INTRODUCTION

This paper contains a description of some of the changes and enhancements to the SAS/ETS product under the Version 5 Release of the SAS System. Because a similar discussion of changes and enhancements was given last year at the 1984 SUGI conference (DeLong, Ihnen and Little, 1984), this paper will concentrate on illustrating the additions with examples.

Briefly, the enhancements to the procedures covered are:

**COMPUTAB**
An output dataset is now available. You can access table cells by using the special name `TABLE` with row and column indexes. Several depreciation functions are now available.

**FORECAST**
A seasonal forecasting method, the Holt-Winters method, has been added. Also, you can now specify ALPHA values for the prediction confidence intervals.

**PDLREG**
This is a new procedure to fit Polynomial Distributed Lag Regression models. A correction can also be made for autocorrelation in the residuals.

**AUTOREG**
The observations used to fit the model no longer need to come from a contiguous stretch of data with no missing values. Confidence intervals are computed for the predicted values. Maximum likelihood and exact least squares estimates are available as optional estimation methods.

**ARIMA**
This procedure is now interactive at the statement level. Also, the noise parameters are restricted to the stationary and invertible regions unless the user specifies the `NOSTABLE` option.

**SYSLIN**
This is a replacement program for SYSREG. Three new estimation methods are provided: minimum expected loss, k-class and iterated three-stage-least-squares. Reduced form estimates, a test for overidentification and residual plots are now available.

**MODEL-SYSNLIN-SIMNLIN**
PROC MODEL is no longer necessary. A RANGE statement has been added and missing values are fully supported. All data step programming statements are supported. Models can be specified and developed in sections which can later be merged into larger models. Model files are now special SAS data sets which can be manipulated with standard SAS utilities. Lag logic and moving average models are now simplified. All SAS library functions can be used in models as well as external user-written subroutines. A FIT statement has been added to SYSNLIN which controls estimation of subsets of parameters, initial grid searches and equations included in the estimation. Several options have been added to SIMNLIN to support simulation of models.

In addition, three new macros are included:

- `%READDR1`: This is a macro for reading data tapes containing data extracted from Data Resources Inc. data banks.
- `%PDL`: This is a macro to automate the use of polynomial distributed lags with the SYSNLIN-SIMNLIN modeling programs.
- `%AR`: This macro aids in the specification of autoregressive models, either univariate or multivariate, for the error terms in SYSNLIN and SIMNLIN.

**COMPUTAB Enhancements**

The COMPUTAB procedure can now have an output data set. Thus, tables can be output for inclusion in subsequent tables. The data set contains BY variables, special variables `_TYPE_` and `_NAME_`, and row or column variables.

SAS/ETS is a trademark of SAS Institute Inc., Cary, NC, USA
The NOTRANS option stops the automatic transposition of the input data set. When you specify NOTRANS, observations become rows, and data set variables become columns of the table. The NOTRANS option affects the input block, the precedence of row and column options, and the structure of the output data set.

You can insert or retrieve values from specific table cells by using the special name TABLE with row and column subscripts (TABLE{rowindex, columnindex}). The terms rowindex and columnindex can be numbers, row and column names, or expressions that produce row and column numbers or names. The term TABLE{rowindex, columnindex} can appear on either side of an equal sign and can be used as a term in an expression.

There are several new financial functions for calculating depreciation. The depreciation methods include straight line, declining balance, declining balance changing to straight line, sum-of-years-digits, and a general table driven depreciation. There are functions for both single period and cumulative depreciation.

The FORECAST Procedure

Under the Version 5 Release, the FORECAST procedure has another forecasting method, the Holt-Winters method, which is designed for seasonal data. Additionally, (1-α) confidence limits can now be requested for all forecast methods where α can be specified in the range of .01 to .99.

The Holt-Winters forecasts are generated by the following set of equations as described by Montgomery and Johnson (1976) and Makridakis, Wheelwright, and McGee (1983). This method is requested by specifying METHOD=WINTERS on the FORECAST procedure statement. The WINTERS method uses updating equations similar to exponential smoothing to fit parameters for the model

\[ x_t = (a + b \cdot t + c \cdot t^2) \cdot s_t + \varepsilon_t \]

where \( a, b, \) and \( c \) are the true trend parameters, and the factor \( s_t \) is the true seasonal parameter for the season corresponding to time \( t \).

The standard Holt-Winters method uses a linear trend. However, PROC FORECAST can fit versions of the method using a constant, linear or quadratic trend. The degree of the trend is controlled by the TREND option.

