THE POWER TO KNOW.

SAS/STAT® 9.3 User's Guide

The CALIS Procedure

(Chapter)

SAS® Documentation

This document is an individual chapter from *SAS/STAT® 9.3 User's Guide*.

The correct bibliographic citation for the complete manual is as follows: SAS Institute Inc. 2011. *SAS/STAT® 9.3 User's Guide*. Cary, NC: SAS Institute Inc.

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, July 2011

Chapter 26

The CALIS Procedure

Contents

Overview: CALIS Procedure

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

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

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

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

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

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

- PATH—provides an intuitive causal path specification interface
- RAM—utilizes the formulation of the reticular action model (McArdle and McDonald 1984) to define models
- REFMODEL—provides a quick way for model referencing and respecification

Various modeling languages are provided to suit a wide range of researchers' background and modeling philosophy. However, statistical situations might arise where one modeling language is more convenient than the others. This will be discussed in the section "Which Modeling Language?" on page 1012.

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

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

- covariance and mean matrices and their properties
- descriptive statistics like means, standard deviations, univariate skewness, and kurtosis measures
- multivariate measures of kurtosis
- coverage of covariances and means, missing patterns summary, and means of the missing patterns when the FIML estimation is used
- weight matrix and its descriptive properties

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

- computing squared multiple correlations and determination coefficients
- direct and indirect effects partitioning with standard error estimates
- model modification tests such as Lagrange multiplier and Wald tests
- computing fit summary indices
- computing predicted moments of the model
- residual analysis
- factor rotations
- standardized solutions with standard errors
- testing parametric functions, individually or simultaneously

When fitting a model, you need to choose an estimation method. The following estimation methods are supported in the CALIS procedure:

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

Estimation methods implemented in PROC CALIS do not exhaust all alternatives in the field. For example, the partial least squares (PLS) method is not implemented. See the section "Estimation Criteria" on page 1246 for details about estimation criteria used in PROC CALIS. Note that there is a SAS/STAT procedure called PROC PLS, which employs the partial least squares technique but for a different class of models than those of PROC CALIS. For general path analysis with latent variables, consider using PROC CALIS.

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

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

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

With initial estimates, PROC CALIS will iterate the solutions so as to achieve the optimum solution as defined by the estimation criterion. This is a process known as optimization. Because numerical problems can occur in any optimization process, the CALIS procedure offers several optimization algorithms so that you can choose alternative algorithms when the one being used fails. The following optimization algorithms are supported in PROC CALIS:

- Levenberg-Marquardt algorithm (Moré 1978)
- trust-region algorithm (Gay 1983)
- Newton-Raphson algorithm with line search
- ridge-stabilized Newton-Raphson algorithm
- various quasi-Newton and dual quasi-Newton algorithms: Broyden-Fletcher-Goldfarb-Shanno and Davidon-Fletcher-Powell, including a sequential quadratic programming algorithm for processing nonlinear equality and inequality constraints
- various conjugate gradient algorithms: automatic restart algorithm of Powell (1977), Fletcher-Reeves, Polak-Ribiere, and conjugate descent algorithm of Fletcher (1980)

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

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

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

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

The CALIS procedure uses ODS Graphics to create graphs as part of its output. High-quality residual histograms are available in PROC CALIS. See Chapter 21, "Statistical Graphics Using ODS," for general information about ODS Graphics. See the section "ODS Graphics" on page 1314 and the PLOTS= option on page 1047 for specific information about the statistical graphics available with the CALIS procedure.

Changes and Enhancements

The following sections describe the new features of this version of PROC CALIS.

Built-In Covariance and Mean Structures

PROC CALIS now supports the fitting of some standard covariance and mean patterns by using the COVPATTERN= and the MEANPATTERN= options. These standard covariance and mean patterns are built into PROC CALIS. You can call these built-in patterns by appropriate keywords without using explicit model specifications such as the MSTRUCT and MATRIX statements. For example, you can now test the compound symmetry pattern of a covariance matrix by simply specifying the COVPATTERN=COMPSYM option. PROC CALIS then generates the compound symmetry pattern internally for model fitting. To specify the same covariance pattern in the previous version of PROC CALIS, you would need to use the MSTRUCT statement and specify the parameters of the covariance pattern in the MATRIX statement. Another example is using the COVPATTERN=EQCOVMAT option to test the equality of covariance matrices among independent groups. See the COVPATTERN= and MEANPATTERN= options for details about the supported covariance and mean patterns.

Covariance and Mean Structure Analysis with the COSAN Model

PROC CALIS now supports covariance and mean structure analysis in the COSAN model. You can specify the central mean vector in each term of the mean structure formula. See the COSAN statement and the section "The COSAN Model" on page 1193 for details.

Extended PATH Modeling Language

You can specify variances, covariances, means, and intercepts as paths in the PATH statement. The syntax enables you to map all the parameters in the path diagram to the PATH statement specification. See the PATH statement for details. Even if you specify variances, covariances, means, or intercepts in the PVAR, PCOV, and MEAN statements (but not in the path statement), you can still display these parameter estimates as paths in the output table for the regular path effect (coefficient) estimates by using the EXTENDPATH option.

Full Information Maximum Likelihood Method

PROC CALIS implements the full information maximum likelihood method (FIML) for treating data with random missing values. The FIML method uses all the available information from the data set, including observations with missing values, so that it is statistically more efficient than the ML (maximum likelihood) method (as implemented in PROC CALIS). You can use METHOD=FIML to invoke the FIML method. In addition to the estimation, the FIML method also provides detailed analysis of the missing patterns such as the coverage statistics of the sample moments, frequencies and proportions of the missing patterns, and the descriptive statistics of the missing patterns. You can use new options MAXMISSPAT=, NOMISSPAT, and TMISSPAT= to control the output of missing patterns analysis.

Improved RAM Model Specification

You can now specify the variable list explicitly in the VAR= option of the RAM statement. This variable list is useful to make immediate references of the variables (manifest or latent) in the model. The mean structure specification of the RAM model is also supported. See the RAM statement and the section "The RAM Model" on page 1229 for details.

Unnamed Free Parameter Specification

You can specify free parameters in all models without using explicit parameter names (that is, unnamed free parameters). This makes your model specification more efficient. For example, in the PATH statement, you can specify only the paths without using the parameter names for the path effects (coefficients). PROC CALIS generates the parameter names automatically. However, you can also input the parameter names whenever it is necessary (for example, for setting parameter constraints). Unnamed free parameters specification is supported in all modeling languages. For details, see the syntax of the following statements: COV, FACTOR, LINEQS, MATRIX, MEAN, PATH, PCOV, PVAR, RAM, and VARIANCE.

