Change   1.0006660673
Radius                   0.0025042942

Convergence criterion (ABSGCONV=0.00001) satisfied.
```

The fit summary is shown in Output 88.4.4.

Output 88.4.4 Fit of the Uncorrelated Factor Model

```
                                  Fit Summary

      Modeling Info        N Observations                              64
                           N Variables                                  9
                           N Moments                                   45
                           N Parameters                                18
                           N Active Constraints                         0
                           Independence Model Chi-Square         272.0467
                           Independence Model Chi-Square DF            36
      Absolute Index       Fit Function                            0.7837
                           Chi-Square                             49.3752
                           Chi-Square DF                               27
                           Pr > Chi-Square                         0.0054
                           Z-Test of Wilson & Hilferty             2.5474
                           Hoelter Critical N                          53
                           Root Mean Square Residual (RMSR)       19.5739
                           Standardized RMSR (SRMSR)               0.2098
                           Goodness of Fit Index (GFI)             0.8555
      Parsimony Index      Adjusted GFI (AGFI)                     0.7592
                           Parsimonious GFI                        0.6416
                           RMSEA Estimate                          0.1147
                           RMSEA Lower 90% Confidence Limit        0.0617
                           RMSEA Upper 90% Confidence Limit        0.1646
                           Probability of Close Fit                0.0271
                           ECVI Estimate                           1.4630
                           ECVI Lower 90% Confidence Limit         1.2069
                           ECVI Upper 90% Confidence Limit         1.8687
                           Akaike Information Criterion           -4.6248
                           Bozdogan CAIC                         -89.9146
                           Schwarz Bayesian Criterion            -62.9146
                           McDonald Centrality                     0.8396
      Incremental Index    Bentler Comparative Fit Index           0.9052
                           Bentler-Bonett NFI                      0.8185
                           Bentler-Bonett Non-normed Index         0.8736
                           Bollen Normed Index Rho1                0.7580
                           Bollen Non-normed Index Delta2          0.9087
                           James et al. Parsimonious NFI           0.6139
```

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 non-normed 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 TCALIS statement has been used to request for computing the LM test indices for model modifications. The results are shown in Output 88.4.5.

Output 88.4.5 Lagrange Multiplier Tests

```
             Rank Order of the 10 Largest LM Stat for Factor Loadings

                                                                      Parm
        Variable     Factor              LM Stat     Pr > ChiSq      Change

        writing1     Read_Factor         9.76596         0.0018     2.95010
        math3        Write_Factor        3.58077         0.0585     1.89703
        math1        Read_Factor         2.15312         0.1423     1.17976
        writing3     Math_Factor         1.87637         0.1707     1.41298
        math3        Read_Factor         1.02954         0.3103     0.95427
        reading2     Write_Factor        0.91230         0.3395     0.99933
        writing2     Math_Factor         0.86221         0.3531     0.95672
        reading1     Write_Factor        0.63403         0.4259     0.73916
        math1        Write_Factor        0.55602         0.4559     0.63906
        reading2     Math_Factor         0.55362         0.4568     0.74628

         Rank Order of the 3 Largest LM Stat for Covariances of Factors

                                                                        Parm
      Var1             Var2                LM Stat     Pr > ChiSq      Change

      Write_Factor     Read_Factor         8.95268         0.0028     0.44165
      Write_Factor     Math_Factor         7.07904         0.0078     0.40132
      Math_Factor      Read_Factor         4.61896         0.0316     0.30411

     Rank Order of the 10 Largest LM Stat for Error Variances and Covariances

          Error        Error                                             Parm
          of           of             LM Stat     Pr > ChiSq           Change

          writing1     math2          5.45986         0.0195        -13.16822
          writing1     math1          5.05573         0.0245         12.32431
          writing3     math3          3.93014         0.0474         13.59149
          writing3     math1          2.83209         0.0924         -9.86342
          writing2     reading1       2.56677         0.1091         10.15901
          writing2     math2          1.94879         0.1627          8.40273
          writing2     reading3       1.75181         0.1856         -7.82777
          writing3     reading1       1.57978         0.2088         -7.97915
          writing1     reading2       1.34894         0.2455          7.77158
          writing2     math3          1.11704         0.2906         -7.23762
```

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 88.4.5 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. This confirms with the initial argument that uncorrelated factors might not reasonable in this case—it fails to explain the covariances among observed variables through the correlations among latent factors. 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. To do this, you need to add the COV statement for specifying the covariances among factors to the original code. The following statements are used to specify the modified factor model with covariances among factors:

```
proc tcalis data=cognitive1 nobs=64;
   factor
      Read_Factor   -> reading1-reading3  = load1-load3,
      Math_Factor   -> math1-math3        = load4-load6,
      Write_Factor  -> writing1-writing3  = load7-load9;
   pvar
      Read_Factor Math_Factor Write_Factor = 3 * 1.,
      reading1-reading3 math1-math3 writing1-writing3 = errvar1-errvar9;
   cov Read_Factor Math_Factor Write_Factor = fcov1-fcov3;
run;
```

The fit summary is shown in Output 88.4.6.

Output 88.4.6 Fit of the Correlated Factor Model

```
                                  Fit Summary

       Modeling Info         N Observations                                64
                             N Variables                                    9
                             N Moments                                     45
                             N Parameters                                  21
                             N Active Constraints                           0
                             Independence Model Chi-Square           272.0467
                             Independence Model Chi-Square DF              36
       Absolute Index        Fit Function                              0.4677
                             Chi-Square                               29.4667
                             Chi-Square DF                                 24
                             Pr > Chi-Square                           0.2031
                             Z-Test of Wilson & Hilferty               0.8320
                             Hoelter Critical N                            79
                             Root Mean Square Residual (RMSR)          5.7038
                             Standardized RMSR (SRMSR)                 0.0607
                             Goodness of Fit Index (GFI)               0.9109
       Parsimony Index       Adjusted GFI (AGFI)                       0.8330
                             Parsimonious GFI                          0.6073
                             RMSEA Estimate                            0.0601
                             RMSEA Lower 90% Confidence Limit               .
                             RMSEA Upper 90% Confidence Limit          0.1244
                             Probability of Close Fit                  0.3814
                             ECVI Estimate                             1.2602
                             ECVI Lower 90% Confidence Limit                .
                             ECVI Upper 90% Confidence Limit           1.5637
                             Akaike Information Criterion            -18.5333
                             Bozdogan CAIC                           -94.3465
                             Schwarz Bayesian Criterion              -70.3465
                             McDonald Centrality                       0.9582
       Incremental Index     Bentler Comparative Fit Index             0.9768
                             Bentler-Bonett NFI                        0.8917
                             Bentler-Bonett Non-normed Index           0.9653
                             Bollen Normed Index Rho1                  0.8375
                             Bollen Non-normed Index Delta2            0.9780
                             James et al. Parsimonious NFI             0.5945
```

The model fit chi-square value is 29.27, which is about 20 less than the model with uncorrelated factors. The p-value is 0.20, indicating a fairly 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 non-normed index. By all counts, the correlated factor model is a much better fit than the uncorrelated factor model.

In Output 88.4.7, the estimation results for factor loadings are shown. All these loadings are statistically significant, indicating non-chance relationships with the factors.

Output 88.4.7 Estimation of the Factor Loading Matrix

```
                 Factor Loading Matrix: Estimate/StdErr/t-value

                         Read_Factor        Math_Factor       Write_Factor

          reading1            6.7657                  0                  0
                              1.0459
                              6.4689
                             [load1]

          reading2            7.8579                  0                  0
                              1.1890
                              6.6090
                             [load2]

          reading3            9.1344                  0                  0
                              1.0712
                              8.5269
                             [load3]

          math1                    0             7.5488                  0
                                                 1.0128
                                                 7.4536
                                                [load4]

          math2                    0             8.4401                  0
                                                 1.0838
                                                 7.7874
                                                [load5]

          math3                    0             6.8194                  0
                                                 1.0910
                                                 6.2506
                                                [load6]

          writing1                 0                  0             7.9677
                                                                    1.1254
                                                                    7.0797
                                                                   [load7]

          writing2                 0                  0             6.8742
                                                                    1.1986
                                                                    5.7350
                                                                   [load8]

          writing3                 0                  0             7.0949
                                                                    1.2057
                                                                    5.8844
                                                                   [load9]
```

In Output 88.4.8, 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 88.4.8 Estimation of the Correlations of Factors

```
                 Factor Covariance Matrix: Estimate/StdErr/t-value

                            Read_Factor        Math_Factor       Write_Factor

         Read_Factor             1.0000             0.3272             0.4810
                                                    0.1311             0.1208
                                                    2.4955             3.9813
                                                   [fcov1]            [fcov2]

         Math_Factor             0.3272             1.0000             0.3992
                                 0.1311                                0.1313
                                 2.4955                                3.0417
                                [fcov1]                               [fcov3]

         Write_Factor            0.4810             0.3992             1.0000
                                 0.1208             0.1313
                                 3.9813             3.0417
                                [fcov2]            [fcov3]
```

In Output 88.4.9, the error variances for variables are shown.

Output 88.4.9 Estimation of the Error Variances

```
                               Error Variances

                                                  Standard
          Variable     Parameter        Estimate       Error         t Value

          reading1     errvar1          37.24939     8.33997         4.46637
          reading2     errvar2          46.49695    10.69869         4.34604
          reading3     errvar3          15.90447     9.26097         1.71737
          math1        errvar4          25.22889     7.72269         3.26685
          math2        errvar5          24.89032     8.98327         2.77074
          math3        errvar6          42.12110     9.20362         4.57658
          writing1     errvar7          27.24965    10.36489         2.62903
          writing2     errvar8          49.28881    11.39812         4.32429
          writing3     errvar9          48.10684    11.48868         4.18733
```

All t values except the one for reading3 are bigger 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 nonsignificant 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 TCALIS also outputs supplementary results that could be useful for interpretations. In Output 88.4.10, the squared multiple correlations and the factor scores regression coefficients are shown.

Output 88.4.10 Supplementary Estimation Results

```
                     Squared Multiple Correlations

                              Error          Total
              Variable       Variance       Variance     R-Square

              reading1       37.24939       83.02400       0.5513
              reading2       46.49695      108.24300       0.5704
              reading3       15.90447       99.34100       0.8399
              math1          25.22889       82.21400       0.6931
              math2          24.89032       96.12500       0.7411
              math3          42.12110       88.62500       0.5247
              writing1       27.24965       90.73400       0.6997
              writing2       49.28881       96.54300       0.4895
              writing3       48.10684       98.44500       0.5113

                  Factor Scores Regression Coefficients

                     Read_Factor        Math_Factor       Write_Factor

       reading1          0.02001          0.0006807            0.00198
       reading2          0.01861          0.0006334            0.00185
       reading3          0.06326            0.00215            0.00628
       math1             0.00112            0.04035            0.00281
       math2             0.00127            0.04572            0.00318
       math3           0.0006068            0.02183            0.00152
       writing1          0.00319            0.00274            0.05128
       writing2          0.00152            0.00131            0.02446
       writing3          0.00161            0.00138            0.02587
```

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.

Example 88.5: Testing Equality of Two Covariance Matrices Using a Multiple-Group Analysis

You can use PROC TCALIS 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. See Example 88.10 for a more sophisticated example of multiple-group analysis.

In this example, a reaction time experiment 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, you will use PROC TCALIS to see the tenability of such an assumption. The covariance matrices for the two groups are stored in the data sets Expert and Novice, as shown in the following:

```
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 TCALIS specification:

```
proc tcalis;
   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;
         matrix _COV_ [1,1] = v_high v_medium v_low,
                      [2,]  = cov21,
                      [3,]  = cov31 cov32;
   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. The NOBS= option is used in both statements to provide the number of observations of these groups. The LABEL= option is used in these statements to provide meaningful group labels.

The MODEL statement defines MODEL 1. In the analysis, this model fits both groups 1 and 2, as indicated by the GROUPS= option of the statement. 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. These three variables are also the row and column variables in the _COV_ matrix defined in the MATRIX statement.

In the first entry of the MATRIX statement, the diagonal elements of the _COV_ matrix are specified. Starting with the element [1,1] (and implicitly followed by [2,2] and [3,3]), variances for the three variables are parameters named v_high, v_medium, and v_low, respectively. In the second entry of the MATRIX statement, parameter specification starts with (and also ends with) the first element of the second row (that is, [2,1]) of the _COV_ matrix. The covariance between the variables high and medium is specified as cov21. In the last entry of the MATRIX statement, parameter specification starts with the first element of the third row (that is, [3,1] and then [3,2]) of the _COV_ matrix. The covariance between the variables high and low is specified as cov31 and the covariance between the variables medium and low is specified as cov32. No initial values are provided in this example. PROC TCALIS generates initial values by some reasonable methods.

There are six parameters in the covariance matrix _COV_, which also has six nonredundant elements in its general form as a symmetric matrix. Hence, model 1 is saturated. If this model were to fit a single group, it would have indicated a perfect model fit with zero chi-square value. Such a model is quite trivial and not of interest. But with two independent groups being fitted by this single model, it becomes a nontrivial model in which you test the equality of the two covariance matrices. Having this main purpose in mind, you use options in the FITINDEX statement to extract the relevant results that could answer your question. First, 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, only the model fit chi-square statistic, degrees of freedom, and the probability value of the test are requested. 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 TCALIS as a built-in chi-square correction option.

In addition, because you are not interested in all displayed output, you use the ODS SELECT statement to display only those output (or ODS tables) of interest. In this example, only the modeling information, the variables involved, the initial covariance matrix specification, and the fit summary table are printed. All output in PROC TCALIS are named as an ODS table. To locate a particular output in PROC TCALIS, you must know the corresponding ODS table names. See the section "ODS Table Names" on page 6931 for a listing of ODS table names used by PROC TCALIS.

Output 88.5.1 displays some information regarding the basic model setup.

Output 88.5.1 Modeling Information and Initial Specification