The estimates at time \( t \) of the constant, linear, and quadratic trend parameters, \( a_t, b_t, \) and \( c_t \), are updated as follows:

\[ a_t = w_1 \cdot \frac{x_t}{s_{t-L}} + (1-w_1) \cdot (a_{t-1} + b_{t-1} + c_{t-1}) \]

\[ b_t = w_2 \cdot (a_t - a_{t-1} + c_{t-1}) + (1-w_2) \cdot (b_{t-1} + c_{t-1}) \]

\[ c_t = w_2 \left( b_t - b_{t-1} \right) / 2 + (1-w_2) \cdot c_{t-1} \]

where \( w_1 \) and \( w_2 \) are smoothing weights.

The seasonal parameters are updated when the season changes, using the mean of the ratios of the actual to the predicted values for the season. For example, the seasons can be months and the observations daily. The update for January's seasonal factor then depends on the performance of predicted values averaged over the month of January. In the simplest case of one observation for each season, the seasonal parameters are updated using the formula

\[ s_t = \frac{w_3 \cdot x_t / a_t + (1-w_3) \cdot s_{t-L}}{1} \]

where \( s_t \) is the (updated) seasonal parameter, \( x_t \) is the actual value of the series at time \( t \), \( a_t \) is the smoothed value of the series at time \( t \), \( L \) is the number of seasonal factors which comprise an entire cycle, and \( w_3 \) is the seasonal smoothing weight. Multiple seasons can be used with a separate set of seasonal factors for each specified seasonality. The weights, \( w_1, w_2, w_3 \), which control the amount of smoothing, can be specified or they can be allowed to remain at the default values.

In this updating system, the trend polynomial is always centered at the current period so that the intercept parameter of the trend polynomial for predicted values at times after \( t \) is always the updated intercept parameter \( a_t \). The predicted value for \( r \) periods ahead is

\[ x_{h+1} = \left( a_t + b \cdot t + c \cdot t^2 \right) \cdot s_{t+r} \]

The parameters are normalized so that the seasonal factors for each complete cycle have a mean of 1.0. This normalization is performed after each complete cycle and at the end of the data. Thus, if INTERVAL=MONTH and SEASONS=MONTH and a series begins with a July value, then the seasonal factors for the series are normalized at observations for July and at the last observation in the data set. The normalization is performed by dividing each of the seasonal factors, and multiplying each of the trend parameters, by the mean of the unnormalized seasonal parameters.

A method for calculating exact forecast confidence limits for the Winters method is not available. Therefore, the approach taken in
PROC FORECAST is to assume that the true seasonal factors have small variability about a set of fixed seasonal factors and that the remaining variation of the series is small relative to the mean level of the series. That is, the equations are written

\[ s_t = \mu (1 + \delta_t) \]
\[ x_t = \mu \alpha_t (1 + \theta_t) \]
\[ a_t = \mu (1 + \alpha_t) \]

where \( \mu \) is the mean level and \( \alpha_t \) are the fixed seasonal factors. Assuming that \( \alpha_t \) and \( \delta_t \) are small, the forecast equations are linearized with only first-order terms in \( \delta_t \) and \( \alpha_t \) kept. For the case of a constant trend model, \( \text{TREND}=1 \), the following system results:

\[ a_t = \omega_1 (r_t - \mu) + (1 - \omega_1) a_{t-1} \]
\[ \delta_t = \omega_2 (r_t - \mu) + (1 - \omega_2) \delta_{t-1} \]

In terms of forecasts for \( r_t \), this linearized system is equivalent to a seasonal ARIMA model. Confidence limits for \( r_t \) are based on this ARIMA model and converted into confidence limits for \( x_t \) using \( s_t \) as estimates \( \mu \).

Example: A Forecast of Nondurable Goods Using Winters Method

In this example we generate forecasts of the retail sales of nondurable goods using the Holt-Winters method. The data consist of 8 years of monthly sales and have a strong seasonal component. Forecasts are generated for the next 15 months. The resulting forecasts are plotted in Figure 1.

```
PROC FORECAST DATA=A OUT=B TREND=2 OUTDATA OUT1STEP OUTLIMIT INTERVAL=MONTH LEAD=15 METHOD=WINTERS SEASONS=MONTH ALPHA=.10;
   ID DATE;
   VAR NONDUR;
```