Structural Equation Modeling Application

The Structural Equation Modeling Application is a graphical user interface to structural equation modeling techniques. You can specify models in graphical form to represent the hypothesized relationships among the variables. It is accessed from JMP software and uses the CALIS procedure for its computations.

The application enables you to define model variables in a path diagram by dragging data set variables to the diagram and to define the relationship between the model variables by using an arrow tool. You can move the variables to arrange the path diagram exactly the way you want. You can easily make a copy of a model, modify it, and analyze the new model, and you can compare several models with appropriate fit statistics. Finally, you can save the model specifications and the results for later use.

The Structural Equation Modeling Application provides access to a subset of the capabilities in the CALIS procedure. It supports mainly the PATH model specification through the path diagram interface. It does not support many other advanced features in PROC CALIS. For example, multiple-group analysis and full information maximum likelihood estimation are not available in the Structural Equation Modeling Application. For details, see *The Structural Equation Modeling Application*.

A Guide to the PROC CALIS Documentation

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

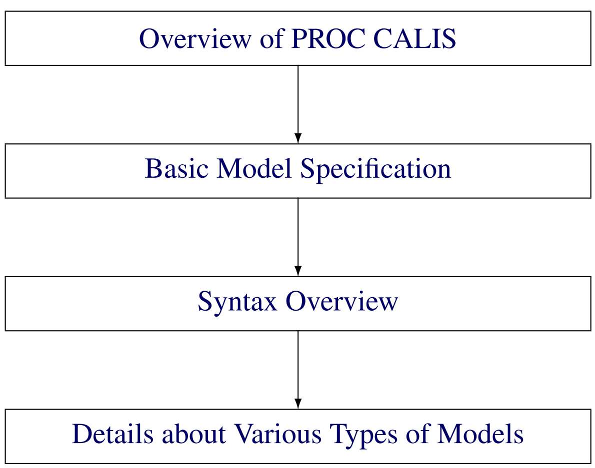
Reading this chapter sequentially is not a good strategy for learning about PROC CALIS. This section provides a guide or "road map" to the rest of the PROC CALIS chapter, starting with the basics and continuing through more advanced topics. Many sections assume that you already have a basic understanding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

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

Level	Milestone	Starting Section
Basic	You are able to specify simple models, but might make mistakes.	"Guide to the Basic Skill Level" on page 992
Intermediate	You are able to specify more sophisticated models with few syntactic and semantic mistakes.	"Guide to the Intermediate Skill Level" on page 997
Advanced	You are able to use the advanced options provided by PROC CALIS.	"Guide to the Advanced Skill Level" on page 999

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

Guide to the Basic Skill Level



Overview of PROC CALIS

The section "Overview: CALIS Procedure" on page 986 gives you an overall picture of the CALIS procedure but without the details.

Basic Model Specification

The structural equation example in the section "Getting Started: CALIS Procedure" on page 1002 provides the starting point to learn the basic model specification. You learn how to represent your theory by using a path diagram and then translate the diagram into the PATH model for PROC CALIS to analyze. Because the PATH modeling language is new, this example is useful whether or not you have previous experience with PROC CALIS. The PATH model is specified in the section "PATH Model" on page 1004. The corresponding results are shown and discussed in Example 26.16.

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

- You can continue to learn more modeling languages in the section "Getting Started: CALIS Procedure" on page 1002.
- You can skip to the section "Syntax Overview" on page 996 for an overview of the PROC CALIS syntax and learn other modeling languages at a later time.

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

- LINEQS: Section "LINEQS Model" on page 1006
- RAM: Section "RAM Model" on page 1005
- LISMOD: Section "LISMOD Model" on page 1008
- FACTOR: Section "A Factor Model Example" on page 1009
- MSTRUCT: Section "Direct Covariance Structures Analysis" on page 1011

After studying the examples in the "Getting Started: CALIS Procedure" section, you can strengthen your understanding of the various modeling languages by studying more examples such as those in section "Examples: CALIS Procedure" on page 1315. Unlike the examples in the "Getting Started: CALIS Procedure" section, the examples in the "Examples: CALIS Procedure" section include the analysis results in addition to the explanations of the model specifications.

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

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

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

- "Example 26.1: Estimating Covariances and Correlations" on page 1315 shows how you can estimate the covariances and correlations with standard error estimates for the variables in your model. The model you fit is a saturated covariance structure model.
- "Example 26.2: Estimating Covariances and Means Simultaneously" on page 1320 extends Example 26.1 to include the mean structures in the model. The model you fit is a saturated mean and covariance structure model.
- "Example 26.3: Testing Uncorrelatedness of Variables" on page 1322 shows a very basic covariance structure model, in which the covariance structures can be specified directly. The variables in this model are uncorrelated. You learn how to specify the covariance pattern directly.
- "Example 26.4: Testing Covariance Patterns" on page 1325 extends Example 26.3 to include other covariance structures that you can specify directly.
- "Example 26.5: Testing Some Standard Covariance Pattern Hypotheses" on page 1327 illustrates the use of built-in covariance patterns supported by PROC CALIS.

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

- "Example 26.6: Linear Regression Model" on page 1331 shows how you can fit a linear regression model with the PATH modeling language of PROC CALIS. This example also introduces the path diagram representation of "causal" models. You compare results obtained from PROC CALIS and from the REG procedure, which is designed specifically for regression analysis.
- "Example 26.7: Multivariate Regression Models" on page 1336 extends Example 26.6 in several different ways. You fit covariance structure models with more than one predictor, with direct and indirect effects. This example also discusses how you can choose the "best" model for your data.
- "Example 26.8: Measurement Error Models" on page 1354 explores the case where the predictor in simple linear regression is measured with error. The concept of latent true score variable is introduced. You use PROC CALIS to fit a simple measurement error model.
- "Example 26.9: Testing Specific Measurement Error Models" on page 1361 extends Example 26.8 to test special measurement error models with constraints. By using PROC CALIS, you can constrain your measurement error models in many different ways. For example, you can constrain the error variances or the intercepts to test specific hypotheses.
- "Example 26.10: Measurement Error Models with Multiple Predictors" on page 1367 extends Example 26.8 to include more predictors in the measurement error models. The measurement errors in the predictors can be correlated in the model.