```
                         Modeling Information

        Group     Label      Data Set          N Obs     Model       Type

            1     Expert     WORK.EXPERT          20     Model 1     MSTRUCT
            2     Novice     WORK.NOVICE          18     Model 1     MSTRUCT
```

Output 88.5.1 *continued*

```
                    Model 1. Variables in the Model

                        high   medium   low

                        Number of Variables = 3

                 Model 1. Initial MSTRUCT _COV_ Matrix

                              high           medium              low

        high                     .                .                .
                          [v_high]          [cov21]          [cov31]

        medium                   .                .                .
                           [cov21]       [v_medium]          [cov32]

        low                      .                .                .
                           [cov31]          [cov32]          [v_low]
```

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. The parameters for the entries in the covariance matrix are shown. The names of parameters are displayed in parentheses. No initial estimates are given as input, as indicated by the missing value '.'.

The fit summary table is shown in Output 88.5.2, which has been much simplified for the current example due to the uses of some options in the FITINDEX statement.

Output 88.5.2 Model Fit

```
                          Fit Summary

                 Chi-Square               2.4924
                 Chi-Square DF                 6
                 Pr > Chi-Square          0.8693
```

As shown in Table 88.5.2, 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 fitted model is supported, which means that the equality of the covariance matrices of the groups is tenable.

Example 88.6: Illustrating Various General Modeling Languages

In PROC TCALIS, 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 6706 for guidance. In this example, the data and the model in Example 88.1 are used to illustrate how a particular model can be specified by various general modeling languages.

RAM Model Specification

In Example 88.1, the PATH modeling language was used. The PATH modeling language is closely related to the RAM modeling language. The following RAM model specification shows similar syntax to that of the PATH model specification.

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

In the RAM model, a list of parameter locations are specified in the RAM statement. In each entry of the list, the parameter type, the variables involved, and the parameter are specified in order. These entries are separated by commas.

The parameter type is listed first in each list entry. In this example, three parameter types are specified: PATH, PVAR, and PCOV. These types correspond to the PATH, PVAR, and PCOV statements in the PATH model specification. There are other parameter types you can use for the RAM model specification, but they are not shown in this example. See the RAM statement on page 6797 for more details.

The variables involved are specified after the parameter type in each list entry. Depending on the parameter type, the number of variables required is either one or two. For entries with the PATH parameter type, two variables must be specified—one for the outcome and one for the predictor. In this example, the two variables involved in the PATH entries are separated by arrows "<–" that indicate the prediction directions. When the arrows are not specified, the first variable is assumed to be the outcome variable, and the second one is the predictor variable. For the PVAR entries, which stand for partial variance, only one variable is required after the parameter type specification. For the PCOV entries, which stand for partial covariance, two variables are required. The order of these two variables in the PCOV entries is not important, however.

The last specification in each list entry is the parameter. The parameter can either be a fixed value (a numerical value alone), a free parameter without an initial value (a name alone), or a free parameter with an initial value (a name followed by a parenthesized numerical value). In this example, you specify five fixed values for the path coefficients (or effects) in the first five entries. In the next four paths and all other PVAR and PCOV entries, you specify free parameters without initial values.

When comparing with the PATH model specification, the RAM specification is quite similar. Certainly, the model fit would be the same because both specifications are mathematically equivalent. Unlike the PATH model that uses different tables for displaying different types of estimates, all estimates of the RAM model are displayed in a single table as shown in Output 88.6.1.

Output 88.6.1 RAM Model Estimates

```
                                    RAM Pattern and Estimates

                                                                          Standard
Type           Var1                Var2          Parameter      Estimate        Error       t Value

Path           Anomie67       <-   Alien67                       1.00000
               Powerless67    <-   Alien67                       0.83300
               Anomie71       <-   Alien71                       1.00000
               Powerless71    <-   Alien71                       0.83300
               Education      <-   SES                           1.00000
               SEI            <-   SES           lambda          5.36883      0.43371      12.37880
               Alien67        <-   SES           gamma1         -0.62994      0.05634     -11.18092
               Alien71        <-   SES           gamma2         -0.24086      0.05489      -4.38836
               Alien71        <-   Alien67       beta            0.59312      0.04678      12.67884
Partial Var    Anomie67                          theta1          3.60796      0.20092      17.95717
               Powerless67                       theta2          3.59488      0.16448      21.85563
               Anomie71                          theta1          3.60796      0.20092      17.95717
               Powerless71                       theta2          3.59488      0.16448      21.85563
               Education                         theta3          2.99366      0.49861       6.00398
               SEI                               theta4        259.57639     18.31151      14.17559
               Alien67                           psi1            5.67046      0.42301      13.40500
               Alien71                           psi2            4.51479      0.33532      13.46394
               SES                               phi             6.61634      0.63914      10.35190
Partial Cov    Anomie67            Anomie71      theta5          0.90580      0.12167       7.44472
               Powerless67         Powerless71   theta5          0.90580      0.12167       7.44472
```

These RAM model estimates match the set of estimates using the PATH model specification.

LINEQS Model Specification

Another way to specify the model in Example 88.1 is to use the LINEQS modeling language, which is shown in the following:

```
proc tcalis 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;
   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 model, all paths in the PATH model are now specified in equations. In each equation, you list an outcome variable on the left-hand-side of the equation and all its predictors (including the error or disturbance variable) on the right-hand-side of the equation. Because each outcome variable 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 model do not match necessarily. In this example, there are eight equations in the LINEQS statement, but there would be nine paths specified in the corresponding PATH model.

In addition, in the LINEQS model, you need to follow a convention of naming latent variables. Names for latent variables that are neither errors nor disturbances must start with either 'F' or 'f.' Names for errors must start with either 'E' or 'e,' while names for disturbances must start with either 'D' or 'd.' Recall that in the PATH 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 model. Furthermore, there are no explicit error terms that need to be specified in the PATH model, let alone specific prefixes for the error terms.

The PVAR statement in the PATH model is now replaced with the STD statement in the LINEQS model, and the PCOV statement with the COV statement. Recall that 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. But for the LINEQS model, errors or disturbances are named explicitly as **exogenous** variables so that the partial variance or covariance concepts are no longer necessary. In this example, the variances of the errors (“E”-variables) and disturbances (“D”-variables) specified in the STD 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 88.6.2. Again, they are essentially the same estimates obtained from the PATH model specified in Example 88.1.

Output 88.6.2 LINEQS Model Estimates

```
                              Linear Equations

   Anomie67     =    1.0000 f_Alien67 +  1.0000 E1
   Powerless67 =    0.8330 f_Alien67 +  1.0000 E2
   Anomie71     =    1.0000 f_Alien71 +  1.0000 E3
   Powerless71 =    0.8330 f_Alien71 +  1.0000 E4
   Education    =    1.0000 f_SES     +  1.0000 E5
   SEI          =    5.3688*f_SES     +  1.0000 E6
   Std Err           0.4337 lambda
   t Value          12.3788
   f_Alien67    =   -0.6299*f_SES     +  1.0000 D1
   Std Err           0.0563 gamma1
   t Value         -11.1809
   f_Alien71    =   -0.2409*f_SES     +  0.5931*f_Alien67 +  1.0000 D2
   Std Err           0.0549 gamma2       0.0468 beta
   t Value          -4.3884             12.6788

                 Estimates for Variances of Exogenous Variables

 Variable                                                  Standard
 Type            Variable    Parameter      Estimate          Error       t Value

 Error           E1          theta1          3.60796        0.20092      17.95717
                 E2          theta2          3.59488        0.16448      21.85563
                 E3          theta1          3.60796        0.20092      17.95717
                 E4          theta2          3.59488        0.16448      21.85563
                 E5          theta3          2.99366        0.49861       6.00398
                 E6          theta4        259.57639       18.31151      14.17559
 Disturbance     D1          psi1            5.67046        0.42301      13.40500
                 D2          psi2            4.51479        0.33532      13.46394
 Latent          f_SES       phi             6.61634        0.63914      10.35190

                     Covariances Among Exogenous Variables

                                                    Standard
      Var1     Var2     Parameter      Estimate        Error        t Value

      E1       E3       theta5          0.90580      0.12167        7.44472
      E2       E4       theta5          0.90580      0.12167        7.44472
```

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 6839 for details.

Four types of variables must be recognized before you can specify a LISMOD model. The η-variables (eta-variables) are latent factors that are endogenous, or predicted by other latent factors. The ξ-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 η-variables, and the x-variables are manifest variables that are indicators of the ξ-variables. In this example, Alien67 and Alien71 are the η-variables, and SES is the ξ-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_, _LAMBDAX_, _GAMMA_, and _BETA_ are matrices for the path coefficients or effects. The _THETAY, _THETAX_, _PSI_, and _PHI_ are matrices for the variances and covariances.

The following is the LISMOD specification for the model in Example 88.1:

```
proc tcalis 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 are not arbitrary. They are the orders applied to the row and column variables in the model matrices, of which the parameter locations are specified.

The estimated model is divided into three conceptual parts. The first part is the measurement model that relates the η-variables with the y-variables, as shown in Output 88.6.3:

Output 88.6.3 LISMOD Model Measurement Model for the Eta-Variables

```
                                _LAMBDAY_ Matrix: Estimate/StdErr/t-value

                                                  Alien67        Alien71

                                Anomie67           1.0000              0



                                Powerless67        0.8330              0



                                Anomie71                0         1.0000



                                Powerless71             0         0.8330




                                _THETAY_ Matrix: Estimate/StdErr/t-value

                                 Anomie67    Powerless67        Anomie71    Powerless71

              Anomie67             3.6080              0          0.9058              0
                                   0.2009                         0.1217
                                  17.9572                         7.4447
                                 [theta1]                       [theta5]

              Powerless67               0         3.5949               0         0.9058
                                                  0.1645                         0.1217
                                                 21.8556                         7.4447
                                                [theta2]                       [theta5]

              Anomie71             0.9058              0          3.6080              0
                                   0.1217                         0.2009
                                   7.4447                        17.9572
                                 [theta5]                       [theta1]

              Powerless71               0         0.9058               0         3.5949
                                                  0.1217                         0.1645
                                                  7.4447                        21.8556
                                                [theta5]                       [theta2]
```

The _LAMBDAY_ matrix contains the coefficients or effects of the η-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 ξ-variable with the x-variables, as shown in Output 88.6.4:

Output 88.6.4 LISMOD Model Measurement Model for the Xi-Variables

```
                      _LAMBDAX_ Matrix: Estimate/StdErr/t-value

                                              SES

                              Education         1.0000




                              SEI               5.3688
                                                0.4337
                                               12.3788
                                              [lambda]

                      _THETAX_ Matrix: Estimate/StdErr/t-value

                                     Education             SEI

                      Education         2.9937               0
                                        0.4986
                                        6.0040
                                       [theta3]

                      SEI                    0        259.5764
                                                       18.3115
                                                       14.1756
                                                      [theta4]
```

The _LAMBDAX_ 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 _THETAX_ 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 88.6.5:

Output 88.6.5 LISMOD Structural Model for the Latent Variables

```
                    _BETA_ Matrix: Estimate/StdErr/t-value

                                 Alien67        Alien71

                    Alien67            0              0




                    Alien71       0.5931              0
                                  0.0468
                                 12.6788
                                  [beta]

                    _GAMMA_ Matrix: Estimate/StdErr/t-value

                                            SES

                            Alien67     -0.6299
                                         0.0563
                                       -11.1809
                                       [gamma1]

                            Alien71     -0.2409
                                         0.0549
                                        -4.3884
                                       [gamma2]

                    _PSI_ Matrix: Estimate/StdErr/t-value

                                 Alien67        Alien71

                    Alien67       5.6705              0
                                  0.4230
                                 13.4050
                                  [psi1]

                    Alien71            0         4.5148
                                                 0.3353
                                                13.4639
                                                 [psi2]

                     _PHI_ Matrix: Estimate/StdErr/t-value

                                          SES

                              SES      6.6163
                                       0.6391
                                      10.3519
                                        [phi]
```

The _BETA_ matrix contains effects of η-variables on themselves. In our 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 ξ-variable, which is SES in this example, on the η-variables Alien67 on Alien71. These effects are represented by 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 ξ-variables. In this example, there is only one ξ-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 88.7: 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
 9.2  13.5  17.8  19.2  21.1
18.3  23.5  27.9  30.2  34.6
11.2  15.6  20.8  22.7  30.4
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 ($t = 1, 2, \ldots, 5$), α represents the intercept, β 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 α_i and β_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 α and β 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 (α, β) 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 (α, β).

To use PROC TCALIS 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_α and f_β in the following structural equation:

$$y_t = f_\alpha + T_t f_\beta + e_t$$

where f_α and f_β 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.

```
proc tcalis method=ml data=growth nostand noparmname;
   lineqs
      y1 = f_alpha              + e1,
      y2 = f_alpha  +  1 f_beta + e2,
      y3 = f_alpha  +  2 f_beta + e3,
      y4 = f_alpha  +  3 f_beta + e4,
      y5 = f_alpha  +  4 f_beta + e5;
   std
      f_alpha   = parm__,
      f_beta    = parm__,
      e1-e5     = 5 * evar;
   cov
      f_alpha f_beta = parm__;
   mean
      f_alpha   = parm__,
      f_beta    = parm__;
   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 not arbitrary but is required as a convention in the LINEQS modeling language. See the sections “Naming Variables in the LINEQS Model” on page 6832 and “Naming Variables and Parameters” on page 6872 for details.