PROC GPLOT DATA=B;
   PLOT (NONDUR) DATE=_TYPE_; TITLE 'PLOT OF FORECAST: WINTERS METHOD'; SYMBOL1 I=J; SYMBOL2 I=J; SYMBOL3 I=J;
```

The PDREG Procedure and Distributed Lag Models

The PDREG procedure which fits (Almon) polynomial distributed lag regression models is a new addition to the SAS/ETS library. A distributed lag model allows a change in an independent variable to affect not only the current value of the dependent variable but later values as well. The form of a distributed lag model for \( y_t \) is (we illustrate using only one independent variable)

\[ y_t = a + \sum_{j=1}^{p} b_j x_{t-j} + \epsilon_t \]

where \( x_t \) are the values of the independent variable, \( b_j \) are the coefficients of the lag distribution, and \( \epsilon_t \) is the error term. For the polynomial distributed lag, the coefficient values are parameterized as the values of a polynomial over some finite range. Outside this range the lag coefficients are set to zero. Thus, given the range for the polynomial and the degree, \( q \), the lag coefficients are

\[ b_j = \sum_{k=0}^{q} a_k j^k \quad j=0,1,\ldots, p. \]

where \( a_k \) are the free polynomial parameters. Often endpoint restrictions of the form \( b_{p+1}=0 \) or \( b_0=0 \) are imposed as well. If specified, the PDREG procedure can also correct for autocorrelation in the error term by fitting an autoregressive model to the residuals and refitting the model using generalized least squares.

The syntax for the MODEL statement of the PDREG procedure is similar to that of the other regression procedures but allows each independent variable to be followed by a polynomial lag specification of the form

\[ (p,q_{\text{max}}^{q_{\text{min}}},\text{FIRST}|\text{LAST}|\text{BOTH}) \]

where \( p \) specifies the range of the lag distribution, \( q_{\text{max}} \) specifies the maximum degree polynomial to be fit, \( q_{\text{min}} \) specifies the minimum degree polynomial to be fit, and the last argument indicates what endpoint restrictions are to be imposed. Specifying \text{FIRST} imposes the constraint, \( b_{p+1}=0 \), specifying \text{LAST} imposes the constraint, \( b_0=0 \), and specifying \text{BOTH} imposes both endpoint constraints. The default is to impose no endpoint constraints. For example,

\[ X(9,4,2,LAST) \]

specifies that the variable \( X \) be modeled using a polynomial distributed lag of length 9, that is lags 0 through 9, and that polynomials of degree 2, 3, and 4 with right endpoint restrictions be used to model the lag distribution.
%PDL Macro

Polynomial and other distributed lag models can also be estimated with PROC SYSLIN and simulated or forecast with PROC SIMLIN. Polynomial distributed lag models are so common that a SAS macro, %PDL, has been developed to generate automatically the programming statements to compute the lag coefficients and to apply them to the lags. The source for this macro is included in the sample library distributed with the SAS/ETS Software product. (See your SAS user representative to find out how to access the SAS/ETS sample library at your site.)

To use the %PDL macro in a model program, you first call it to declare the lag distribution; later you call it again to apply the PDL to a variable or expression. The first call generates a PARMS statement for the polynomial parameters and assignment statements to compute the lag coefficients. The second call generates an expression which applies the lag coefficients to the lags of the specified variable or expression. A PDL can be declared only once, but it can be used any number of times (that is, the second call can be repeated).

The initial declaratory call has the form

%PDL(pdlname, nlags, degree <, R=code> )

where pdlname is a name (up to 8 characters) that you give to identify the PDL, nlags is the lag length, and degree is the degree of the polynomial for the distribution. R=code is optional for endpoint restrictions. The code can be FIRST, LAST, or BOTH, the same as for the POL REG procedure.

The later calls to apply the PDL have the form

%PDL(pdlname, expression)

where pdlname is the name of the PDL and expression is the variable or expression to which the PDL is to be applied. The pdlname given must be the same as the name used to declare the PDL.

Example of the Use of the %PDL Macro

PROC SYSLIN;
ENDO Y1 Y2; EXO X Z; PARMS INT1 INT2;
%PDL(LOGXPDL,5,3)
%PDL(ZPDL,6,4)
Y1 = INT1 + %PDL(LOGXPDL,LOG(X)) + %PDL(ZPDL,Z);
Y2 = INT2 + %PDL(ZPDL,Z);

This example models two variables, Y1 and Y2, and uses two PDLs. A (5,3) PDL of the log of X is used in the equation for Y1. A (6,4) PDL of Z is used in the equations for both Y1 and Y2. Since the same ZPDL is used in both equations, the lag coefficients for Z are the same for the Y1 and Y2 equations, and the polynomial parameters for ZPDL are shared by the two equations.

Rational Distributed Lags

Distributed lag models with rational (Jorgenson) lag distributions can be fit with the ARIMA procedure where they are usually referred to as transfer function models. In this case, the coefficients of the lag distribution are modeled as the solution to a difference equation, that is

\[ \phi(z_j) b(z_j) = \theta(z_j) \]

where \( \phi_j \) and \( \theta_j \) are the free parameters of the lag distribution. This kind of lag distribution is often written as

\[ b(z) = \theta(z) / \phi(z) \]

where \( b(z) = \theta_1 z_1 + \theta_2 z_2 + \theta_3 z_3 + \theta_4 z_4 \) and \( \phi(z) = \phi_1 z_1 + \phi_2 z_2 + \phi_3 z_3 + \phi_4 z_4 \). The two polynomials, \( \theta(z) \) and \( \phi(z) \), are divided and the result equated term-by-term to \( b(z) \). In general, these lag distributions have infinite extent. However, the terms are usually only significant for a few lags.

Examples of Fitting Lag Distributions

As an example of the use of this procedure, consider the following set of data from Maddala (1977, p.370). This quarterly data consist of capital expenditures (Y) and appropriations (X) from 1953 to 1967. The models fit with the PDLREG procedure have lag distributions ranging from lag 0 to lag 7. This lag distribution is modeled as a cubic polynomial and a quartic polynomial with no endpoint constraints. A correction is made for autocorrelation in the residuals by using a seasonal autoregressive model for the residuals.

A rational distributed lag model is fit to the same data using the ARIMA procedure. The lag distribution fit by PROC ARIMA can be written as

\[ b(z) = (0 - \theta_4 z^4) / (1 - \theta_1 z - \theta_2 z^2) \]

The resulting lag distributions are plotted in Figure 2. The cubic polynomial lag distribution and the rational lag distribution both have four free parameters, and the estimates of the lag distributions are quite similar.
The model assumed for the response time series, $y_t$, is

$$y_t = x_t' \beta + n_t$$

where $x_t$ is a vector of covariates and $n_t$ is an error term which may be autocorrelated.

Two kinds of predicted values are of interest for this model. The first kind of predicted value predicts $y_t$ without using the information in the preceding residual values. This predicted value is computed as

$$y_t^{(1)} = x_t' \beta$$

where $\beta$ is the estimate of $\beta$. The confidence limits associated with $y_t^{(1)}$ are computed from the estimated covariance matrix of $\beta$. The second kind of predicted value, which was available in previous versions of the AUTOREG procedure, also uses the preceding residuals to predict the error term in the model. This predicted value is computed as

$$y_t^{(2)} = x_t' \beta + n_t$$

where $n_t^*$ is the prediction of the error term based on the autoregressive model for $n_t$. The confidence limits for this second type of predicted value use the covariance of $\beta$ and the estimated autoregressive model for the residual to estimate the variance of $x_t' \beta$ and the variance of $n_t^*$, the prediction error for the residual. The second type of predicted value is generally used for forecasting future values from a given set of data.

Missing values are a common feature of time series data sets. The new version of AUTOREG can use time series with embedded missing values. Only observations with nonmissing values for the response and all covariates can be used to fit the model, but they do not have to be contiguous. As proposed by Jones (1980), a Kalman filter algorithm, which easily handles missing values, is used to do the autocorrelation corrections. Either the default Yule-Walker or maximum likelihood estimates can be used. However, with missing values, the Yule-Walker estimates are more likely to encounter difficulty staying within the stable region of the parameter space.

In addition to the new maximum likelihood estimates, exact least squares estimates are available. The exact least squares estimates find the values of the regression and autoregressive parameters which minimize the quadratic form

$$(y-X \beta)' V^{-1} (y-X \beta)$$

where $V$ is the covariance matrix of the error terms as a function of the autoregressive parameters, $\theta$.

The following example illustrates the above new features of AUTOREG. The forecasts, confidence limits and data are plotted in Figure 3.