More elaborate examples about the MSTRUCT and PATH models are listed as follows:

- "Example 26.16: Path Analysis: Stability of Alienation" on page 1415 shows you how to specify a simple PATH model and interpret the basic estimation results. The results are shown in considerable detail. The output and analyses include: a model summary, an initial model specification, an initial estimation method, an optimization history and results, residual analyses, residual graphics, estimation results, squared multiple correlations, and standardized results.

- "Example 26.18: Fitting Direct Covariance Structures" on page 1437 shows you how to fit your covariance structures directly on the covariance matrix by using the MSTRUCT modeling language. You also learn how to use the FITINDEX statement to create a customized model fit summary and how to save the fit summary statistics into an external file.
- "Example 26.20: Testing Equality of Two Covariance Matrices Using a Multiple-Group Analysis" on page 1453 uses the MSTRUCT modeling language to illustrate a simple multiple-group analysis. You also learn how to use the ODS SELECT statement to customize your printed output.
- "Example 26.21: Testing Equality of Covariance and Mean Matrices between Independent Groups" on page 1458 uses the COVPATTERN= and MEANPATTERN= options to show some tests of equality of covariance and mean matrices between independent groups. It also illustrates how you can improve your model fit by the exploratory use of the Lagrange multiplier statistics for releasing equality constraints.
- "Example 26.23: Testing Competing Path Models for the Career Aspiration Data" on page 1492 illustrates how you can fit competing models by using the OUTMODEL= and INMODEL= data sets for transferring and modifying model information from one analysis to another. This example also demonstrates how you can choose the best model among several competing models for the same data.

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

- "Example 26.11: Measurement Error Models Specified As Linear Equations" on page 1372 explores another way to specify measurement error models in PROC CALIS. The LINEQS modeling language is introduced. You learn how to specify linear equations of the measurement error model by using the LINEQS statement. Unlike the PATH modeling language, in the LINEQS modeling language, you need to specify the error terms explicitly in the model specification.
- "Example 26.12: Confirmatory Factor Models" on page 1378 introduces a basic confirmatory factor model for test items. You use the FACTOR modeling language to specify the factor-variable relationships.
- "Example 26.13: Confirmatory Factor Models: Some Variations" on page 1389 extends Example 26.12 to include some variants of the confirmatory factor model. With the flexibility of the FACTOR modeling language, this example shows how you fit models with parallel items, tau-equivalent items, or partially parallel items.

More advanced examples that use the LINEQS and FACTOR modeling languages are listed as follows:

- "Example 26.14: The Full Information Maximum Likelihood Method" on page 1399 shows how you can use the full information maximum likelihood (FIML) method to estimate your model when your data contain missing values. It illustrates the analysis of the data coverage of the sample variances, covariances, and means and the analysis of missing patterns and the mean profile. It also shows that the full information maximum likelihood method makes the maximum use of the available information from the data, as compared with the default ML (maximum likelihood) methods.

- "Example 26.15: Comparing the ML and FIML Estimation" on page 1409 discusses the similarities and differences between the ML and FIML estimation methods as implemented in PROC CALIS. It uses an empirical example to show how ML and FIML obtain the same estimation results when the data do not contain missing values.
- "Example 26.17: Simultaneous Equations with Mean Structures and Reciprocal Paths" on page 1430 is an econometric example that shows you how to specify models using the LINEQS modeling language. This example also illustrates the specification of reciprocal effects, the simultaneous analysis of the mean and covariance structures, the setting of bounds for parameters, and the definitions of metaparameters by using the PARAMETERS statement and SAS programming statements. You also learn how to shorten your output results by using some global display options such as the PSHORT and NOSTAND options in the PROC CALIS statement.
- "Example 26.19: Confirmatory Factor Analysis: Cognitive Abilities" on page 1441 uses the FACTOR modeling language to illustrate confirmatory factor analysis. In addition, you use the MODIFICATION option in the PROC CALIS statement to compute LM test indices for model modifications.
- "Example 26.24: Fitting a Latent Growth Curve Model" on page 1507 is an advanced example that illustrates the use of structural equation modeling techniques for fitting latent growth curve models. You learn how to specify random intercepts and random slopes by using the LINEQS modeling language. In addition to the modeling of the covariance structures, you also learn how to specify the mean structure parameters.

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

- "Example 26.22: Illustrating Various General Modeling Languages" on page 1483 extends Example 26.16, 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 26.16 and get equivalent results. This example shows the connections between the general modeling languages supported in PROC CALIS. A good understanding of Example 26.16 is a prerequisite for this example.

Once you are familiar with various modeling languages, you might wonder which modeling language should be used in a given situation. The section "Which Modeling Language?" on page 1012 provides some guidelines and suggestions.

Syntax Overview

The section "Syntax: CALIS Procedure" on page 1014 shows the syntactic structure of PROC CALIS. However, reading the "Syntax: CALIS Procedure" section sequentially might not be a good strategy. The statements used in PROC CALIS are classified in the section "Classes of Statements in PROC CALIS" on page 1015. Understanding this section is a prerequisite for understanding single-group and multiple-group analyses in PROC CALIS. Syntax for single-group analyses is described in the section "Single-Group Analysis Syntax" on page 1018, and syntax for multiple-group analyses is described in the section "Multiple-Group Multiple-Model Analysis Syntax" on page 1019.

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

- "Data Set Options" on page 1020
- "Model and Estimation Options" on page 1021
- "Options for Fit Statistics" on page 1021
- "Options for Statistical Analysis" on page 1022
- "Global Display Options" on page 1022
- "Optimization Options" on page 1024

Details about Various Types of Models

Several subsections in the section "Details: CALIS Procedure" on page 1173 can help you gain a deeper understanding of the various types of modeling languages, as shown in the following table:

Language	Section
COSAN	"The COSAN Model" on page 1193
FACTOR	"The FACTOR Model" on page 1197
LINEQS	"The LINEQS Model" on page 1205
LISMOD	"The LISMOD Model and Submodels" on page 1212
MSTRUCT	"The MSTRUCT Model" on page 1220
PATH	"The PATH Model" on page 1223
RAM	"The RAM Model" on page 1229

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 Skill Level

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

- The section "Naming Variables and Parameters" on page 1238 summarizes the naming rules and conventions for variable and parameter names in specifying models.
- The section "Setting Constraints on Parameters" on page 1239 covers various techniques of constraining parameters in model specifications.
- The section "Automatic Variable Selection" on page 1245 discusses how PROC CALIS treats variables in the models and variables in the data sets. It also discusses situations where the VAR statement specification is deemed necessary.
- The section "Computational Problems" on page 1291 discusses computational problems that occur quite commonly in structural equation modeling. It also discusses some possible remedies of the computational problem.
- The section "Missing Values and the Analysis of Missing Patterns" on page 1279 describes the default treatment of missing values.
- The statements REFMODEL on page 1158 and RENAMEPARM on page 1160 are useful when you need to make references to well-defined models when specifying a "new" model. See Example 26.27 for an application.