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 STD statement, while their covariance is specified in the COV statement. The means of f_alpha and f_beta are specified in the MEAN statement. Notice that the parameter name parm__ is used for all these parameters. Parameter names ending with two underscores (‘__’) are used as generic names for generating new parameter names. Each time such a generic name is parsed, PROC TCALIS replaces the underscores with an unique integer. Therefore, in the various locations of parm__ in the LINEQS model, the actual parameter names registered with PROC TCALIS are parm1, parm2, ..., parm5. For example, the variance parameters of f_alpha and f_beta are named parm1 and parm2, respectively, and the mean parameter of f_beta is named parm5.

The error variances for e1–e5 are also specified in the STD statement. Using the shorthand notation 5 *, the parameter name evar is repeated five times for the five error variances. This implicitly constrains the error variances for e1–e5 to be equal.

Special printing options are used in this example. In the PROC TCALIS 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 TCALIS statement.

This option suppresses the printing of parameter names in the output for estimation. This makes the output look more precise when you do not need direct 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 6752 for details.

The fit summary table is shown in Output 88.7.1.

Output 88.7.1 Random Intercepts and Effects with Constrained Error Variances: Model Fit

```
                              Fit Summary

                    Chi-Square                31.4310
                    Chi-Square DF                  14
                    Pr > Chi-Square            0.0048
```

In Output 88.7.1, 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 88.7.2 Estimation of Random Intercepts and Effects with Constrained Error Variances

```
                 Estimates for Variances of Exogenous Variables

        Variable                                       Standard
        Type            Variable       Estimate           Error        t Value

        Latent          f_alpha        13.89140         5.81540        2.38873
                        f_beta          0.80742         0.42198        1.91342
        Error           e1              3.32185         0.70031        4.74342
                        e2              3.32185         0.70031        4.74342
                        e3              3.32185         0.70031        4.74342
                        e4              3.32185         0.70031        4.74342
                        e5              3.32185         0.70031        4.74342

                      Covariances Among Exogenous Variables

                                                    Standard
           Var1          Var2          Estimate        Error         t Value

           f_alpha       f_beta        -0.35281      1.13815        -0.30998

                                Mean Parameters

          Variable                                       Standard
          Type            Variable       Estimate           Error        t Value

          Latent          f_alpha        14.15875         1.02906       13.75890
                          f_beta          4.04813         0.27563       14.68665
```

In Output 88.7.2, the estimated variance of the random intercept α, 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 β, 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.

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. This can be done by using the following specification:

```
proc tcalis method=ml data=growth nostand noparmname;
   lineqs
      y1 = f_alpha              + e1,
      y2 = f_alpha   +  1 f_beta + e2,
      y3 = f_alpha   +  2 f_beta + e3,
      y4 = f_alpha   +  3 f_beta + e4,
      y5 = f_alpha   +  4 f_beta + e5;
   std
      f_alpha    = parm__,
      f_beta     = parm__,
      e1-e5      = 5 * parm__;
   cov
      f_alpha f_beta = parm__;
   mean
      f_alpha    = parm__,
      f_beta     = parm__;
   fitindex on(only)=[chisq df probchi];
run;
```

In this new specification, there is only one change in the STD statement from the previous specification. That is, you now use five different parameters for the five error variances, as highlighted in the following:

```
e1-e5   = 5 * parm__;
```

This specification is equivalent to putting parm__ five times for the five error variance parameters, which is shown as follows:

```
e1-e5   = parm__ parm__ parm__ parm__ parm__;
```

The model fit summary is shown in Output 88.7.3.

Output 88.7.3 Random Intercepts and Effects with Unconstrained Error Variances: Model Fit

```
                              Fit Summary

                  Chi-Square              11.6250
                  Chi-Square DF                10
                  Pr > Chi-Square          0.3109
```

As you can see from the fit summary 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 α=0.01, indicating a significant improvement of model fit by using the unconstrained model.

Estimation results are shown in Output 88.7.4.

Output 88.7.4 Estimation of Random Intercepts and Effects with Unconstrained Error Variances

```
              Estimates for Variances of Exogenous Variables

      Variable                                       Standard
      Type           Variable       Estimate            Error         t Value

      Latent         f_alpha        14.70071          5.66943         2.59298
                     f_beta          0.45059          0.29867         1.50867
      Error          e1              2.81712          1.35332         2.08164
                     e2              0.32213          0.46118         0.69848
                     e3              1.94429          0.86824         2.23935
                     e4              1.88569          1.21306         1.55448
                     e5             14.65193          5.99354         2.44462

                   Covariances Among Exogenous Variables

                                                  Standard
         Var1         Var2         Estimate          Error         t Value

         f_alpha      f_beta        0.35291        0.90366         0.39054

                              Mean Parameters

        Variable                                       Standard
        Type           Variable       Estimate            Error        t Value

        Latent         f_alpha        14.03046          1.01534       13.81851
                       f_beta          3.96793          0.22612       17.54781
```

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 more data should be collected and the model should be fit again to check for the significance of the covariance between the random intercept and slope. If the covariance estimate is still not significant, future models might have this covariance fixed at 0 in analysis.

Example 88.8: Higher-Order and Hierarchical Factor Models

In this example, confirmatory higher-order and hierarchical factor models are fitted by PROC TCALIS.

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 factor model you have nine manifest variables V1–V9 with three first-order factors F1–F3. The first-order factor pattern of the model might appear like the following:

```
        F1   F2   F3
   V1    x
   V2    x
   V3    x
   V4         x
   V5         x
   V6         x
   V7              x
   V8              x
   V9              x
```

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 F4 is hypothesized with the following second-order factor pattern:

```
        F4
   F1    x
   F2    x
   F3    x
```

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:

```
         First Layer  |  Second Layer
        F1   F2   F3  |      F4
   V1    x            |       x
   V2    x            |       x
   V3    x            |       x
   V4         x       |       x
   V5         x       |       x
   V6         x       |       x
   V7              x  |       x
   V8              x  |       x
   V9              x  |       x
```

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 a 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 TCALIS 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 TCALIS 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.

```
proc tcalis 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;
   std
      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 predicted 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 STD 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, options are used to control the output. In the PROC TCALIS statement, 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 options in PROC TCALIS 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 88.8.1, parameters and their initial values, gradients, and bounds are shown.

Output 88.8.1 Parameters in the Model

```
                              Optimization Start
                              Parameter Estimates

    N     Parameter      Estimate       Gradient     Lower Bound     Upper Bound

    1     X11             1.00000        0.13476               .               .
    2     X21             1.01408        0.17327               .               .
    3     X31             0.95518        0.12174               .               .
    4     X42             1.00000        0.22548               .               .
    5     X52             0.96603        0.21304               .               .
    6     X62             0.88305        0.19782               .               .
    7     X73             1.00000        0.21041               .               .
    8     X83             1.03403        0.39324               .               .
    9     X93             0.91752        0.19880               .               .
   10     L1g             0.75060       -0.57492               .               .
   11     L2g             0.64268       -0.50975               .               .
   12     L3g             0.60919       -0.56538               .               .
   13     U1              0.18879        0.14837               0               .
   14     U2              0.16579        0.08989               0               .
   15     U3              0.25988       -0.03231               0               .
   16     U4              0.33068        0.20120               0               .
   17     U5              0.37538        0.09124               0               .
   18     U6              0.47808       -0.03595               0               .
   19     U7              0.44813        0.20918               0               .
   20     U8              0.40994       -0.12469               0               .
   21     U9              0.53541        0.05959               0               .

                  Value of Objective Function = 0.5693888709

                    The Number of Dependent Parameters is 3

                           N     Parameter      Estimate

                          22     W1              0.43660
                          23     W2              0.58697
                          24     W3              0.62889
```

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 initial independent parameters.

Output 88.8.2 shows information about optimization—iteration history and the convergence status.

Output 88.8.2 Optimization

```
              Parameter Estimates                                   21
              Functions (Observations)                              45
              Lower Bounds                                           9
              Upper Bounds                                           0
```

Output 88.8.2 *continued*

```
                              Optimization Start

 Active Constraints                         0    Objective Function     0.5693888709
 Max Abs Gradient Element        0.5749163348    Radius                 1.8533033852


                                                                               Actual
                                                           Max Abs               Over
         Rest     Func       Act     Objective  Obj Fun Gradient                 Pred
 Iter    arts    Calls       Con      Function   Change  Element    Lambda     Change

    1       0        5         0       0.38684   0.1825   0.5158     3.214      1.174
    2       0        9         0       0.18706   0.1998   0.1003         0      1.181
    3       0       11         0       0.18039  0.00667   0.0273         0      0.987
    4       0       13         0       0.18020 0.000192  0.00581         0      0.881
    5       0       15         0       0.18017 0.000023  0.00295         0      0.967
    6       0       17         0       0.18017   3.08E-6 0.000686        0      1.083
    7       0       19         0       0.18017 4.606E-7 0.000379         0      1.195
    8       0       21         0       0.18017 7.365E-8 0.000096         0      1.283
    9       0       23         0       0.18017 1.228E-8 0.000054         0      1.342
   10       0       25         0       0.18017 2.098E-9 0.000018         0      1.377
   11       0       27         0       0.18017 3.63E-10 8.561E-6         0      1.397

                             Optimization Results

Iterations                            11   Function Calls                       30
Jacobian Calls                        13   Active Constraints                    0
Objective Function          0.1801712146   Max Abs Gradient Element   8.5605681E-6
Lambda                                 0   Actual Over Pred Change    1.3969225014
Radius                      0.0000572561

Convergence criterion (GCONV=1E-8) satisfied.
```

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.

In the fit summary table shown in Output 88.8.3, the chi-square model fit value is 38.196, with df=24, and p=0.033. This indicates a satisfactory model fit.

Output 88.8.3 Fit Summary

```
                                 Fit Summary

                          Chi-Square             38.1963
                          Chi-Square DF               24
                          Pr > Chi-Square         0.0331
```

The fitted equations with final estimates are shown in Output 88.8.4. 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.

Output 88.8.4 Estimation Results

```
                              Linear Equations

              V1       =     0.9047*Factor1 +   1.0000 E1
                                    X11
              V2       =     0.9138*Factor1 +   1.0000 E2
                                    X21
              V3       =     0.8561*Factor1 +   1.0000 E3
                                    X31
              V4       =     0.8358*Factor2 +   1.0000 E4
                                    X42
              V5       =     0.7972*Factor2 +   1.0000 E5
                                    X52
              V6       =     0.7026*Factor2 +   1.0000 E6
                                    X62
              V7       =     0.7808*Factor3 +   1.0000 E7
                                    X73
              V8       =     0.7202*Factor3 +   1.0000 E8
                                    X83
              V9       =     0.7035*Factor3 +   1.0000 E9
                                    X93
              Factor1 =     0.8221*FactorG +   1.0000 E10
                                    L1g
              Factor2 =     0.7818*FactorG +   1.0000 E11
                                    L2g
              Factor3 =     0.8150*FactorG +   1.0000 E12
                                    L3g


              Estimates for Variances of Exogenous Variables

            Variable
            Type           Variable     Parameter      Estimate

            Latent         FactorG                      1.00000
            Error          E1           U1              0.18150
                           E2           U2              0.16493
                           E3           U3              0.26713
                           E4           U4              0.30150
                           E5           U5              0.36450
                           E6           U6              0.50642
                           E7           U7              0.39033
                           E8           U8              0.48137
                           E9           U9              0.50510
            Disturbance    E10          W1              0.32420
                           E11          W2              0.38879
                           E12          W3              0.33576
```

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 have:

$$(0.8221)^2 + 0.32420 = 1.0000$$

So, this model constraint is verified.

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 tcalis 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;
   std
      Factor1-Factor3 = 3 * 1.,
      FactorG         = 1. ,
      E1-E9           = U1-U9;
   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.

All factors in the bifactor model are uncorrelated. The factor variances are all fixed at 1 in the STD 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. Like the previous PROC TCALIS run, options are specified in the PROC TCALIS 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 88.8.5, which shows the optimization information.

Output 88.8.5 Optimization

```
              Parameter Estimates                                 27
              Functions (Observations)                            45
              Lower Bounds                                         9
              Upper Bounds                                         0


                                  Optimization Start

  Active Constraints                      0    Objective Function    0.8380304146
  Max Abs Gradient Element     2.4076251809    Radius                20.596787596


                                                                        Actual
                                                      Max Abs             Over
           Rest     Func      Act    Objective  Obj Fun  Gradient              Pred
  Iter     arts    Calls      Con     Function   Change   Element   Lambda   Change

     1        0        5        0      0.70566   0.1324    0.4851  0.00140    0.148
     2        0        7        0      0.30090   0.4048    0.3269        0    1.292
     3        0        9        0      0.17403   0.1269    0.2947        0    0.985
     4        0       11        0      0.11759   0.0564    0.0677        0    1.190
     5        0       13        0      0.11455  0.00304    0.0267        0    1.043
     6        0       15        0      0.11426 0.000285   0.00242        0    1.153
     7        0       17        0      0.11423 0.000027   0.00168        0    1.394
     8        0       19        0      0.11423 5.552E-6  0.000478        0    1.413
     9        0       21        0      0.11423 1.154E-6  0.000335        0    1.420
    10        0       23        0      0.11423 2.405E-7  0.000105        0    1.427
    11        0       25        0      0.11423 5.016E-8  0.000068        0    1.432
    12        0       27        0      0.11423 1.047E-8  0.000023        0    1.436
    13        0       29        0      0.11423 2.184E-9  0.000014        0    1.439
    14        0       31        0      0.11423 4.56E-10  4.909E-6        0    1.442