```plaintext
TITLE 'GRUNFELD's INVESTMENT MODELS';
TITLE2 'FIT WITH AUTOREGRESSIVE ERRORS';
DATA GRUNFELD;
INPUT YEAR G1 G2 G3 W1 W2 W3;
```
--create some missing values;
IF RANN(123557>.8 THEN GEI=.;
LABEL GEI='GROSS INVESTMENT GE';
GEC='CAPITAL STOCK LAGGED GE';
GEF='VALUE OF OUTSTANDING SHARES GE LAGGED';

CARDS;

PROC AUTOREG;
MODEL GEl = GEF GEC (NLAG=2 METHOD=ML;
OUTPUT OUT=OUT P=P R=R ALPHACLI=.10 UCL=U LCL=L;

The SYSLIN Procedure

SYSLIN is a new procedure in the SAS/ETS library and is a descendant of the SYSREG procedure. The capabilities of SYSLIN are very similar to that of SYSREG, but the syntax used with SYSLIN is more compatible with that used by the nonlinear modeling procedures, SYSLIN and SIMNLIN. The exogenous and endogenous variables are specified in EXOGENOUS and ENDOGENOUS statements, and the equations that form the system are specified in individual MODEL statements. The kind of estimation desired is specified explicitly as an option in the procedure statement.

Some features of SYSLIN not available in SYSREG are:

- Three new estimation methods, general k-class, MELO (minimum expected loss) and iterated three stage least squares
- Reduced form estimates
- Plots of residuals versus independent variables
- a test for overidentification

In the example below Klein's model I is estimated with PROC SYSLIN using the MELO estimation option. The estimates are also put into reduced form. Plots of the residuals versus the independent variables are requested for the consumption equation with the PLOT option.

----; TITLE 'KLEIN'S MODEL I'; ----
DATA KLEIN;
  INPUT YEAR C P W I X WP G T K WSUM;
  DATE = MDY(I,YEAR);
  FORMAT DATE MONYY.;
  Y = C+I+G-T;
  YR = YEAR-1931;
  KLAG = LAG(K);
  PLAG = LAG(P);
  XLAG = LAG(X);

Some features of SYSLIN not available in SYSREG are:

- Three new estimation methods, general k-class, MELO (minimum expected loss) and iterated three stage least squares
- Reduced form estimates
- Plots of residuals versus independent variables
- a test for overidentification

The SYSLIN Procedure

SYSLIN is a new procedure in the SAS/ETS library and is a descendant of the SYSREG procedure. The capabilities of SYSLIN are very similar to that of SYSREG, but the syntax used with SYSLIN is more compatible with that used by the nonlinear modeling procedures, SYSLIN and SIMNLIN. The exogenous and endogenous variables are specified in EXOGENOUS and ENDOGENOUS statements, and the equations that form the system are specified in individual MODEL statements. The kind of estimation desired is specified explicitly as an option in the procedure statement.

Some features of SYSLIN not available in SYSREG are:

- Three new estimation methods, general k-class, MELO (minimum expected loss) and iterated three stage least squares
- Reduced form estimates
- Plots of residuals versus independent variables
- a test for overidentification

In the example below Klein's model I is estimated with PROC SYSLIN using the MELO estimation option. The estimates are also put into reduced form. Plots of the residuals versus the independent variables are requested for the consumption equation with the PLOT option.

---------; TITLE 'KLEIN'S MODEL I'; ---------
DATA KLEIN;
  INPUT YEAR C P W I X WP G T K WSUM;
  DATE = MDY(I,YEAR);
  FORMAT DATE MONYY.;
  Y = C+I+G-T;
  YR = YEAR-1931;
  KLAG = LAG(K);
  PLAG = LAG(P);
  XLAG = LAG(X);

PROC SYSLIN MELO DATA=KLEIN OUTEST=C REDUCED;
ENDOGENOUS C P W I X WP G T Y;
INSTRUMENTS KLAG PLAG XLAG WP G T YR;
PRODUCT, INCOME, PROFIT, STOCK, WAGE, PRoe PRINT;
MODEL C = P PLAG WSUM/PLOT;
I = P PLAG KLAG;
W = X XLAG YR;
IDENTITY X = Y - T - WP;
IDENTITY Y = C + I + G - T;
IDENTITY P = Y - W - WP;
IDENTITY K = KLAG + I;
IDENTITY WSUM = W + WP;
PROC PRINT;

ENHANCEMENTS TO THE SYSTEM MODELING PROCEDURES

Merging Model Files with PROCS MODEL, SYSLIN and SIMNLIN

All three of the nonlinear modeling procedures, SYSLIN, SIMNLIN, and MODEL, can now combine several model files. You can write and estimate large models by sectors and then combine them for simulation. For example,

*/-- Estimate the individual sectorial models ---*/
PROC SYSLIN OUTMODEL=SECTOR1 . . .
  . . model for sector 1 . . .
PROC SYSLIN OUTMODEL=SECTOR2 . . .
  . . model for sector 2 . . .
PROC SYSLIN OUTMODEL=SECTOR3 . . .
  . . model for sector 3 . . .
*/-- Combine sectors and simulate full model ---*/
PROC SIMNLIN
  MODEL=(SECTOR1 SECTOR2 SECTOR3) . . .
The MODEL=(list) option can be used on all three procedures to merge model files. The model files are combined in the order listed, and any additional model statements are added at the end.

PROC MODEL also has a new INCLUDE statement that allows you precise control in combining model files. The INCLUDE statement allows you to insert a model file into another model program. For example,