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

You can also study the following more advanced examples:

- "Example 26.25: Higher-Order and Hierarchical Factor Models" on page 1513 is an advanced example for confirmatory factor analysis. It involves the specifications of higher-order and hierarchical factor models. Because higher-order factor models cannot be specified by the FACTOR modeling language, you need to use the LINEQS model specification instead. A second-order factor model and a bifactor model are fit. Linear constraints on parameters are illustrated by using the PARAMETERS statement and SAS programming statements. Relationships between the second-order factor model and the bifactor model are numerically illustrated.
- "Example 26.26: Linear Relations among Factor Loadings" on page 1529 is an advanced example of a first-order confirmatory factor analysis that uses the FACTOR modeling language. In this example, you learn how to use the PARAMETERS statement and SAS programming statements to set up dependent parameters in your model. You also learn how to specify the correlation structures for a specific confirmatory factor model.
- "Example 26.27: Multiple-Group Model for Purchasing Behavior" on page 1538 is a sophisticated example of analyzing a path model. The PATH modeling language is used. In this example, a two-group analysis of mean and covariance structures is conducted. You learn how to use the REFMODEL statement to reference properly defined models and the SIMTESTS statement to test a priori simultaneous hypotheses.
- "Example 26.28: Fitting the RAM and EQS Models by the COSAN Modeling Language" on page 1563 introduces the COSAN modeling language by connecting it with general RAM and EQS models. The model matrices of the RAM or EQS model are described. You specify these model matrices and the associated parameters in the COSAN modeling language.

- “Example 26.29: Second-Order Confirmatory Factor Analysis” on page 1596 constructs the covariance structure model of the second-order confirmatory factor model. You define the model matrices by using the COSAN modeling language.
- “Example 26.30: Linear Relations among Factor Loadings: COSAN Model Specification” on page 1604 shows how you can set linear constraints among model parameters under the COSAN model.
- “Example 26.31: Ordinal Relations among Factor Loadings” on page 1610 shows how you can set ordinal constraints among model parameters under the COSAN model.
- “Example 26.32: Longitudinal Factor Analysis” on page 1614 defines the covariance structures of a longitudinal factor model and shows how you can specify the covariance structure model with the COSAN modeling language.

Guide to the Advanced Skill Level

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

Advanced Data Analysis Tools

The following advanced data analysis topics are discussed:

- Assessment of fit

  The section “Assessment of Fit” on page 1260 presents the fit indices used in PROC CALIS. However, the more important topics covered in this section are about how model fit indices are organized and used, how residuals can be used to gauge the fitting of individual parts of the model, and how the coefficients of determination are defined for equations.

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

- Effect partitioning

  The section “Total, Direct, and Indirect Effects” on page 1273 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 1258 describes how PROC CALIS computes model fit degrees of freedom and how you can use some options on the PROC CALIS statement to make degrees-of-freedom adjustments.

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

- Standardized solutions

  Standardization schemes used in PROC CALIS are described and discussed in the section “Standardized Solutions” on page 1275.

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

- Model modifications

  In the section "Modification Indices" on page 1277, modification indices such as Lagrange multiplier test indices and Wald statistics are defined and discussed. These indices can be used either to enhance your model fit or to make your model more precise.

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

- A Priori Parametric Function Testing

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

Advanced Output Control Tools

To be more effective in presenting your analysis results, you need to be more sophisticated in controlling your output. Some customization tools have been discussed in the previous section "Advanced Data Analysis Tools" on page 999 and might have been mentioned in the examples included in the basic and the intermediate levels. In the following topics, these output control tools are presented in a more organized way so that you can have a systematic study scheme of these tools.

- Global output control tools in PROC CALIS

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

  See the ALL, PRINT, PSHORT, PSUMMARY, and NOPRINT options for ways to control the appearances of the output. See the section "Global Display Options" on page 1022 for details about the global display options and their relationships with the individual output display options. Also see the ORDERALL, ORDERGROUPS, ORDERMODELS, ORDERSPEC, PARMNAME, PRIMAT, NOORDERSPEC, NOPARMNAME, NOSTAND, and NOSE options which control the output formats.

- Customized analysis tools in PROC CALIS

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

  - The ON=, OFF=, ON(ONLY)= options in the FITINDEX statement enable you to select individual or groups of model fit indices or modeling information to display. You can still save the information of *all* fit indices in an external file by using the OUTFIT= option.
  - The EFFPART statement enables you to customize the effect analysis. You display only those effects of substantive interest.
  - The LMTESTS statement enables you to customize the sets of LM tests of interest. You test only those potential parameters that are theoretically and substantively possible.

- Output selection and destinations by the ODS system

  This kind of output control is used not only for PROC CALIS, but is used for all procedures that support the ODS system. The most common uses include output selection and output destinations assignment. You use the ODS SELECT statement together with the ODS table names or graph names to select particular output displays. See the section "ODS Table Names" on page 1298 for these names in PROC CALIS.

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

Reference Topics

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

- Measures of multivariate kurtosis and skewness

  This is covered in the section "Measures of Multivariate Kurtosis" on page 1279.

- Estimation criteria and the mathematical functions for estimation

  The section "Estimation Criteria" on page 1246 presents formulas for various estimation criteria. The relationships among these criteria are shown in the section "Relationships among Estimation Criteria" on page 1252. 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 1255, where you can also find information about the computation of the standard error estimates in PROC CALIS.

- Initial estimation

  Initial estimates are necessary for all kinds of iterative optimization techniques. They are described in section "Initial Estimates" on page 1282.

- Use of optimization techniques

  Optimization techniques are covered in section "Use of Optimization Techniques" on page 1283. See this section if you need to fine-tune the optimization.

- Output displays and control

  The output displays in PROC CALIS are listed in the section "Displayed Output" on page 1294. 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 1298 for the ODS table and graph names.

- Input and output files

  PROC CALIS supports several input and output data files for data, model information, weight matrices, estimates, fit indices, and estimation and descriptive statistics. The uses and the structures of these input and output data files are described in the sections "Input Data Sets" on page 1173 and "Output Data Sets" on page 1176.