                                 Optimization Results

 Iterations                            14  Function Calls                       34
 Jacobian Calls                        16  Active Constraints                    0
 Objective Function          0.1142278162  Max Abs Gradient Element   4.9090342E-6
 Lambda                                 0  Actual Over Pred Change    1.4423534599
 Radius                      0.0002294218

 Convergence criterion (GCONV=1E-8) satisfied.
```

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 88.8.6

Output 88.8.6 Fit Summary

```
                                 Fit Summary

                       Chi-Square              24.2163
                       Chi-Square DF                18
                       Pr > Chi-Square          0.1481
```

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 88.8.7. Estimates are left uninterpreted because they are not the main interest of this example.

Output 88.8.7 Estimation Results

```
                              Linear Equations

            V1  =   -0.4879*Factor1 +  0.7679*FactorG +  1.0000 E1
                            X11                X1g
            V2  =   -0.4523*Factor1 +  0.7909*FactorG +  1.0000 E2
                            X21                X2g
            V3  =   -0.4045*Factor1 +  0.7536*FactorG +  1.0000 E3
                            X31                X3g
            V4  =    0.6140*Factor2 +  0.6084*FactorG +  1.0000 E4
                            X42                X4g
            V5  =    0.5058*Factor2 +  0.5973*FactorG +  1.0000 E5
                            X52                X5g
            V6  =    0.3943*Factor2 +  0.5718*FactorG +  1.0000 E6
                            X62                X6g
            V7  =   -0.7273*Factor3 +  0.5669*FactorG +  1.0000 E7
                            X73                X7g
            V8  =   -0.2468*Factor3 +  0.6623*FactorG +  1.0000 E8
                            X83                X8g
            V9  =   -0.4091*Factor3 +  0.5300*FactorG +  1.0000 E9
                            X93                X9g
```

Output 88.8.7 *continued*

```
                 Estimates for Variances of Exogenous Variables

               Variable
               Type           Variable    Parameter      Estimate

               Latent         Factor1                     1.00000
                              Factor2                     1.00000
                              Factor3                     1.00000
                              FactorG                     1.00000
               Error          E1          U1              0.17236
                              E2          U2              0.16984
                              E3          U3              0.26848
                              E4          U4              0.25281
                              E5          U5              0.38735
                              E6          U6              0.51757
                              E7          U7              0.14966
                              E8          U8              0.50039
                              E9          U9              0.55175
```

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:

```
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 88.8.8 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 α-level is set at 0.05, the bifactor model indicates a significantly better fit. But if α-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 TCALIS.

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.

```
proc tcalis 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;
   std
      Factor1-Factor3 = 3 * 1.,
      FactorG         = 1. ,
      E1-E9           = U1-U9;
   bounds
      0. <= U1-U9;
   fitindex ON(ONLY)=[chisq df probchi];
   parameters p1 (.5) p2 (.5) p3 (.5);
   /* Proportionality constraints */
   X1g = p1 * X11;
   X2g = p1 * X21;
   X3g = p1 * X31;
   X4g = p2 * X42;
   X5g = p2 * X52;
   X6g = p2 * X62;
   X7g = p3 * X73;
   X8g = p3 * X83;
   X9g = p3 * X93;
run;
```

In this constrained model, a PARAMETERS statement and nine SAS programming statements are added 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:

$$\frac{\mathrm{X1g}}{\mathrm{X11}} = \frac{\mathrm{X2g}}{\mathrm{X21}} = \frac{\mathrm{X3g}}{\mathrm{X31}}$$

$$\frac{\mathrm{X4g}}{\mathrm{X42}} = \frac{\mathrm{X5g}}{\mathrm{X52}} = \frac{\mathrm{X6g}}{\mathrm{X62}}$$

$$\frac{\mathrm{X7g}}{\mathrm{X73}} = \frac{\mathrm{X8g}}{\mathrm{X83}} = \frac{\mathrm{X9g}}{\mathrm{X93}}$$

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 88.8.10. But first, it is also useful to consider the optimization information of the constrained bifactor model, which is shown in Output 88.8.9.

Output 88.8.9 Optimization

```
            Parameter Estimates                                 21
            Functions (Observations)                            45
            Lower Bounds                                         9
            Upper Bounds                                         0


                              Optimization Start

Active Constraints                        0    Objective Function    12.813623548
Max Abs Gradient Element       0.2163121033    Radius                           1
```

Output 88.8.9 *continued*

```
                                                                      Actual
                                                        Max Abs         Over
        Rest     Func      Act    Objective  Obj Fun  Gradient          Pred
 Iter   arts    Calls      Con     Function   Change   Element  Lambda  Change

    1*     0        5        0     11.77997   1.0337    0.1754   6.390   1.107
    2*     0        7        0     11.40277   0.3772    0.1812   0.827   1.032
    3*     0        9        0     10.63174   0.7710    0.1945   0.362   1.096
    4*     0       11        0      8.97007   1.6617    0.2262   0.152   1.275
    5*     0       13        0      5.18770   3.7824    0.4950  0.0283   1.862
    6*     0       15        0      3.90387   1.2838    0.6547   0.781   1.289
    7*     0       17        0      2.17113   1.7327    0.8946   0.781   1.228
    8*     0       19        0      0.43097   1.7402    0.3567   0.195   1.532
    9      0       21        0      0.19151   0.2395    0.0822       0   1.252
   10      0       23        0      0.18095   0.0106    0.0143       0   1.135
   11      0       25        0      0.18024 0.000713   0.00632       0   1.190
   12      0       27        0      0.18018 0.000059   0.00260       0   1.354
   13      0       29        0      0.18017 8.179E-6  0.000970       0   1.385
   14      0       31        0      0.18017 1.232E-6  0.000387       0   1.390
   15      0       33        0      0.18017 1.882E-7  0.000149       0   1.391
   16      0       35        0      0.18017 2.886E-8  0.000059       0   1.392
   17      0       37        0      0.18017 4.431E-9  0.000023       0   1.392
   18      0       39        0      0.18017  6.8E-10  9.004E-6       0   1.392


                          Optimization Results

Iterations                            18   Function Calls                    42
Jacobian Calls                        20   Active Constraints                 0
Objective Function          0.1801712146   Max Abs Gradient Element  9.004196E-6
Lambda                                 0   Actual Over Pred Change  1.3918856965
Radius                      0.0001807581


Convergence criterion (GCONV=1E-8) satisfied.
```

As shown Output 88.8.9, 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 this 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 88.8.10.

Output 88.8.10 Model Fit

```
                              Fit Summary

                   Chi-Square                 38.1963
                   Chi-Square DF                   24
                   Pr > Chi-Square             0.0331
```

In Output 88.8.10, 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 88.8.11, estimation results of the constrained bifactor model are shown.

Output 88.8.11 Estimation Results

```
                          Linear Equations

        V1 =    -0.5151*Factor1 +  0.7437*FactorG +  1.0000 E1
                        X11                X1g
        V2 =    -0.5203*Factor1 +  0.7512*FactorG +  1.0000 E2
                        X21                X2g
        V3 =    -0.4874*Factor1 +  0.7038*FactorG +  1.0000 E3
                        X31                X3g
        V4 =     0.5211*Factor2 +  0.6534*FactorG +  1.0000 E4
                        X42                X4g
        V5 =     0.4971*Factor2 +  0.6232*FactorG +  1.0000 E5
                        X52                X5g
        V6 =     0.4381*Factor2 +  0.5493*FactorG +  1.0000 E6
                        X62                X6g
        V7 =     0.4524*Factor3 +  0.6364*FactorG +  1.0000 E7
                        X73                X7g
        V8 =     0.4173*Factor3 +  0.5869*FactorG +  1.0000 E8
                        X83                X8g
        V9 =     0.4076*Factor3 +  0.5734*FactorG +  1.0000 E9
                        X93                X9g


               Estimates for Variances of Exogenous Variables

            Variable
            Type           Variable    Parameter      Estimate

            Latent         Factor1                     1.00000
                           Factor2                     1.00000
                           Factor3                     1.00000
                           FactorG                     1.00000
            Error          E1          U1              0.18150
                           E2          U2              0.16493
                           E3          U3              0.26713
                           E4          U4              0.30150
                           E5          U5              0.36450
                           E6          U6              0.50642
                           E7          U7              0.39033
                           E8          U8              0.48138
                           E9          U9              0.50510
```

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 88.8.4 for the second-order factor model match those of the constrained bifactor model exactly in Output 88.8.11. 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 88.8.4 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:

$$\begin{aligned} L_1 &= P_1 U_1 \\ L_2 &= P_1 P_2 \end{aligned}$$

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 88.8.11 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 88.9: Linear Relations among Factor Loadings

In this example, a confirmatory factor-analysis model with linear constraints on loadings is specified by the FACTOR modeling language of PROC TCALIS. SAS programming statements are used to set the constraints. In the context of the current example, the differences between fitting covariance structures and correlation structures will also be discussed.

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:

$$\mathbf{B} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \\ b_{51} & b_{52} \\ b_{61} & b_{62} \end{pmatrix}$$

and

$$b_{j2} = \alpha + \beta b_{j1}, \quad j = 1, \ldots, 6$$

The correlation structures are represented by:

$$\mathbf{P} = \mathbf{B}\mathbf{B}' + \Psi$$

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 α and β 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.

```
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 tcalis 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);
   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.' Initial estimates are given in the parentheses. For example, in the first entry of factor-variable relations, you specify loadings of var1–var6 on factor1. All these loadings are free parameters and are given the same initial value 6. Loadings of var1–var6 on factor2 are also defined as parameters in the same statement. However, they are dependent parameters because they are functionally dependent on other independent parameters, as shown in the SAS programming statements afterwards. Initial values for these dependent parameters are thus unnecessary.

In the PVAR statement, the factor variances are fixed at ones, while the error variances for the variables are free parameters named psi1–psi6. Again, initial estimates are provided in the parentheses that follow after the parameter names.

An additional parameter alpha is specified in the PARAMETERS statement with an initial value of 1. Then, six SAS programming statements are used 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 unique 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 and covariances among factors are fixed zeros by default.

In Output 88.9.1, a concise fit summary table is shown. The chi-square test statistic of model fit is 10.337 with df=8 (p=0.242). This indicates a good model fit.

Output 88.9.1 Fit of the Correlation Structures

```
                          Fit Summary

                  Chi-Square                10.3374
                  Chi-Square DF                   8
                  Pr > Chi-Square            0.2421
```

The estimated factor loading matrix is presented in Output 88.9.2, and the estimated error variances and the estimate for alpha are presented in Output 88.9.3.

Output 88.9.2 Loading Estimates

```
                     Factor Loading Matrix

                           factor1        factor2

                var1        0.3609         0.6174
                            [b11]          [b12]

                var2        0.3212         0.6571
                            [b21]          [b22]

                var3        0.4859         0.4923
                            [b31]          [b32]

                var4        0.5745         0.4038
                            [b41]          [b42]

                var5        0.7985         0.1797
                            [b51]          [b52]

                var6        0.6736         0.3046
                            [b61]          [b62]
```

Output 88.9.3 Unique Variances and the Additional Parameter

```
                          Error Variances

               Variable     Parameter      Estimate

               var1         psi1            0.53036
               var2         psi2            0.44986
               var3         psi3            0.48756
               var4         psi4            0.47278
               var5         psi5            0.31125
               var6         psi6            0.53815

                       Additional Parameters

              Type            Parameter      Estimate

              Independent     alpha            0.97825
```

All these estimates are essentially the same as reported in Browne (1982). Notice that there are no standard error estimates in the output, as requested by the NOSE option in the PROC TCALIS 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 88.9.2, the loading estimates of VAR1 on the two factors are 0.3609 and 0.6174, respectively. In Output 88.9.3, 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 TCALIS statement will not solve the problem. By specifying the CORR option you merely request PROC TCALIS 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 stated 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 TCALIS 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 $(\mathbf{BB}' + \Psi)$ 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 ones just presented are the most convenient and intuitive.

Now, due to the fact that discrepancy functions used in PROC TCALIS are derived for covariance matrices rather than correlation matrices, PROC TCALIS is essentially set up for analyzing covariance structures (with or without mean structures), but not correlation structures. Hence, the statistical theory behind PROC TCALIS applies to covariance structure analysis, but it might not generalize to correlation structure analysis in all situations. Despite that, with some manipulations PROC TCALIS 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:

$$\mathbf{P} = \mathbf{BB}' + \Psi$$

As before, in the **B** matrix, there are six linear constraints on the factor loadings. In addition, the diagonal elements of $(\mathbf{BB}' + \Psi)$ 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 TCALIS, a covariance structure model with such correlation structures embedded is now specified. That is, the covariance structure to be fitted by PROC TCALIS is as follows:

$$\mathbf{\Sigma} = \mathbf{DPD}' = \mathbf{D}(\mathbf{BB}' + \Psi)\mathbf{D}'$$

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 TCALIS. However, there seems to be nowhere in PROC TCALIS 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 $\mathbf{T} = \mathbf{DB}$ and $\mathbf{K} = \mathbf{D}\Psi\mathbf{D}'$. The covariance structure model of interest can now be rewritten as:

$$\mathbf{\Sigma} = \mathbf{TT}' + \mathbf{K}$$

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 Ψ 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.