```sas
PROC MODEL OUTMODEL=COMBINED;
   . . some model statements . .
   INCLUDE MODEL=MODFILE1;
   . . . some more model statements . .
   INCLUDE MODEL=( MODFILE3 MODFILE2 );
   . . . some more model statements . .
PROC SIMNLIN MODEL=COMBINED . . .
```

Source Listing Feature Added

PROC MODEL, SYSNLIN, and SIMNLIN now have a LIST option to print the model program. The listing contains the original source statements. The names, values, and t-statistics for the parameters appearing in each statement are printed after the statement.

Using SAS Library Functions

SAS/ETS models can now use SAS library functions, provided that they are not used in a way that requires differentiation. For example, the LGAMMA function is not supported by the SAS/ETS nonlinear modeling system, but it is in the SAS function library. Therefore you can use LGAMMA in writing models, but the modeling system does not know how to differentiate it.

You can use a library function in a context that does require its derivative provided that you calculate the derivative yourself. You can then supply the derivative of the library function to SYSNLIN or SIMNLIN by using a programming trick involving the special ZERODER built-in function.

For example, the derivative of the LGAMMA function is the DIGAMMA function, which is also in the SAS function library. If you wanted to estimate the model

\[ y = a + \sqrt{\ln(1 + bx) + \tau} \]

you could write