Getting Started: CALIS Procedure

A Structural Equation Example

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

Anomie67:	Anomie 1967
Powerless67:	Powerlessness 1967
Anomie71:	Anomie 1971
Powerless71:	Powerlessness 1971
Education:	Education level (years of schooling)
SEI:	Duncan's socioeconomic index (SEI)

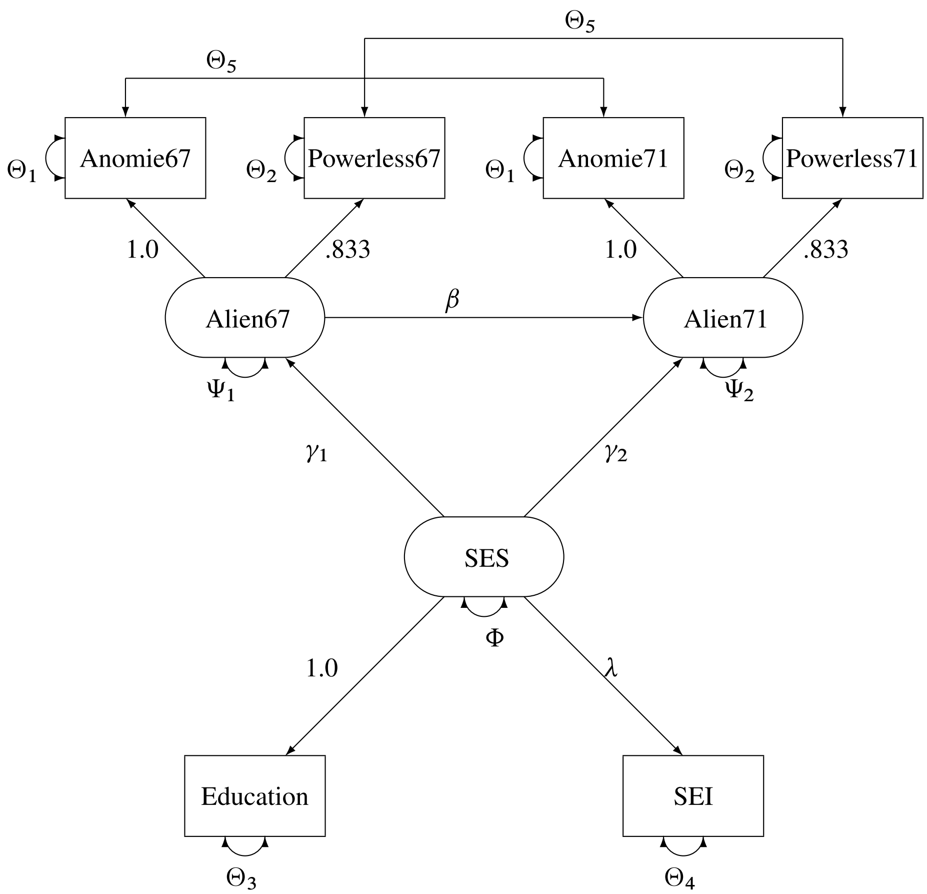
The covariance matrix of these six variables is stored in the data set named Wheaton.

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

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

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

Figure 26.1 Path Diagram of Stability and Alienation Example



In the path diagram shown in Figure 26.1, regressions of variables are represented by one-headed arrows. Regression coefficients are indicated along these one-headed arrows. Variances and covariances among the variables are represented by two-headed arrows. Error variances and covariances are also represented by two-headed arrows. This scheme of representing paths, variances and covariances, and error variances and covariances (McArdle 1988; McDonald 1985) is helpful in translating the path diagram to the PATH or RAM model input in the CALIS procedure.

PATH Model

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

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

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

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

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

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

See Example 26.16 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 1223 and the PATH statement on page 1137.

RAM Model

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

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

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

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

See Example 26.22 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 1229 and the RAM statement on page 1151.

LINEQS Model

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

```
proc calis nobs=932 data=Wheaton;
   lineqs
      Anomie67     = 1.0    * f_Alien67 + E1,
      Powerless67  = 0.833  * f_Alien67 + E2,
      Anomie71     = 1.0    * f_Alien71 + E3,
      Powerless71  = 0.833  * f_Alien71 + E4,
      Education    = 1.0    * f_SES     + E5,
      SEI          = lambda * f_SES     + E6,
      f_Alien67    = gamma1 * f_SES     + D1,
      f_Alien71    = gamma2 * f_SES     + beta * 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 VARIANCE statement, and covariance parameters among exogenous variables, including errors and disturbances, are specified in COV statement.

See Example 26.22 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 1205 and the LINEQS statement on page 1090.

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 calis nobs=932 data=Wheaton;
   lismod
      yvar = Anomie67 Powerless67 Anomie71 Powerless71,
      xvar = Education SEI,
      etav = Alien67 Alien71,
      xiv  = SES;
   matrix _LAMBDAY_ [1,1] = 1.0,
                    [2,1] = 0.833,
                    [3,2] = 1.0,
                    [4,2] = 0.833;
   matrix _LAMBDAX_ [1,1] = 1.0,
                    [2,1] = lambda;
   matrix _GAMMA_   [1,1] = gamma1,
                    [2,1] = gamma2;
   matrix _BETA_    [2,1] = beta;
   matrix _THETAY_  [1,1] = theta1,
                    [2,2] = theta2,
                    [3,3] = theta1,
                    [4,4] = theta2,
                    [3,1] = theta5,
                    [4,2] = theta5;
   matrix _THETAX_  [1,1] = theta3,
                    [2,2] = theta4;
   matrix _PSI_     [1,1] = psi1,
                    [2,2] = psi2;
   matrix _PHI_     [1,1] = phi;
run;
```

In the LISMOD statement, you specify the lists of variables in the model. In the MATRIX statements, you specify the parameters in the LISMOD model matrices. Each MATRIX statement contains the matrix name of interest and then locations of the parameters, followed by the parameter names or fixed parameter values. It would be difficult to explain the LISMOD specification here without better knowledge about the formulation of the mathematical model. For this purpose, see the section "The LISMOD Model and Submodels" on page 1212 and the LISMOD statement on page 1097. See also Example 26.22 for partial results of the current LISMOD model analysis.

COSAN Model

The COSAN model specification is even more abstract than all of the modeling languages considered. Like the LISMOD model specification, to specify a COSAN model you need to define parameters as entries in model matrices. In addition, you must also provide the definitions of the model matrices and the matrix formula for the covariance structures in the COSAN model specification. Therefore, the COSAN model specification requires sophisticated knowledge about the formulation of the mathematical model. For this reason, the COSAN model specification of the preceding path model is not discussed here (but see Example 26.28). For more details about the COSAN model specification, see the section "The COSAN Model" on page 1193 and the COSAN statement on page 1055.