```
proc tcalis data=Kinzer nobs=326 nose;
   factor
      factor1 -> var1-var6 = t11 t21 t31 t41 t51 t61,
      factor2 -> var1-var6 = t12 t22 t32 t42 t52 t62;
   pvar
      factor1-factor2 = 2 * 1.,
      var1-var6       = k1-k6;
   parameters alpha (1.) d1-d6 (6 * 1.)
              b11 b21 b31 b41 b51 b61 (6 *.6),
              b12 b22 b32 b42 b52 b62
              psi1-psi6;
   /* SAS Programming Statements */
   /* 12 Constraints on Correlation structures */
   b12  = alpha - b11;
   b22  = alpha - b21;
   b32  = alpha - b31;
   b42  = alpha - b41;
   b52  = alpha - b51;
   b62  = alpha - b61;
   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;
   /* Defining Covariance Structure Parameters */
   t11  = d1 * b11;
   t21  = d2 * b21;
   t31  = d3 * b31;
   t41  = d4 * b41;
   t51  = d5 * b51;
   t61  = d6 * b61;
   t12  = d1 * b12;
   t22  = d2 * b22;
   t32  = d3 * b32;
   t42  = d4 * b42;
   t52  = d5 * b52;
   t62  = d6 * b62;
   k1   = d1 * d1 * psi1;
   k2   = d2 * d2 * psi2;
   k3   = d3 * d3 * psi3;
   k4   = d4 * d4 * psi4;
   k5   = d5 * d5 * psi5;
   k6   = d6 * d6 * psi6;
   fitindex on(only)=[chisq df probchi];
run;
```

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: $\mathbf{T} = \mathbf{DB}$ and $\mathbf{K} = \mathbf{D}\Psi\mathbf{D}'$, 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 88.9.4. 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 88.9.4 Model Fit of the Correlation Structures

```
                              Fit Summary

                      Chi-Square              14.6269
                      Chi-Square DF                 8
                      Pr > Chi-Square          0.0668
```

Estimates of the loadings and error variances are presented in Output 88.9.5. 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 88.9.5 Estimates of Loadings and Error Variances

```
                         Factor Loading Matrix

                               factor1        factor2

                var1            0.3448         0.6367
                                 [t11]          [t12]

                var2            0.3200         0.6512
                                 [t21]          [t22]

                var3            0.4873         0.4778
                                 [t31]          [t32]

                var4            0.5703         0.3948
                                 [t41]          [t42]

                var5            0.7741         0.1964
                                 [t51]          [t52]

                var6            0.6778         0.3126
                                 [t61]          [t62]

                       Factor Covariance Matrix

                                factor1        factor2

              factor1            1.0000              0
              factor2                 0         1.0000

                            Error Variances

               Variable     Parameter       Estimate

               var1         k1               0.49119
               var2         k2               0.46780
               var3         k3               0.51597
               var4         k4               0.50070
               var5         k5               0.35505
               var6         k6               0.47685
```

The parameter estimates of the embedded correlation structures are shown in Output 88.9.6 as “additional” parameters.

Output 88.9.6 Estimates of Correlation Structure Parameters

```
                          Additional Parameters

              Type           Parameter      Estimate

              Independent    alpha           0.97400
                             d1              1.00771
                             d2              0.99712
                             d3              0.99078
                             d4              0.99085
                             d5              0.99640
                             d6              1.01687
                             b11             0.34217
                             b21             0.32095
                             b31             0.49179
                             b41             0.57553
                             b51             0.77686
                             b61             0.66659
              Dependent      b12             0.63183
                             b22             0.65305
                             b32             0.48222
                             b42             0.39848
                             b52             0.19714
                             b62             0.30742
                             psi1            0.48371
                             psi2            0.47051
                             psi3            0.52561
                             psi4            0.50998
                             psi5            0.35762
                             psi6            0.46116
```

Except for the population standard deviation parameter `d`'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 88.10: A 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 will be 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/qualities were surveyed between 2002 and 2003, Spend02 would represent current purchases and Spend03 would represent 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 like customer service quality and product quality? How would you measure them in the model?

In measuring these hypothetical constructs, one 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/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, structural equation modeling approach is the method of the choice in the current analysis.

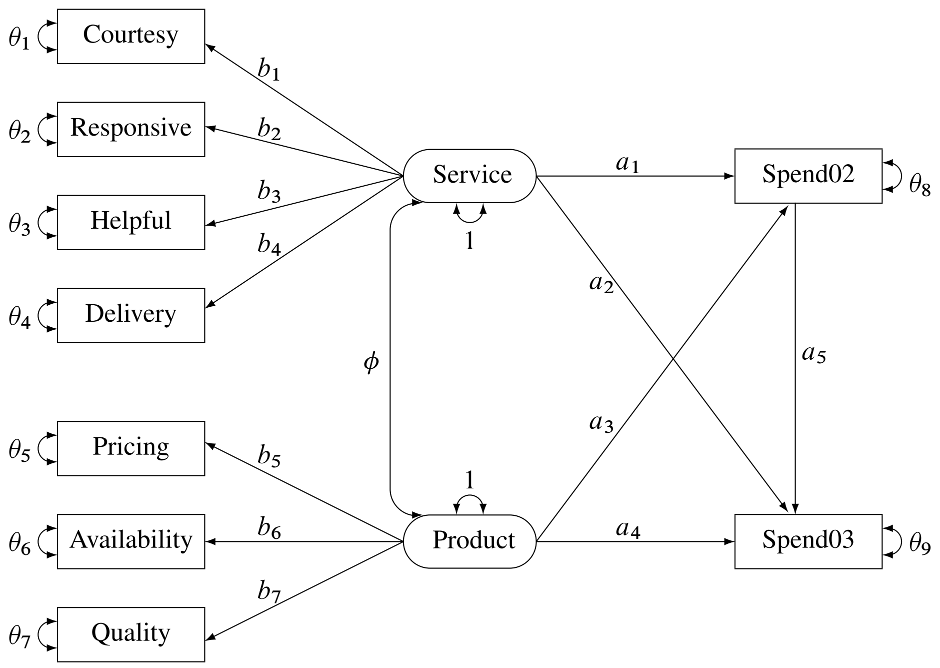
In practice, you must also make sure that there are enough indicators for the identification of the underlying latent factor, and hence the identification of the entire model. Necessary and sufficient rules for identification are very complicated to describe and are out of the scope of the current discussion (however, see Bollen 1989b for discussions of identification rules for various classes of structural equation models). Some simple rules of thumb may 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 that represents 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 is used to represent the structural equation model for the purchase behavior. In Figure 88.5, observed or manifest variables are represented by rectangles, and latent variables are represented by ovals. As mentioned, two latent variables or factors, Service and Product are created as overall measures of customer service and product qualities, respectively.

Figure 88.5 Path Diagram of Purchasing Behavior



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 or effects for this part of functional relationships are represented by a1–a5 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 tagged on these variables represent error variances. In the diagram, θ_1 to θ_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 tagged on 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 are pointing to two variables, they represent covariances in the path diagram. For example, in Figure 88.5 the covariance between Service and Product is represented by the parameter ϕ.

A Macro for the Basic Path Model

For the moment, it is hypothesized that both Region 1 and Region 2 data are fitted by the same model as shown in Figure 88.5. Once the path diagram is drawn, it is readily translated into the PATH modeling language. See the PATH statement on page 6791 for details about how to use the PATH modeling language to specify structural equation models. A macro for the path model in Figure 88.5 using the PATH modeling language is shown in the following:

```
%macro BasepathModel;
path
   Service -> Spend02         a1,
   Service -> Spend03         a2,
   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;
mean
   Service Product = 2 * 0.,
   Spend02         = InterSpend02,
   Spend03         = InterSpend03,
   Courtesy Responsive Helpful
   Delivery Pricing
   Availability Quality  = intercept01-intercept07;
%mend;
```

The name of this macro is BasePathModel. Defining a model by a SAS macro is not essential to run PROC TCALIS. You could have inserted the PATH model code directly into a PROC TCALIS run. The reason for using a macro here is to facilitate the presentation of the subsequent code so that the coding structure is made clearer. In the BasePathModel macro, the PATH statement captures all the path coefficient specification as well as the direction of the paths in the model. The first five paths are for defining how Spend02 and Spend03 are predicted from the latent variables Service, Product, and Spend02. The next seven paths are for defining the measurement model.

The PVAR statement captures the specification of the error variances and the variances of exogenous variables. The PCOV statement captures the specification of covariance between the two latent variables in the model.

The MEAN statement captures the specification of means and intercepts. Unlike some other representation schemes proposed by various researchers in the field, the mean parameters are not depicted in Figure 88.5. 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 exogenous variables, the specifications in the MEAN statement are for the means of these variables.
- For endogenous variables, the specifications in the MEAN statement are for the intercepts of these variables.
- For variables that are not specified in the MEAN statement, their means or intercepts 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 in the model, their specification in the MEAN statement are for nine intercepts: intercept01–intercept07, InterSpend02, and InterSpend03. The means of the latent variables Service and Product are also specified in the MEAN statement as fixed zeros.

A Restrictive Model with Invariant Mean and Covariance Structures

The following shows the SAS data set being analyzed and the PROC TCALIS specification of a restrictive mean and covariance structure model:

```
data region1(type=cov);
   input _type_ $6. _name_ $12. Spend02 Spend03 Courtesy Responsive
         Helpful Delivery Pricing Availability Quality;
   datalines;
COV   Spend02      14.428  2.206  0.439 0.520 0.459 0.498 0.635 0.642 0.769
COV   Spend03       2.206 14.178  0.540 0.665 0.560 0.622 0.535 0.588 0.715
COV   Courtesy      0.439  0.540  1.642 0.541 0.473 0.506 0.109 0.120 0.126
COV   Responsive    0.520  0.665  0.541 2.977 0.582 0.629 0.119 0.253 0.184
COV   Helpful       0.459  0.560  0.473 0.582 2.801 0.546 0.113 0.121 0.139
COV   Delivery      0.498  0.622  0.506 0.629 0.546 3.830 0.120 0.132 0.145
COV   Pricing       0.635  0.535  0.109 0.119 0.113 0.120 2.152 0.491 0.538
COV   Availability  0.642  0.588  0.120 0.253 0.121 0.132 0.491 2.372 0.589
COV   Quality       0.769  0.715  0.126 0.184 0.139 0.145 0.538 0.589 2.753
MEAN     .        183.500 301.921 4.312 4.724 3.921 4.357 6.144 4.994 5.971
;

data region2(type=cov);
   input _type_ $6. _name_ $12. Spend02 Spend03 Courtesy Responsive
         Helpful Delivery Pricing Availability Quality;
   datalines;
COV   Spend02       14.489   2.193 0.442 0.541 0.469 0.508 0.637 0.675 0.769
COV   Spend03        2.193  14.168 0.542 0.663 0.574 0.623 0.607 0.642 0.732
COV   Courtesy       0.442   0.542 3.282 0.883 0.477 0.120 0.248 0.283 0.387
COV   Responsive     0.541   0.663 0.883 2.717 0.477 0.601 0.421 0.104 0.105
COV   Helpful        0.469   0.574 0.477 0.477 2.018 0.507 0.187 0.162 0.205
COV   Delivery       0.508   0.623 0.120 0.601 0.507 2.999 0.179 0.334 0.099
COV   Pricing        0.637   0.607 0.248 0.421 0.187 0.179 2.512 0.477 0.423
COV   Availability   0.675   0.642 0.283 0.104 0.162 0.334 0.477 2.085 0.675
COV   Quality        0.769   0.732 0.387 0.105 0.205 0.099 0.423 0.675 2.698
MEAN     .         156.250 313.670 2.412 2.727 5.224 6.376 7.147 3.233 5.119
;

proc tcalis maxiter=1000 omethod=nrr;
   group 1 / data=region1 label="Region 1" nobs=378;
   group 2 / data=region2 label="Region 2" nobs=423;
   model 1 / group=1,2;
      %BasepathModel
   run;
```

In the PROC TCALIS specification, 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 are 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 BasePathModel macro is included for defining the model for the two groups. Equivalently, you can use two MODEL statements to define two models for the two groups. This way you must also constrain each parameter in the two models. But using a single MODEL statement for the two groups is a short cut for defining such a restrictive model with total parameter invariance among groups.

The OMETHOD=NRR (Newton-Raphson ridge optimization) is used because the default Levenberg-Marquardt optimization did not converge in 1000 iterations.

In Output 88.10.1, a summary of modeling information is presented. 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 88.10.1 Modeling Information and Variables

```
                              Modeling Information

       Group     Label        Data Set          N Obs     Model       Type

           1     Region 1     WORK.REGION1        378     Model 1     PATH
           2     Region 2     WORK.REGION2        423     Model 1     PATH


                        Model 1. Variables in the Model

      Endogenous     Manifest     Availability  Courtesy  Delivery  Helpful
                                  Pricing  Quality  Responsive  Spend02
                                  Spend03
                     Latent
      Exogenous      Manifest
                     Latent       Product  Service

                          Number of Endogenous Variables = 9
                          Number of Exogenous Variables  = 2
```

The optimization converges in 201 iterations. The fit summary table is presented in Output 88.10.2.

Output 88.10.2 Fit Summary