```sas
PROC SYSNLIN DATA=D;
   ENDOGENOUS Y; EXOGENOUS X; PARMS A B 1;
   T = B * X;
   Z = ZERODER( T );
   R = LGAMMA( Z ) + DIGAMMA( Z ) * ( T - Z );
   Y = A + SQRT( R );
```

The special built-in function ZERODER tells SYSNLIN to assume a zero derivative for the argument. This allows you to use the library functions without causing the differentiator to complain. The coding trick is to add and subtract to the LGAMMA result the product of the DIGAMMA result and the argument B*X. One product uses the argument passed through the ZERODER function and the other does not.

Calling User Written Subroutines

You can use the same trick for user written functions. You can also use the CALL statement to call external subroutines. Any function or subroutine that can be called from a SAS DATA step can also be called from a model program executed by SYSNLIN or SIMNLIN. Either the external program can calculate derivatives or numerical derivatives can be computed.

If your interface program takes derivatives of the submodel, then you can use SYSNLIN to estimate parameters involving the submodel and you can use the default NEWTON method to solve the model with SIMNLIN. The SAS/ETS User's Guide, Version 5 Edition, contains a detailed explanation of how to link external programs to SYSNLIN and SIMNLIN with derivatives.

Monte Carlo Simulation with PROC SIMNLIN

Monte Carlo simulation with PROC SIMNLIN was the number one item in the SAS/ETS section of the 1984 SASware Ballot. PROC SIMNLIN now supports Monte Carlo simulation with the new RANDOM= option.

The RANDOM=n option causes SIMNLIN to repeat the solution n times. SIMNLIN can automatically generate pseudo-random perturbations of parameters and predicted values from multivariate normal distributions with covariance matrices read from ESTDATA= or SDATA= data sets. Alternatively, you can use the SAS random number generator functions in the model program to generate random shocks to the model from any distribution in any way that you like.

PROC SYSNLIN can store estimates of the covariance of the equation errors and of the covariance of the parameter estimates in output data sets. These data sets can then be used by
SIMNLIN to generate Monte Carlo forecasts. These random forecasts can then be averaged with PROC SUMMARY to produce mean forecasts. You can also use PROC SUMMARY to compute the variance of the simulated forecasts at each forecast period and then use these sample variances to compute “empirical” forecast confidence limits. For example,