A Factor Model Example

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

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

```
proc calis nobs=932 data=Wheaton;
   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 CALIS can also be done equivalently by the FACTOR procedure, as shown in the following statements for the Wheaton et al. (1977) data:

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

Note that METHOD=ML is necessary because maximum likelihood is not the default method in PROC FACTOR.

Whereas you can use either the CALIS or FACTOR procedure to fit certain exploratory factor models, you can only use the CALIS procedure to fit confirmatory factor models. In a confirmatory factor model, you are assumed to have some prior knowledge about the variable-factor relations. For example, in your substantive theory, some observed variables are not related to certain factors in the model. The following statements illustrate the specification of a confirmatory factor model for Wheaton et al. (1977) data:

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

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

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

For more information about exploratory and confirmatory factor models and the FACTOR modeling language, see the section "The FACTOR Model" on page 1197 or the FACTOR statement on page 1072.

Direct Covariance Structures Analysis

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

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

The covariance structures are hypothesized as follows:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \phi_1 & \theta_1 & \theta_2 & \theta_1 \\ \theta_1 & \phi_2 & \theta_1 & \theta_3 \\ \theta_2 & \theta_1 & \phi_1 & \theta_1 \\ \theta_1 & \theta_3 & \theta_1 & \phi_2 \end{pmatrix}$$

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

In the MSTRUCT statement, you specify the list of variables of interest with the VAR= option. The order of the variables in the list will be the order in the hypothesized covariance matrix. Next, you use the MATRIX _COV_ statement to specify the parameters in the covariance matrix. The specification is a direct translation from the hypothesized covariance matrix. For example, the `[1,1]` element of the covariance matrix is fitted by the free parameter `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 26.18.

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

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

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

For more information about the MSTRUCT modeling language, see the section "The MSTRUCT Model" on page 1220 and the MSTRUCT statement on page 1130.

Which Modeling Language?

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

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

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

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

In PROC CALIS, the FACTOR and MSTRUCT modeling languages are considered more specialized, while other languages are more general in applications. Whenever possible, you should use the more specialized languages. However, if your model involves some novel covariance or mean structures that are not covered by the more specialized modeling languages, you can consider the more generalized modeling languages. See Example 26.32 for an application of the generalized COSAN model.

Syntax: CALIS Procedure

PROC CALIS < *options* > **;**
BOUNDS *boundary constraints* **;**
BY *variables* **;**
COSAN *set of variables, cosan model* **;**
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 *set of variables, ram list* **;**
REFMODEL *model number* < / *options* > **;**
RENAMEPARM *parameter renaming* **;**
SIMTESTS *simultaneous tests definitions* **;**
STD *variance parameters* **;**
STRUCTEQ *set of variables* < *label* > **;**
TESTFUNC *parametric functions* **;**
VAR *variables* **;**
VARIANCE *variance parameters* **;**
VARNAMES *name assignments* **;**
WEIGHT *variable* **;**
SAS Programming statements **;**

Classes of Statements in PROC CALIS

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

PROC CALIS Statement

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

GROUP Statement

signifies the beginning of a group specification. A group in the CALIS procedure is an independent sample of observations. You can specify options for input and output data sets, printing, and statistical computations in this statement. Some of these group options in the GROUP statement can also be specified in the MODEL or PROC CALIS statement, but the options specified in the GROUP statement supersede those specified in the MODEL or PROC CALIS statement for the group designated in the GROUP statement. For group options that are available in both of the GROUP and PROC CALIS statements, see the section "Options Available in the GROUP and PROC CALIS Statements" on page 1087. For group options that are available in the GROUP, MODEL, and PROC CALIS statements, see the section "Options Available in GROUP, MODEL, and PROC CALIS Statements" on page 1088. If no GROUP statement is used, a single-group analysis is assumed. The group options for a single-group analysis are specified in the PROC CALIS statement.

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

Subsidiary Group Specification Statements

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

Statement	Description
FREQ on page 1086	Specifies the frequency variable for the input observations
PARTIAL on page 1136	Specifies the partial variables
VAR on page 1164	Specifies the set of variables in analysis
WEIGHT on page 1172	Specifies the weight variable for the input observations

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

MODEL Statement

signifies the beginning of a model specification. In the MODEL statement, you can specify the fitted groups, input and output data sets for model specification or estimates, printing options, statistical analysis, and computational options. Some of the options in the MODEL statement can also be specified in the PROC CALIS statement. These options are called model options. Model options specified in the MODEL statement supersede those specified in the PROC CALIS statement. For model options that are available in both of the MODEL and PROC CALIS statements, see the section "Options Available in the MODEL and PROC CALIS Statements" on page 1128. If no MODEL statement is used, a single model is assumed and the model options are specified in the PROC CALIS statement.

Some of the options in the MODEL statement can also be specified in the GROUP statement. These options are called group options. The group options in the MODEL statement are transferred to the groups being fitted, but they are superseded by the group options specified in the associated GROUP statement. For group options that are available in the GROUP and the MODEL statements, see the section "Options Available in GROUP, MODEL, and PROC CALIS Statements" on page 1088.

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
COSAN on page 1055	Specifies general mean and covariance structures in matrix terms
FACTOR on page 1072	Specifies confirmatory or exploratory factor models
LINEQS on page 1090	Specifies models by using linear equations
LISMOD on page 1097	Specifies models in terms of LISREL-like model matrices
MSTRUCT on page 1130	Specifies parameters directly in the mean and covariance matrices
PATH on page 1137	Specifies models by using the causal paths of variables
RAM on page 1151	Specifies models by using RAM-like lists of parameters
REFMODEL on page 1158	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 1065	Covariance parameters	FACTOR, LINEQS
MATRIX on page 1111	Parameters in matrices	COSAN, LISMOD, MSTRUCT
MEAN on page 1125	Mean or intercept parameters	FACTOR, LINEQS, PATH
PCOV on page 1147	(Partial) covariance parameters	PATH
PVAR on page 1149	(Partial) variance parameters	FACTOR, PATH
RENAMEPARM on page 1160	New parameters by renaming	REFMODEL
VARIANCE on page 1167	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 1070	Sets variable groups for computing the determination coefficients; same as the STRUCTEQ statement
EFFPART on page 1031	Displays and partitions the effects in the model
FITINDEX on page 1082	Controls the fit summary output
LMTESTS on page 1101	Defines the Lagrange multiplier test regions
SIMTESTS on page 1161	Defines simultaneous parametric function tests
STRUCTEQ on page 1070	Sets variable groups for computing the determination coefficients; same as the DETERM statement
TESTFUNC on page 1163	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 1053	Defines the bounds of parameters
LINCON on page 1089	Defines the linear constraints of parameters