```
                                   Fit Summary

       Modeling Info        N Observations                               801
                            N Variables                                    9
                            N Moments                                    108
                            N Parameters                                  31
                            N Active Constraints                           0
                            Independence Model Chi-Square           399.7468
                            Independence Model Chi-Square DF              72
       Absolute Index       Fit Function                              3.5310
                            Chi-Square                             2821.2798
                            Chi-Square DF                                 77
                            Pr > Chi-Square                           0.0000
                            Z-Test of Wilson & Hilferty              43.2651
                            Hoelter Critical N                            29
                            Root Mean Square Residual (RMSR)         28.2211
                            Standardized RMSR (SRMSR)                 2.1370
                            Goodness of Fit Index (GFI)               0.9996
       Parsimony Index      Adjusted GFI (AGFI)                       0.9995
                            Parsimonious GFI                          1.0690
                            RMSEA Estimate                            0.2987
                            RMSEA Lower 90% Confidence Limit          0.2893
                            RMSEA Upper 90% Confidence Limit          0.3082
                            Probability of Close Fit                       .
                            Akaike Information Criterion           2667.2798
                            Bozdogan CAIC                          2229.4685
                            Schwarz Bayesian Criterion             2306.4685
                            McDonald Centrality                       0.1803
       Incremental Index    Bentler Comparative Fit Index            -7.3732
                            Bentler-Bonett NFI                       -6.0577
                            Bentler-Bonett Non-normed Index          -6.8295
                            Bollen Normed Index Rho1                 -5.5994
                            Bollen Non-normed Index Delta2           -7.5029
                            James et al. Parsimonious NFI            -6.4783
```

The model chi-square statistic is 2818.732. 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 independence model and the structural model defined. The independence 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 2818.132. 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 independence model is deemed inappropriate for the data. The RMSEA for the structural model is .2985, which also indicates a bad model fit. However, 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 is a warning in the output:

```
WARNING: Model 1. Although all predicted variances for the
         latent variables are positive, the corresponding
         predicted covariance matrix is not positive definite.
         It has one negative eigenvalue.
```

PROC TCALIS routinely checks the properties of the predicted covariance matrix. It will issue warnings when there are problems. In this case, the predicted covariance matrix for the latent variables is not positive definite and has one negative eigenvalue. The same fact can also be deduced from the output:

Output 88.10.3 Covariance of Latent Variables

```
                 Model 1. Covariances Among Exogenous Variables

                                                      Standard
 Var1         Var2         Parameter       Estimate        Error         t Value

 Service      Product      phi             -0.78911      0.05542       -14.23750
```

In Output 88.10.3, the covariance between Service and Product is estimated at -1.115. Recall that the variances of these two variables are fixed at ones. Therefore, the predicted covariance matrix for the latent variables is:

$$\begin{pmatrix} 1.000 & -1.115 \\ -1.115 & 1.000 \end{pmatrix}$$

This matrix is certainly not a proper covariance matrix, as the covariance is larger in magnitude than the variances. The eigenvalues for this matrix are 2.115 and $-.115$, indicating a negative definite covariance matrix for the 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.

```
proc tcalis omethod=nrr;
   group 1 / data=region1 label="Region 1" nobs=378;
   group 2 / data=region2 label="Region 2" nobs=423;
   model 1 / groups=1;
      %BasepathModel
   model 2 / groups=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. 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 in 10 iterations without problems. The fit summary table is displayed in Output 88.10.4. 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, GFI, AGFI, Bentler CFI, Bentler-Bonett NFI, Bollen non-normed index delta2 are all close to one, and RMSEA is close to zero.

Output 88.10.4 Fit Summary

```
                                Fit Summary

      Modeling Info       N Observations                              801
                          N Variables                                   9
                          N Moments                                   108
                          N Parameters                                 62
                          N Active Constraints                          0
                          Independence Model Chi-Square          399.7468
                          Independence Model Chi-Square DF             72
      Absolute Index      Fit Function                             0.0371
                          Chi-Square                              29.6131
                          Chi-Square DF                                46
                          Pr > Chi-Square                          0.9710
                          Z-Test of Wilson & Hilferty             -1.8950
                          Hoelter Critical N                         1697
                          Root Mean Square Residual (RMSR)         0.0670
                          Standardized RMSR (SRMSR)                0.0220
                          Goodness of Fit Index (GFI)              1.0000
      Parsimony Index     Adjusted GFI (AGFI)                      1.0000
                          Parsimonious GFI                         0.6389
                          RMSEA Estimate                           0.0000
                          RMSEA Lower 90% Confidence Limit              .
                          RMSEA Upper 90% Confidence Limit              .
                          Probability of Close Fit                 1.0000
                          Akaike Information Criterion           -62.3869
                          Bozdogan CAIC                         -323.9365
                          Schwarz Bayesian Criterion            -277.9365
                          McDonald Centrality                      1.0103
      Incremental Index   Bentler Comparative Fit Index            1.0000
                          Bentler-Bonett NFI                       0.9259
                          Bentler-Bonett Non-normed Index          1.0783
                          Bollen Normed Index Rho1                 0.8840
                          Bollen Non-normed Index Delta2           1.0463
                          James et al. Parsimonious NFI            0.5916
```

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 will give you the same overall chi-square as in Output 88.10.4.

PROC TCALIS also provides a table for comparing relative model fit of the groups. In Output 88.10.5, basic modeling information and some measures of fit for the two groups are shown along with the corresponding overall measures.

Output 88.10.5 Fit Comparison among Groups

```
                                 Fit Comparison Among Groups

                                                            Overall      Region 1      Region 2

Modeling Info    N Observations                                 801           378           423
                 N Variables                                      9             9             9
                 N Moments                                      108            54            54
                 N Parameters                                    62            31            31
                 N Active Constraints                             0             0             0
                 Independence Model Chi-Square             399.7468      173.4482      226.2986
                 Independence Model Chi-Square DF                72            36            36
Fit Index        Fit Function                                0.0371        0.0023        0.0681
                 Percent Contribution to Chi-Square             100             3            97
                 Root Mean Square Residual (RMSR)            0.0670        0.0172        0.0907
                 Standardized RMSR (SRMSR)                   0.0220        0.0057        0.0298
                 Goodness of Fit Index (GFI)                 1.0000        1.0000        1.0000
                 Bentler-Bonett NFI                          0.9259        0.9950        0.8730
```

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, 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 88.10.5, 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-squared distributed under the null hypothesis. Nonetheless, this fact is not important to our group comparison by using the descriptive statistics in Output 88.10.5. 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 ($f_1 = .0023$) than is group 2 (Region 2) by its model ($f_2 = .0681$). 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 RMSR, SRMSR, 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. 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 specification is for such an intermediate model with cross-group constraints.

```
proc tcalis omethod=nrr modification;
   group 1 / data=region1 label="Region 1" nobs=378;
   group 2 / data=region2 label="Region 2" nobs=423;
   model 1 / groups=1;
      %BasepathModel
   model 2 / groups=2;
      refmodel 1;
      mean
         Spend02 Spend03 = G2_InterSpend02 G2_InterSpend03,
         Courtesy Responsive Helpful
         Delivery Pricing Availability
         Quality = G2_intercept01-G2_intercept07;
   simtest
      SpendDiff       = (Spend02Diff Spend03Diff)
      MeasurementDiff = (CourtesyDiff ResponsiveDiff
                         HelpfulDiff DeliveryDiff
                         PricingDiff AvailabilityDiff
                         QualityDiff);
   Spend02Diff      = G2_InterSpend02 - InterSpend02;
   Spend03Diff      = G2_InterSpend03 - InterSpend03;
   CourtesyDiff     = G2_intercept01  - intercept01;
   ResponsiveDiff   = G2_intercept02  - intercept02;
   HelpfulDiff      = G2_intercept03  - intercept03;
   DeliveryDiff     = G2_intercept04  - intercept04;
   PricingDiff      = G2_intercept05  - intercept05;
   AvailabilityDiff = G2_intercept06  - intercept06;
   QualityDiff      = G2_intercept07  - intercept07;
   run;
```

In this specification, Region 1 is fitted by model 1, which is essentially unchanged from the previous analyses. Region 2 is fitted by model 2, which, again, is modified from model 1, as specified in the REFMODEL statement. The modification being made is the addition of unique mean parameters. In the MEAN statement, nine new parameters are specified. All these parameters are prefixed with G2_ to be distinguished from the parameters in model 1. Parameters G2_InterSpend02 and G2_InterSpend03 are intercepts for variables Spend02 and Spend03, respectively. Parameters G2_intercept01–G2_intercept07 are intercepts for the indicator measures of the latent variables. Because the latent variables in the model have means at fixed zeros, the intercept parameters G2_intercept01–G2_intercept07 and G2_InterSpend02 are also the means for the corresponding variables. To summarize, 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.

Additional statistics or tests are requested in the current PROC TCALIS run. The MODIFICATION option in the PROC TCALIS statement requests 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 (or eliminated) without significantly affecting the overall model fit. These tests will be discussed with more details when the results are presented.

In the SIMTEST 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 88.10.6.

Output 88.10.6 Fit Summary

```
                                   Fit Summary

       Modeling Info        N Observations                                 801
                            N Variables                                      9
                            N Moments                                      108
                            N Parameters                                    40
                            N Active Constraints                             0
                            Independence Model Chi-Square             399.7468
                            Independence Model Chi-Square DF                72
       Absolute Index       Fit Function                                0.1346
                            Chi-Square                                107.5461
                            Chi-Square DF                                   68
                            Pr > Chi-Square                             0.0016
                            Z-Test of Wilson & Hilferty                 2.9452
                            Hoelter Critical N                             657
                            Root Mean Square Residual (RMSR)            0.1577
                            Standardized RMSR (SRMSR)                   0.0678
                            Goodness of Fit Index (GFI)                 1.0000
       Parsimony Index      Adjusted GFI (AGFI)                         0.9999
                            Parsimonious GFI                            0.9444
                            RMSEA Estimate                              0.0382
                            RMSEA Lower 90% Confidence Limit            0.0237
                            RMSEA Upper 90% Confidence Limit            0.0514
                            Probability of Close Fit                    0.9275
                            Akaike Information Criterion              -28.4539
                            Bozdogan CAIC                            -415.0924
                            Schwarz Bayesian Criterion               -347.0924
                            McDonald Centrality                         0.9756
       Incremental Index    Bentler Comparative Fit Index               0.8793
                            Bentler-Bonett NFI                          0.7310
                            Bentler-Bonett Non-normed Index             0.8722
                            Bollen Normed Index Rho1                    0.7151
                            Bollen Non-normed Index Delta2              0.8808
                            James et al. Parsimonious NFI               0.6904
```

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 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 .0382, which indicates a good fit. The probability level of close fit is .9969, indicating that a good population fit (RMSEA < 0.05) hypothesis cannot be rejected. GFI, AGFI, and parsimonious GFI all indicate good fit. However, the incremental indices show only respectable model fit.

Comparison of the model fit to the groups is shown in Output 88.10.7.

Output 88.10.7 Fit Comparison among Groups

```
                                Fit Comparison Among Groups

                                                             Overall      Region 1      Region 2

Modeling Info     N Observations                                 801           378           423
                  N Variables                                      9             9             9
                  N Moments                                      108            54            54
                  N Parameters                                    40            31            31
                  N Active Constraints                             0             0             0
                  Independence Model Chi-Square             399.7468      173.4482      226.2986
                  Independence Model Chi-Square DF                72            36            36
Fit Index         Fit Function                                0.1346        0.1261        0.1422
                  Percent Contribution to Chi-Square             100            44            56
                  Root Mean Square Residual (RMSR)            0.1577        0.1552        0.1599
                  Standardized RMSR (SRMSR)                   0.0678        0.0792        0.0557
                  Goodness of Fit Index (GFI)                 1.0000        1.0000        1.0000
                  Bentler-Bonett NFI                          0.7310        0.7260        0.7348
```

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 RMSR, which does not take the sample size into account, the fitting of two groups are about the same. The standardized RMSR 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.

Constrained parameter estimates for the two regions are shown in Output 88.10.8.

Output 88.10.8 Estimates of Path Coefficients and Other Covariance Parameters

```
                              Model 1. PATH List

                                                                Standard
-----------Path------------     Parameter     Estimate         Error       t Value

Service     ->    Spend02        a1             0.37475       0.21318      1.75795
Service     ->    Spend03        a2             0.53851       0.20840      2.58401
Product     ->    Spend02        a3             0.80372       0.21939      3.66347
Product     ->    Spend03        a4             0.59879       0.22144      2.70409
Spend02     ->    Spend03        a5             0.08952       0.03694      2.42326
Service     ->    Courtesy       b1             0.72418       0.07989      9.06482
Service     ->    Responsive     b2             0.90452       0.08886     10.17972
Service     ->    Helpful        b3             0.64969       0.07683      8.45574
Service     ->    Delivery       b4             0.64473       0.09021      7.14677
Product     ->    Pricing        b5             0.63452       0.07916      8.01600
Product     ->    Availability   b6             0.76737       0.08265      9.28516
Product     ->    Quality        b7             0.79716       0.08922      8.93470
```

Output 88.10.8 *continued*

```
                          Model 1. Variance Parameters

Variance                                                      Standard
Type         Variable         Parameter      Estimate            Error        t Value

Error        Courtesy         theta01         1.98374          0.13169       15.06379
             Responsive       theta02         2.02152          0.16159       12.51005
             Helpful          theta03         1.96535          0.12263       16.02727
             Delivery         theta04         2.97542          0.17049       17.45184
             Pricing          theta05         1.93952          0.12326       15.73583
             Availability     theta06         1.63156          0.13067       12.48646
             Quality          theta07         2.08849          0.15329       13.62464
             Spend02          theta08        13.47066          0.71842       18.75051
             Spend03          theta09        13.02883          0.68682       18.96966
Exogenous    Service                          1.00000
             Product                          1.00000

                  Model 1. Covariances Among Exogenous Variables

                                                          Standard
   Var1        Var2        Parameter      Estimate           Error        t Value

   Service     Product     phi             0.33725         0.07061        4.77599
```

All parameter estimates but one are statistically significant at $\alpha = 0.05$. The parameter a1, 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 88.10.9.