```
PROC SYSNLIN DATA=HISTORY OUTMODEL=M OUTFTEST=EST COVOUT OUT=EST;
ENDOGENOUS Y1 Y2;...
Y1 = ...; Y2 = ...;
RANGE DATE;
PROC SIMNLIN DATA=ASSUME MODEL=M OUT=REPS ESTDATA=EST SDATA=S RANDOM=100;
RANGE DATE;
PROC SUMMARY DATA=REPS;
VAR Y1 Y2;
GLASS DATE;
OUTPUT OUT=FOREGAST MEAN=;
PROC PRINT DATA=FOREGAST; ID DATE;
TITLE 'Average of Random Forecast Simulations';
```

Vector ARMA Models with SYSNLIN and SIMNLIN

SYSNLIN and SIMNLIN can estimate and simulate models with vector ARMA error processes. In this sort of model, the error term of the regression model is not independent across time, but instead is generated by an autoregressive moving average process.

In SYSNLIN and SIMNLIN these models are handled by explicitly including a function of the lagged residuals in the structural equation. The system does not support vectors and matrices, so you have to write out the vector model with scalar expressions. This can be inconvenient for large models, but it has the advantage of providing precise control of structural constraints on the parameter matrix.

For example, suppose a two vector Y is a function of a covariate two vector X and that the error term is generated by a vector ARMA(1,1) model with innovations E. The model is

\[
U_t = \phi U_{t-1} + E_t + \theta E_{t-1}
\]

\[
Y_t = \gamma X_t + U_t
\]

We can write this in SYSNLIN as:

```
PARMS A1 A2 P_1_1 P_1_2 P_2_1 P_2_2
X_1_1 X_1_2 X_2_1 X_2_2
T_1_1 T_1_2 T_2_1 T_2_2;

Y1AT = A1 + X_1_1 * X1 + X_1_2 * X2;
Y2AT = A2 + X_2_1 * X1 + X_2_2 * X2;
U1 = P_1_1 * LAG( Y1-Y1AT )
   + P_1_2 * LAG( Y2-Y2AT )
   + T_1_1 * ZLAG( RESID.Y1 )
   + T_1_2 * ZLAG( RESID.Y2 );
U2 = P_2_1 * LAG( Y1-Y1AT )
   + P_2_2 * LAG( Y2-Y2AT )
   + T_2_1 * ZLAG( RESID.Y1 )
   + T_2_2 * ZLAG( RESID.Y2 );
Y1 = Y1AT + U1;
Y2 = Y2AT + U2;
```

To write such a model, compute the structural part first, Y1AT and Y2AT above, then use the differences, Y1-Y1AT and Y2-Y2AT, as the noise process to be modeled. Notice that the ZLAG function must be used for the moving average lags. If the LAG function were used, the program would generate only missing value estimates for U1 and U2. In such a case the SYSNLIN procedure prints an error message warning of an infinite recurrence.

The %AR Macro

Because autoregressive models are often used to deal with autocorrelation, a SAS macro, %AR, has been developed to generate programming statements for autoregressive models with PROC SYSNLIN and PROC SIMNLIN. The %AR macro can be used both for univariate and vector autoregressive models. The autoregressive process can be applied to the structural equation errors or to the endogenous series themselves.

The %AR macro can be used in three different ways:

- Univariate autoregression
- Unrestricted vector autoregression
- Restricted vector autoregression

The %AR macro is distributed with the SAS/EST product and is included in the "SAMPLE" library (member name "AR").

Univariate Autoregression To model the error term of an equation as an autoregressive process use %AR(varname, nlags) after the equation. For
example, suppose that $Y$ is a linear function of $X_1$ and $X_2$, and an $AR(2)$ random error. You would write:

```plaintext
PROC SYSLIN DATA=D;
ENDOGENOUS Y; EXOGENOUS X1 X2;
PARMS A B C;
Y = A + B*X1 + C*X2;
%AR( Y, 2 );
```

You can also restrict the autoregressive parameters to zero at selected lags. For example, if you wanted autoregressive parameters at lags one, twelve, and thirteen, you could use:

```plaintext
PROC SYSLIN DATA=D;
ENDOGENOUS Y; EXOGENOUS X1 X2;
PARMS A B C;
Y = A + B*X1 + C*X2;
%AR( Y, 13, 11213 );
```

By default, %AR uses the conditional least squares (CLS) method for the autoregressive terms and includes the first nlags start-up residuals in the objective function. By using the M=method option, you can request that %AR use the unconditional least squares (ULS), maximum likelihood (ML), or not include the first $k$ start-up residuals in the objective function (CLS$k$). For example:

```plaintext
Y = A + B*X1 + C*X2;
%AR( Y, 2, M=ULS );
```

You use %AR to apply the autoregressive model to the endogenous variable by using the TYPE=V option. For example, if you want to add 2 past lags of $Y$ to the equation in the example above, you could use %AR to generate the parameters and lags.