NLINCON on page 1132	Defines the nonlinear constraints of parameters
NLOPTIONS on page 1133	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 1054	Fits a model to different groups separately
OUTFILES on page 1134	Controls multiple output data sets
PARAMETERS on page 1136	Defines additional parameters or superparameters
SAS programming statements on page 1161	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 CALIS < *options* > ;
    subsidiary group specification statements ;
    main model specification statement ;
        subsidiary model specification statements ;
    model analysis statements ;
    optimization statements ;
    other statements ;

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

Multiple-Group Multiple-Model Analysis Syntax

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

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

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

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

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

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

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

PROC CALIS Statement

PROC CALIS < *options* > **;**

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

Data Set Options

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

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
READADDPARM	Inputs the generated default parameters in the INMODEL= data set

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
COVPATTERN=	Specifies one of the built-in covariance structures
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
MEANPATTERN=	Specifies one of the built-in mean patterns
MEANSTR	Analyzes the mean structures
METHOD=	Specifies the estimation method
NOBS=	Defines the number of observations
NOMEANSTR	Deactivates the inherited MEANSTR option
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
EXTENDPATH	Displays the extended path estimates
G4=	Specifies the algorithm for computing standard errors
KURTOSIS	Computes and displays kurtosis
MAXMISSPAT=	Specifies the maximum number of missing patterns to display
MODIFICATION	Computes modification indices
NOMISSPAT	Suppresses the display of missing pattern analysis
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
TMISSPAT=	Specifies the data proportion threshold for displaying the missing patterns

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 1022, 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 CALIS Statement Options

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

ASINGULAR | 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:

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

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

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

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

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

BIASKUR

computes univariate skewness and kurtosis by formulas uncorrected for bias.

See the section "Measures of Multivariate Kurtosis" on page 1279 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. Application of these correction factors requires appropriate specification of the covariance structural model suitable for the chi-square correction. For example, using CHICORRECT=UNCORR assumes that you are fitting a covariance structure with free parameters on the diagonal elements and fixed zeros off-diagonal elements of the covariance matrix. Because all the built-in correction factors assume multivariate normality in their derivations, the appropriateness of applying these built-in chi-square corrections to estimation methods other than METHOD=ML is not known.

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

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

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

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

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

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

where p represents the number of variables, k ($k > 1$) represents the number of groups, and $n_i = (N_i - 1)$, with N_i denoting the number of observations in the i-th group. This option is not applied when you also analyze the mean structures or when you fit single-group models.

FIXCOV specifies the correction factor due to Bartlett (1954) for testing a covariance matrix against a hypothetical fixed covariance matrix. The correction factor is:

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

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

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

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

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

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

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

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

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

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

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

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

CLOSEFIT=*p*

defines the criterion value *p* for indicating a close fit. The smaller the better fit. The default value for close fit is .05.

CORRELATION | CORR

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

COVARIANCE | COV

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

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

COVPATTERN | COVPAT=*name*

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

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

$$\boldsymbol{\Sigma} = \begin{pmatrix} \_\mathrm{varparm} & \_\mathrm{covparm} & \_\mathrm{covparm} & \_\mathrm{covparm} \\ \_\mathrm{covparm} & \_\mathrm{varparm} & \_\mathrm{covparm} & \_\mathrm{covparm} \\ \_\mathrm{covparm} & \_\mathrm{covparm} & \_\mathrm{varparm} & \_\mathrm{covparm} \\ \_\mathrm{covparm} & \_\mathrm{covparm} & \_\mathrm{covparm} & \_\mathrm{varparm} \end{pmatrix}$$

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

EQCOVMAT specifies the equality of covariance matrices between multiple groups. That is, this option tests the null hypothesis that

$$H_0 : \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \ldots = \boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$$

where $\boldsymbol{\Sigma}$ is a common covariance matrix for the k $\boldsymbol{\Sigma}_j$'s ($j = 1, \ldots, k; k > 1$). The elements of $\boldsymbol{\Sigma}$ are named _cov_xx_yy automatically by PROC CALIS, where xx represents the row number and yy represents the column number. For example, if there are four variables in the analysis, the common $\boldsymbol{\Sigma}$ is defined as:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \_\mathrm{cov}\_1\_1 & \_\mathrm{cov}\_1\_2 & \_\mathrm{cov}\_1\_3 & \_\mathrm{cov}\_1\_4 \\ \_\mathrm{cov}\_2\_1 & \_\mathrm{cov}\_2\_2 & \_\mathrm{cov}\_2\_3 & \_\mathrm{cov}\_2\_4 \\ \_\mathrm{cov}\_3\_1 & \_\mathrm{cov}\_3\_2 & \_\mathrm{cov}\_3\_3 & \_\mathrm{cov}\_3\_4 \\ \_\mathrm{cov}\_4\_1 & \_\mathrm{cov}\_4\_2 & \_\mathrm{cov}\_4\_3 & \_\mathrm{cov}\_4\_4 \end{pmatrix}$$

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

SATURATED specifies a saturated covariance structure model. This is the default option when you specify the MEANPATTERN= option without using the COVPATTERN= option. The elements of $\mathbf{\Sigma}$ are named _cov_xx_yy automatically by PROC CALIS, where xx represents the row number and yy represents the column number. For example, if there are three variables in the analysis, $\mathbf{\Sigma}$ is defined as:

$$\mathbf{\Sigma} = \begin{pmatrix} \_cov\_1\_1 & \_cov\_1\_2 & \_cov\_1\_3 \\ \_cov\_2\_1 & \_cov\_2\_2 & \_cov\_2\_3 \\ \_cov\_3\_1 & \_cov\_3\_2 & \_cov\_3\_3 \end{pmatrix}$$

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

$$H_0 : \mathbf{\Sigma} = \sigma^2 \mathbf{I}$$

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

$$\mathbf{\Sigma} = \begin{pmatrix} \_varparm & 0 & 0 \\ 0 & \_varparm & 0 \\ 0 & 0 & \_varparm \end{pmatrix}$$

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

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

$$\mathbf{\Sigma} = \begin{pmatrix} \_varparm\_1 & 0 & 0 \\ 0 & \_varparm\_2 & 0 \\ 0 & 0 & \_varparm\_3 \end{pmatrix}$$

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

When you specify the covariance structure model by means of the COVPATTERN= option, you can define the set of variables in the analysis by the VAR statement (either within the scope of the PROC

CALIS statement or the GROUP statements). If the VAR statement is not used, PROC CALIS uses all numerical variables in the data sets.

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

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