Output 88.10.9 Estimates of Means and Intercepts

```
                         Model 1. Means and Intercepts

                                                              Standard
 Type         Variable        Parameter          Estimate        Error        t Value

 Mean         Service                                   0
              Product                                   0
 Intercept    Spend02         InterSpend02      183.50000      0.19585      936.95628
              Spend03         InterSpend03      285.49480      6.78127       42.10048
              Courtesy        intercept01         4.31200      0.08157       52.86519
              Responsive      intercept02         4.72400      0.08679       54.43096
              Helpful         intercept03         3.92100      0.07958       49.27201
              Delivery        intercept04         4.35700      0.09484       45.93968
              Pricing         intercept05         6.14400      0.07882       77.94992
              Availability    intercept06         4.99400      0.07674       65.07315
              Quality         intercept07         5.97100      0.08500       70.24543
```

Output 88.10.9 *continued*

```
                           Model 2. Means and Intercepts

                                                                 Standard
  Type         Variable      Parameter             Estimate         Error      t Value

  Mean         Service                                    0
               Product                                    0
  Intercept    Spend02       G2_InterSpend02      156.25000       0.18511    844.09015
               Spend03       G2_InterSpend03      299.68311       5.77478     51.89515
               Courtesy      G2_intercept01         2.41200       0.07709     31.28628
               Responsive    G2_intercept02         2.72700       0.08203     33.24350
               Helpful       G2_intercept03         5.22400       0.07522     69.45319
               Delivery      G2_intercept04         6.37600       0.08964     71.12697
               Pricing       G2_intercept05         7.14700       0.07450     95.93427
               Availability  G2_intercept06         3.23300       0.07254     44.57020
               Quality       G2_intercept07         5.11900       0.08034     63.71500
```

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 88.10.10 can confirm this.

In Output 88.10.10, there are two simultaneous tests, as requested in the original statements.

Output 88.10.10 Simultaneous Tests

```
                                Simultaneous Tests

 Simultaneous       Parametric            Function
 Test               Function                 Value       DF    Chi-Square    p Value

 SpendDiff                                                2         10458     <.0001
                    Spend02Diff          -27.25000        1         10225     <.0001
                    Spend03Diff           14.18831        1     185.86725     <.0001
 MeasurementDiff                                          7          1610     <.0001
                    CourtesyDiff          -1.90000        1     286.58605     <.0001
                    ResponsiveDiff        -1.99700        1     279.63659     <.0001
                    HelpfulDiff            1.30300        1     141.59942     <.0001
                    DeliveryDiff           2.01900        1     239.35318     <.0001
                    PricingDiff            1.00300        1      85.52567     <.0001
                    AvailabilityDiff      -1.76100        1     278.09360     <.0001
                    QualityDiff           -0.85200        1      53.06240     <.0001
```

The first one is SpendDiff, which tests simultaneously the hypotheses that:

$$H_0 : \text{G2\_InterSpend02 - InterSpend02} = 0$$
$$H_0 : \text{G2\_InterSpend03 - InterSpend03} = 0$$

The exceedingly large chi-square value 10548 suggests the composite null hypothesis is false. Individual tests for these hypotheses suggest each of these hypotheses should be rejected. The chi-square values for individual tests are 10225 and 185.97, 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 asked. The overall customer service (Service) does affect the future purchase (Spend03), but not the current purchase (Spend02), as the path coefficient a1 is only marginally significant. This perhaps 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 88.10.11, ranked LM statistics for the path coefficients in model 1 and model 2 are shown.

Output 88.10.11 LM Tests for Path Coefficients

```
         Model 1. Rank Order of the 10 Largest LM Stat for Path Relations

                                                                      Parm
        To              From                LM Stat     Pr > ChiSq      Change

        Service         Courtesy           11.15249         0.0008      -0.17145
        Service         Helpful             3.09038         0.0788       0.09431
        Service         Delivery            2.59511         0.1072       0.07504
        Courtesy        Responsive          1.75943         0.1847      -0.07730
        Delivery        Courtesy            1.66721         0.1966       0.08669
        Helpful         Courtesy            1.62005         0.2031       0.07277
        Courtesy        Product             1.48928         0.2223      -0.14815
        Service         Product             0.83498         0.3608      -0.12327
        Responsive      Helpful             0.76664         0.3813      -0.05625
        Product         Helpful             0.53020         0.4665      -0.03831
```

Output 88.10.11 *continued*

```
      Model 2. Rank Order of the 10 Largest LM Stat for Path Relations

                                                                 Parm
     To             From               LM Stat    Pr > ChiSq      Change

     Delivery       Courtesy          16.91167        <.0001    -0.26641
     Service        Courtesy           9.11235        0.0025     0.15430
     Courtesy       Delivery           8.12091        0.0044    -0.12989
     Courtesy       Responsive         8.03954        0.0046     0.16215
     Pricing        Responsive         5.48406        0.0192     0.10424
     Courtesy       Product            4.39347        0.0361     0.24412
     Courtesy       Quality            3.52147        0.0606     0.08672
     Service        Delivery           3.20160        0.0736    -0.08281
     Service        Helpful            2.97015        0.0848    -0.09198
     Responsive     Pricing            2.91498        0.0878     0.08943
```

Path coefficients that lead to better improvement (larger chi-square drop) are shown first in the tables. For example, the first path coefficient 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 applications and the theoretical and practical implications. For instance, 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 drop 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 in the bottom of Output 88.10.11. 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 88.10.12.

Output 88.10.12 LM Tests for Error Covariances

```
                        Model 1. Rank Order of the 10 Largest LM
                        Stat for Error Variances and Covariances

     Error          Error                                               Parm
     of             of                  LM Stat     Pr > ChiSq        Change

     Responsive     Helpful             1.26589         0.2605      -0.15774
     Delivery       Courtesy            0.70230         0.4020       0.12577
     Helpful        Courtesy            0.50167         0.4788       0.09103
     Quality        Availability        0.47993         0.4885      -0.09739
     Quality        Pricing             0.45925         0.4980       0.09449
     Responsive     Availability        0.25734         0.6120       0.05965
     Helpful        Availability        0.24811         0.6184      -0.05413
     Responsive     Pricing             0.23748         0.6260      -0.05911
     Spend02        Availability        0.19634         0.6577      -0.13200
     Responsive     Courtesy            0.18212         0.6696       0.06201

                        Model 2. Rank Order of the 10 Largest LM
                        Stat for Error Variances and Covariances

     Error          Error                                               Parm
     of             of                  LM Stat     Pr > ChiSq        Change

     Delivery       Courtesy           16.00996         <.0001      -0.57408
     Responsive     Pricing             4.89190         0.0270       0.25403
     Helpful        Delivery            3.33480         0.0678       0.25299
     Delivery       Availability        2.79513         0.0946       0.20656
     Responsive     Availability        2.16944         0.1408      -0.16421
     Quality        Courtesy            2.14952         0.1426       0.17094
     Responsive     Courtesy            2.12832         0.1446       0.20604
     Quality        Pricing             2.00978         0.1563      -0.19154
     Quality        Availability        1.99477         0.1578       0.19459
     Responsive     Quality             1.88736         0.1695      -0.16963
```

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 simply not shown here for conserving space.

PROC TCALIS 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 will then be 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 6768 for details.

The next group of LM tests is for releasing implicit equality constraints in your model, as shown in Output 88.10.13.

Output 88.10.13 LM Tests for Equality Constraints

```
                Lagrange Multiplier Statistics for Releasing Equality Constraints

        ------------Released Parameter------------                      -----Changes-----
                                                                        Original Released
Parm    Model Type       Var1         Var2           LM Stat Pr > ChiSq     Parm     Parm

a1          1 DV_IV      Spend02      Service        0.01554     0.9008  -0.0213   0.0238
            2 DV_IV      Spend02      Service        0.01554     0.9008   0.0238  -0.0213
a2          1 DV_IV      Spend03      Service        0.01763     0.8944  -0.0222   0.0248
            2 DV_IV      Spend03      Service        0.01763     0.8944   0.0248  -0.0222
a3          1 DV_IV      Spend02      Product      0.0003403     0.9853 -0.00321  0.00355
            2 DV_IV      Spend02      Product      0.0003403     0.9853  0.00355 -0.00321
a4          1 DV_IV      Spend03      Product        0.00176     0.9665  0.00714 -0.00802
            2 DV_IV      Spend03      Product        0.00176     0.9665 -0.00802  0.00714
a5          1 DV_DV      Spend03      Spend02      0.0009698     0.9752 -0.00100  0.00112
            2 DV_DV      Spend03      Spend02      0.0009698     0.9752  0.00112 -0.00100
b1          1 DV_IV      Courtesy     Service       19.17225     <.0001   0.2851  -0.3191
            2 DV_IV      Courtesy     Service       19.17225     <.0001  -0.3191   0.2851
b2          1 DV_IV      Responsive   Service        0.21266     0.6447  -0.0304   0.0341
            2 DV_IV      Responsive   Service        0.21266     0.6447   0.0341  -0.0304
b3          1 DV_IV      Helpful      Service        4.60629     0.0319  -0.1389   0.1555
            2 DV_IV      Helpful      Service        4.60629     0.0319   0.1555  -0.1389
b4          1 DV_IV      Delivery     Service        3.59763     0.0579  -0.1508   0.1687
            2 DV_IV      Delivery     Service        3.59763     0.0579   0.1687  -0.1508
b5          1 DV_IV      Pricing      Product        0.50974     0.4753   0.0468  -0.0524
            2 DV_IV      Pricing      Product        0.50974     0.4753  -0.0524   0.0468
b6          1 DV_IV      Availability Product        0.57701     0.4475  -0.0457   0.0512
            2 DV_IV      Availability Product        0.57701     0.4475   0.0512  -0.0457
b7          1 DV_IV      Quality      Product        0.00566     0.9400 -0.00511  0.00574
            2 DV_IV      Quality      Product        0.00566     0.9400  0.00574 -0.00511
theta01     1 COVERR     Courtesy     Courtesy      45.24725     <.0001   0.7204  -0.8064
            2 COVERR     Courtesy     Courtesy      45.24725     <.0001  -0.8064   0.7204
theta02     1 COVERR     Responsive   Responsive     1.73499     0.1878  -0.1555   0.1740
            2 COVERR     Responsive   Responsive     1.73499     0.1878   0.1740  -0.1555
theta03     1 COVERR     Helpful      Helpful       11.13266     0.0008  -0.3448   0.3860
            2 COVERR     Helpful      Helpful       11.13266     0.0008   0.3860  -0.3448
theta04     1 COVERR     Delivery     Delivery       4.99097     0.0255  -0.3364   0.3766
            2 COVERR     Delivery     Delivery       4.99097     0.0255   0.3766  -0.3364
theta05     1 COVERR     Pricing      Pricing        2.86428     0.0906   0.1729  -0.1936
            2 COVERR     Pricing      Pricing        2.86428     0.0906  -0.1936   0.1729
theta06     1 COVERR     Availability Availability   2.53147     0.1116  -0.1494   0.1672
            2 COVERR     Availability Availability   2.53147     0.1116   0.1672  -0.1494
theta07     1 COVERR     Quality      Quality        0.07328     0.7866  -0.0315   0.0352
            2 COVERR     Quality      Quality        0.07328     0.7866   0.0352  -0.0315
theta08     1 COVERR     Spend02      Spend02        0.00214     0.9631   0.0304  -0.0340
            2 COVERR     Spend02      Spend02        0.00214     0.9631  -0.0340   0.0304
theta09     1 COVERR     Spend03      Spend03      0.0001773     0.9894 -0.00842  0.00946
            2 COVERR     Spend03      Spend03      0.0001773     0.9894  0.00946 -0.00842
phi         1 COVEXOG    Service      Product        0.87147     0.3505   0.0605  -0.0678
            2 COVEXOG    Service      Product        0.87147     0.3505  -0.0678   0.0605
```

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 88.10.13 shows you which parameter can be unconstrained so that your overall model fit might improve. For example, if you unconstrain the first parameter a1 for the two models, the expected chi-square drop (LM Stat) is about 0.0155, which is not significant ($p = .9008$). The associated parameter changes are small too. However, if you consider unconstraining parameter b1, the expected drop of chi-square is 19.17 ($p < 0.0001$). 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 b1 as the parameter for the Courtesy <– Service path in model 2), and fit the model again, the new estimate of b1 will be 0.2851 larger than the previous b1 estimate. The estimate of "new" would be 0.3191 less than the previous b1 estimate. The second row for the b1 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 big expected improvement of model fit is by freeing the constrained variances of Courtesy among the two models. The corresponding row to look at is the row with parameter theta01, where the parameter type is labeled "COVERR" and the values for the Var1 and Var2 are both "Courtesy." The LM statistic is 45.247, which is a significant chi-square drop if you free either parameter locations. If you choose to rename the error variance for Courtesy in model 1, the new theta01 estimate will be 0.8064 smaller than the original theta01 estimate. The new estimate of the error variance for Courtesy in model 2 will be 0.7204 larger than the previous theta01 estimate. Finally, the constrained parameter theta03, 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 TCALIS 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 88.10.14.

Output 88.10.14 Wald Tests

```
                        Stepwise Multivariate Wald Test

                  ------Cumulative Statistics-----    --Univariate Increment--
Parm              Chi-Square       DF     Pr > ChiSq    Chi-Square     Pr > ChiSq

a1                   3.09039        1         0.0788       3.09039         0.0788
```

In Output 88.10.14, you see that a1 is suggested to be a fixed zero parameter (or 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 3.090 (p=.079), which is only marginally significant at $\alpha = .05$.

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.

References

Akaike, H. (1974), "A New Look at the Statistical Identification Model," *IEEE Transactions on Automatic Control*, 19, 716–723.

Akaike, H. (1987), "Factor Analysis and AIC," *Psychometrika*, 52, 317–332.