```plaintext
PROC SYSLIN DATA=D;
ENDOGENOUS Y; EXOGENOUS X1 X2;
PARMS A B C;
Y = A + B*X1 + C*X2;
%AR( Y, 2, TYPE=V );
```

This model predicts $Y$ as a linear combination of $X_1$, $X_2$, an intercept, and the values of $Y$ in the most recent two previous periods. Because it is the endogenous variables that are lagged, the independent variables have a distributed lag effect on the response.

Unrestricted Vector Autoregression To model the error terms of a set of equations as a vector autoregressive process with no zero restrictions use %AR(process-name,nlags,variable-list) after the equations. The process-name is any name that you supply for %AR to use in making names for the autoregressive parameters. The variable-list is the list of endogenous variables for the equations.

For example, suppose that the errors for $Y_1$, $Y_2$, and $Y_3$ are generated by a second order vector autoregressive process. You could write:

```plaintext
PROC SYSLIN DATA=D;
ENDOGENOUS Y1 Y2 Y3;
Y1 = . . . /*structural equation for Y1--*/
Y2 = . . . /*structural equation for Y2--*/
Y3 = . . . /*structural equation for Y3--*/
%AR( ANYNAME, 2, Y1 Y2 Y3 );
```

Only the conditional least squares method can be used for vector processes. You can also use this same form with restrictions that the coefficient matrix be zero at selected lags. For example,

```plaintext
%AR( ANYNAME, 5, Y1 Y2 Y3, 13 );
```

applies a third order vector process to the equation errors with all the coefficients at lag two restricted to zero, and with the coefficients at lags one and three unrestricted. (These restrictions are applied by simply omitting the terms from the model.)

You could model the three series $Y_1$-$Y_3$ as a pure vector autoregressive process. If there are no exogenous components to the vector autoregression model, including no intercepts, then assign zero to each of the variables. (There must be an assignment to each of the variables before %AR is called.) For example,

```plaintext
PROC SYSLIN DATA=D;
ENDOGENOUS Y1 Y2 Y3;
Y1 = 0; Y2 = 0; Y3 = 0;
%AR( ANYNAME, 2, Y1 Y2 Y3 );
```

This example models the vector $Y=(Y_1 Y_2 Y_3)$ as a linear function only of its value in the previous two periods and an independent white noise error vector. The number of parameters grows rapidly with the number of endogenous variables. This model has 18 ($=3*3+3*3$) parameters.
Restricted Vector Autoregression If you wish, you can control which parameters are included in the process. (That is, you can restrict those parameters that you do not include to be zero.) First use %AR with the DEFER option to declare the variable list and define the dimension of the process. Then use additional %AR calls to generate terms for selected equations with selected variables at selected lags.

For example:

```sas
PROC SYSLIN DATA=DATA;
ENDOGENOUS Y1 Y2 Y3;
Y1 = . . . . /"--structural equation for Y1--"/
Y2 = . . . . /"--structural equation for Y2--"/
Y3 = . . . . /"--structural equation for Y3--"/
%AR( ANYNAME, 2, Y1 Y2 Y3, DEFER );
%AR( ANYNAME, Y1 Y1 Y2 );
%AR( ANYNAME, Y2 Y3 , 1 );
```

This model says that the errors for Y1 depend on the errors for both Y1 and Y2 (but not Y3) at both lags 1 and 2, and that the errors for Y2 and Y3 depend on the previous errors for all three variables, but only at lag one.

Easy Access to Data Resources Inc. Data Banks

The SAS/ETS Software product has several aids for reading data distributed by data base vendors. The previous release of SAS/ETS Software contains PROC CITIBASE for reading Citibase data tapes, and a macro for reading Compustat files.

The new release includes a new macro for reading data tape containing data extracted from Data Resources Inc. data banks. If you are a DRI user, you can use DRI's DRIDATA/Delivery system to have data extracted from DRI's data banks and delivered to your site. DRI has provided a utility that automatically translates the DRI series names to valid SAS variable names so that the extracted data can be easily read with SAS.

You can read data from a DRIDATA/Delivery file into a SAS data set with the %READDRI macro supplied with SAS/ETS Software. You allocate a file to the DRI tape, decide on a name for the SAS data set, and invoke the %READDRI macro:

```sas
%READDRI( TAPEFILE, DATASET, DESC );
```


REFERENCES


Comparison of Polynomial and Rational Lag Distributions for Industrial Conference Board Data
Missing Values and Confidence Limits

Figure 3.