Beale, E. M. L. (1972), "A Derivation of Conjugate Gradients," in *Numerical Methods for Nonlinear Optimization*, ed. F. A. Lootsma, London: Academic Press.

Bentler, P. M. (1985), *Theory and Implementation of EQS: A Structural Equations Program*, Manual for Program Version 2.0, Los Angeles: BMDP Statistical Software.

Bentler, P. M. (1986), *Lagrange Multiplier and Wald Tests for EQS and EQS/PC*, Los Angeles: BMDP Statistical Software.

Bentler, P. M. (1995), *EQS, Structural Equations Program Manual*, Program Version 5.0, Encino, CA: Multivariate Software.

Bentler, P. M. and Bonett, D. G. (1980), "Significance Tests and Goodness of Fit in the Analysis of Covariance Structures," *Psychological Bulletin*, 88, 588–606.

Bentler, P. M. and Freeman, E. H. (1983), "Test for Stability in Linear Structural Equation Systems," *Psychometrika*, 48, 143–145.

Bentler, P. M. and Weeks, D. G. (1980), "Linear Structural Equations with Latent Variables," *Psychometrika*, 45, 289–308.

Bishop, Y. M. M. , Fienberg, S. E., and Holland, P. W. (1977), *Discrete Multivariate Analysis: Theory and Practice*, Cambridge and London: MIT Press.

Bollen, K. A. (1986), "Sample Size and Bentler and Bonett's Nonnormed Fit Index," *Psychometrika*, 51, 375–377.

Bollen, K. A. (1989a), "A New Incremental Fit Index for General Structural Equation Models," *Sociological Methods and Research*, 17, 303–316.

Bollen, K. A. (1989b), *Structural Equations with Latent Variables*, New York: John Wiley & Sons.

Box, G. E. P. (1949), "A General Distribution Theory for a Class of Likelihood Criteria," *Biometrika*, 36, 317–346.

Bozdogan, H. (1987), "Model Selection and Akaike's Information Criterion (AIC): The General Theory and its Analytical Extensions," *Psychometrika*, 52, 345–370.

Browne, M. W. (1974), "Generalized Least Squares Estimators in the Analysis of Covariance Structures," *South African Statistical Journal*, 8, 1–24.

Browne, M. W. (1982), "Covariance Structures," in *Topics in Multivariate Analyses*, ed. D. M. Hawkins, New York: Cambridge University Press.

Browne, M. W. (1984), "Asymptotically Distribution-Free Methods for the Analysis of Covariance Structures," *British Journal of Mathematical and Statistical Psychology*, 37, 62–83.

Browne, M. W. and Cudeck, R. (1993), "Alternative Ways of Assessing Model Fit," in *Testing Structural Equation Models*, ed. K. A. Bollen and S. Long, Newbury Park, CA: Sage Publications.

Browne, M. W. and Du Toit, S. H. C. (1992), "Automated Fitting of Nonstandard Models," *Multivariate Behavioral Research*, 27, 269–300.

Browne, M. W. and Shapiro, A. (1986), "The Asymptotic Covariance Matrix of Sample Correlation Coefficients under General Conditions," *Linear Algebra and Its Applications*, 82, 169–176.

Bunch, J. R. and Kaufman, K. (1977), "Some Stable Methods for Calculating Inertia and Solving Symmetric Linear Systems," *Mathematics of Computation*, 31, 162–179.

Buse, A. (1982), "The Likelihood Ratio, Wald, and Lagrange Multiplier Tests: An Expository Note," *The American Statistician*, 36, 153–157.

Chamberlain, R. M., Powell, M. J. D., Lemarechal, C., and Pedersen, H. C. (1982), "The Watchdog Technique for Forcing Convergence in Algorithms for Constrained Optimization," *Mathematical Programming*, 16, 1–17.

Crawford, C. B. and Ferguson, G. A. (1970), "A General Rotation Criterion and Its Use in Orthogonal Rotation," *Psychometrika,* 35, 321–332.

DeLeeuw, J. (1983), "Models and Methods for the Analysis of Correlation Coefficients," *Journal of Econometrics*, 22, 113–137.

Dennis, J. E. and Mei, H. H. W. (1979), "Two New Unconstrained Optimization Algorithms Which Use Function and Gradient Values," *Journal of Optimization Theory and Applications*, 28, 453–482.

Dijkstra, T. K. (1992), "On Statistical Inference with Parameter Estimates on the Boundary of the Parameter Space," *British Journal of Mathematical and Statistical Psychology*, 45, 289–309.

Everitt, B. S. (1984), *An Introduction to Latent Variable Methods*, London: Chapman & Hall.

Fletcher, R. (1980), *Practical Methods of Optimization*, Vol. 1, Chichester: John Wiley & Sons.

Fletcher, R. (1987), *Practical Methods of Optimization*, Second Edition, Chichester: John Wiley & Sons.

Fuller, W. A. (1987), *Measurement Error Models*, New York: John Wiley & Sons.

Gay, D. M. (1983), "Subroutines for Unconstrained Minimization," *ACM Transactions on Mathematical Software*, 9, 503–524.

Gill, E. P., Murray, W., Saunders, M. A., and Wright, M. H. (1984), "Procedures for Optimization Problems with a Mixture of Bounds and General Linear Constraints," *ACM Transactions on Mathematical Software*, 10, 282–298.

Guttman, L. (1957), "Empirical Verification of the Radex Structure of Mental Abilities and Personality Traits," *Educational and Psychological Measurement*, 17, 391–407.

Hägglund, G. (1982), “Factor Analysis by Instrumental Variable Methods,” *Psychometrika*, 47, 209–222.

Harman, H. H. (1976), *Modern Factor Analysis,* Third Edition, Chicago: University of Chicago Press.

Hoelter, J. W. (1983), “The Analysis of Covariance Structures: Goodness-of-Fit Indices,” *Sociological Methods and Research*, 11, 325–344.

Holzinger, K. J. and Swineford, F. (1937), “The Bi-Factor Method,” *Psychometrika*, 2, 41–54.

James, L. R., Mulaik, S. A., and Brett, J. M. (1982), *Causal Analysis: Assumptions, Models, and Data*, Beverly Hills, CA: Sage Publications.

Jennrich, R. I. (1973), “Standard Errors for Obliquely Rotated Factor Loadings,” *Psychometrika,* 38, 593–604.

Jennrich, R. I. (1987), “Tableau Algorithms for Factor Analysis by Instrumental Variable Methods,” *Psychometrika*, 52, 469–476.

Jöreskog, K. G. (1973), “A General Method for Estimating a Linear Structural Equation System,” in *Structural Equation Models in the Social Sciences*, ed. A. S. Goldberger and O. D. Duncan, New York: Seminar Press.

Jöreskog, K. G. (1978), “Structural Analysis of Covariance and Correlation Matrices,” *Psychometrika*, 43, 443–477.

Jöreskog, K. G. and Sörbom, D. (1985), *LISREL VI; Analysis of Linear Structural Relationships by Maximum Likelihood, Instrumental Variables, and Least Squares*, Uppsala: University of Uppsala.

Jöreskog, K. G. and Sörbom, D. (1988), *LISREL 7: A Guide to the Program and Applications*, Chicago: SPSS Inc.

Keesling, J. W. (1972), “Maximum Likelihood Approaches to Causal Analysis,” Ph. D. dissertation, University of Chicago, 1972.

Kmenta, J. (1971), *Elements of Econometrics*, New York: Macmillan.

Lawley, D. N. and Maxwell, A. E. (1971), *Factor Analysis as a Statistical Method*, New York: American Elsevier.

Lee, S. Y. (1985), “On Testing Functional Constraints in Structural Equation Models,” *Biometrika*, 72, 125–131.

Loehlin, J. C. (2004), *Latent Variable Models, An Introduction to Factor, Path, and Structural Analysis (4th ed.)*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Long, J. S. (1983), *Covariance Structure Models, an Introduction to LISREL*, Beverly Hills, CA: Sage Publications.

MacCallum, R. (1986), “Specification Searches in Covariance Structure Modeling,” *Psychological Bulletin*, 100, 107–120.

MacCallum, R. C., Roznowski, M., and Necowitz, L. B. (1992), "Model Modification in Covariance Structure Analysis: The Problem of Capitalization on Chance," *Psychological Bulletin*, 111, 490–504.

Mauchly, J. W. (1940), "Significance Test for Sphericity of a Normal N-Variate Distribution," *Annals of Mathematical Statistics*, 11, 204–209.

McArdle, J. J. (1980), "Causal Modeling Applied to Psychonomic Systems Simulation," *Behavior Research Methods & Instrumentation*, 12, 193–209.

McArdle, J. J. (1988), "Dynamic but Structural Equation Modeling of Repeated Measures Data," in *The Handbook of Multivariate Experimental Psychology*, ed. J. R. Nesselroade and R. B. Cattell, New York: Plenum Press.

McArdle, J. J. and McDonald, R. P. (1984), "Some Algebraic Properties of the Reticular Action Model," *British Journal of Mathematical and Statistical Psychology*, 37, 234–251.

McDonald, R. P. (1978), "A Simple Comprehensive Model for the Analysis of Covariance Structures," *British Journal of Mathematical and Statistical Psychology*, 31, 59–72.

McDonald, R. P. (1980), "A Simple Comprehensive Model for the Analysis of Covariance Structures: Some Remarks on Applications," *British Journal of Mathematical and Statistical Psychology*, 33, 161–183.

McDonald, R. P. (1985), *Factor Analysis and Related Methods*, Hillsdale, NJ: Lawrence Erlbaum Associates.

McDonald, R. P. (1989), "An Index of Goodness-of-Fit Based on Noncentrality," *Journal of Classification*, 6, 97–103.

McDonald, R. P. and Hartmann, W. (1992), "A Procedure for Obtaining Initial Values of Parameters in the RAM Model," *Multivariate Behavioral Research*, 27, 57–176.

Moré, J. J. (1978), "The Levenberg-Marquardt Algorithm: Implementation and Theory," in *Numerical Analysis–Dundee 1977*, ed. G. A. Watson, Lecture Notes in Mathematics 630, Berlin: Springer-Verlag.

Moré, J. J. and Sorensen, D. C. (1983), "Computing a Trust-Region Step," *SIAM Journal on Scientific and Statistical Computing*, 4, 553–572.

Mulaik, S. A., James, L. R., Van Alstine, J., Bennett, N., Lind, S., and Stilwell, C. D. (1989), "Evaluation of Goodness-of-Fit Indices for Structural Equation Models," *Psychological Bulletin*, 105, 430–445.

Mulaik, S. A. and Quartetti, D. A. (1997). "First Order or Higher Order General Factor," *Structural Equation Modeling*, 4, 193–211.

Polak, E. (1971), *Computational Methods in Optimization*, New York: Academic Press.

Powell, J. M. D. (1977), "Restart Procedures for the Conjugate Gradient Method," *Mathematical Programming*, 12, 241–254.

Powell, J. M. D. (1978a), "A Fast Algorithm for Nonlinearly Constraint Optimization Calculations," in *Numerical Analysis, Dundee 1977, Lecture Notes in Mathematics 630*, ed. G. A. Watson, Berlin: Springer Verlag, 144–175.

Powell, J. M. D. (1978b), "Algorithms for Nonlinear Constraints That Use Lagrangian Functions," *Mathematical Programming*, 14, 224–248.

Powell, M. J. D. (1982a), "Extensions to Subroutine VF02AD," in *Systems Modeling and Optimization, Lecture Notes In Control and Information Sciences 38*, ed. R. F. Drenick and F. Kozin, Berlin: Springer Verlag, 529–538.

Powell, J. M. D. (1982b), "VMCWD: A Fortran Subroutine for Constrained Optimization," *DAMTP 1982/NA4*, Cambridge, England.

Schmid, J. and Leiman, J. M. (1957), "The Development of Hierarchical Factor Solutions," *Psychometrika*, 22, 53–61.

Schwarz, G. (1978), "Estimating the Dimension of a Model," *Annals of Statistics*, 6, 461–464.

Sclove, L. S. (1987), "Application of Model-Selection Criteria to Some Problems in Multivariate Analysis," *Psychometrika*, 52, 333–343.

Steiger, J. H. (1998), "A Note on Multiple Sample Extensions of the RMSEA Fit Index," *Structural Equation Modeling*, 5, 411–419.

Steiger, J. H. and Lind, J. C. (1980), "Statistically Based Tests for the Number of Common Factors," paper presented at the annual meeting of the Psychometric Society, Iowa City, IA.

Swaminathan, H. (1974), "A General Factor Model for the Description of Change," Report LR-74-9, Laboratory of Psychometric and Evaluative Research, University of Massachusetts.

Tucker, L. R. and Lewis, C. (1973), "A Reliability Coefficient for Maximum Likelihood Factor Analysis," *Psychometrika*, 38, 1–10.

Wheaton, B., Muthèn, B., Alwin, D. F., and Summers, G. F. (1977), "Assessing Reliability and Stability in Panel Models," in *Sociological Methodology*, ed. D. R. Heise, San Francisco: Jossey Bass.

Wiley, D. E. (1973), "The Identification Problem for Structural Equation Models with Unmeasured Variables," in *Structural Equation Models in the Social Sciences*, ed. A. S. Goldberger and O. D. Duncan, New York: Seminar Press, 69–83.

Wilson, E. B. and Hilferty, M. M. (1931), "The Distribution of Chi-Square," *Proceeding of the National Academy of Science*, 17, 694.

Yung, Y. F., Thissen, D., and McLeod, L. D. (1999), "On the Relationship Between the Higher-order Factor Model and the Hierarchical Factor Model," *Psychometrika*, 64, 113–128.

Subject Index

Syntax Index

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **`yourturn@sas.com`**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **`suggest@sas.com`**.