In this specification, all three groups are fitted by the compound symmetry pattern. However, there would be no constraints across these groups. PROC CALIS generates two distinct parameters for each group: _varparm_mdl1 and _covparm_mdl1 for Group 1, _varparm_mdl2 and _covparm_mdl2 for Group 2, and _varparm_mdl3 and _covparm_mdl3 for Group 3. Similarly, the _mdlxx suffix, where xx represents the model number, is applied to the parameters defined by the SATURATED, SPHERICITY (or SIGSQI), and UNCORR (or DIAG) covariance patterns in multiple-group situations. However, chi-square correction, whenever it is applicable to single-group analysis, is not applied to such multiple-group analyses.

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

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

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

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

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

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

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

COVSING=*r*

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

DATA=*SAS-data-set*

specifies an input data set that can be an ordinary SAS data set or a specially structured TYPE=CORR, TYPE=COV, TYPE=UCORR, TYPE=UCOV, TYPE=SSCP, or TYPE=FACTOR SAS data set, as described in the section "Input Data Sets" on page 1173. 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.

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 CALIS 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 CALIS does not show results for total effects and their partitioning. See the section "Stability Coefficient of Reciprocal Causation" on page 1275 for more information about the computation of the stability coefficient.

EXTENDPATH | GENPATH

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

FCONV | 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 1246 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,

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

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

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

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

Sometimes, you might create an INMODEL= data set from modifying an existing OUTMODEL= data set. However, editing and modifying OUTMODEL= data sets requires good understanding of the formats and contents of the OUTMODEL= data sets. This process could be error-prone for novice users. For details about the format of INMODEL= or OUTMODEL= data sets, see the section"Input Data Sets" on page 1173.

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

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

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

INSTEP=*r*

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

- For trust-region algorithms (TRUREG, DBLDOG, and LEVMAR), the INSTEP option specifies a positive factor for the initial radius of the trust region. The default initial trust-region radius is the length of the scaled gradient, and it corresponds to the default radius factor of $r = 1$.

- For line-search algorithms (NEWRAP, CONGRA, and QUANEW), INSTEP specifies an upper bound for the initial step length for the line search during the first five iterations. The default initial step length is $r = 1$.

For more details, see the section "Computational Problems" on page 1291.

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

specifies an input data set that contains the weight matrix **W** used in generalized least squares (GLS), weighted least squares (WLS, ADF), or diagonally weighted least squares (DWLS) estimation, if you do not specify the INV option at the same time. The weight matrix must be positive definite because its inverse must be defined in the computation of the objective function. If the weight matrix **W** defined by an INWGT= data set is not positive definite, it can be ridged using the WRIDGE= option. See the section "Estimation Criteria" on page 1246 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 1173. Typically, this input data set is an OUTWGT= data set from a previous PROC CALIS analysis.

INWGTINV

specifies that the INWGT= data set contains the inverse of the weight matrix, rather than the weight matrix itself. This option is effective only with an input weight matrix specified in the INWGT= data set and with the generalized least squares (GLS), weighted least squares (WLS or ADF), or diagonally weighted least squares (DWLS) estimation. With this option, the input matrix provided in the INWGT= data set is not required to be positive definite. Also, the ridging requested by the WRIDGE= option is ignored when you specify the INWGTINV option.

KURTOSIS | KU

computes and displays univariate kurtosis and skewness, various coefficients of multivariate kurtosis, and the numbers of observations that contribute most to the normalized multivariate kurtosis. See the section "Measures of Multivariate Kurtosis" on page 1279 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.

MAXMISSPAT=*n*

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

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

TMISSPAT= value (default at 0.05) until the total number of missing patterns displayed reaches the maximum set by the MAXMISSPAT= option.

MEANPATTERN | MEANPAT=*name*

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

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

$$H_0 : \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \ldots = \boldsymbol{\mu}_k = \boldsymbol{\mu}$$

where $\boldsymbol{\mu}$ is a common mean vector for the k $\boldsymbol{\mu}_j$'s ($j = 1, \ldots, k$). The elements of $\boldsymbol{\mu}$ are named _mean_xx automatically by PROC CALIS, where xx represents the row number. For example, if there are four variables in the analysis, the common $\boldsymbol{\mu}$ is defined as:

$$\boldsymbol{\mu} = \begin{pmatrix} \text{\_mean\_1} \\ \text{\_mean\_2} \\ \text{\_mean\_3} \\ \text{\_mean\_4} \end{pmatrix}$$

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

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

SATURATED specifies a saturated mean structure model. This is the default mean structure pattern when the covariance structures are specified by the COVPATTERN= pattern and the mean structure analysis is invoked by MEANSTR option. The elements of $\boldsymbol{\mu}$ are named _mean_xx automatically by PROC CALIS, where xx represents the row number. For example, if there are three variables in the analysis, $\boldsymbol{\mu}$ is defined as:

$$\boldsymbol{\mu} = \begin{pmatrix} \text{\_mean\_1} \\ \text{\_mean\_2} \\ \text{\_mean\_3} \end{pmatrix}$$

UNIFORM specifies a mean vector with a uniform mean parameter _meanparm. For example, if there are three variables in the analysis, the mean pattern generated by PROC CALIS is

$$\boldsymbol{\mu} = \begin{pmatrix} \text{\_meanparm} \\ \text{\_meanparm} \\ \text{\_meanparm} \end{pmatrix}$$

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

$$\mu = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

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

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

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

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

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

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

Group 3 is fitted by Model 3, which is specified explicitly by a PATH model with a distinct mean parameter mean_x1.

If the EQMEANVEC pattern is specified instead, as shown in the following statements, the equality of mean vectors still holds for Groups 1 and 2:

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

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

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

MEANSTR

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

METHOD | MET | M=*name*

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

FIML	performs full information maximum-likelihood parameter estimation for data with missing values. This method assumes raw input data sets. Exploratory factor analysis and model modification indices are not available with FIML in this version of PROC CALIS. If METHOD=FIML is specified with exploratory factor models, ML is used instead.
ML | M | MAX	performs normal-theory maximum-likelihood parameter estimation. The ML method requires a nonsingular covariance or correlation matrix.
GLS | G	performs generalized least squares parameter estimation. If no INWGT= data set is specified, the GLS method uses the inverse sample covariance or correlation matrix as the weight matrix **W**. Therefore, METHOD=GLS requires a nonsingular covariance or correlation matrix.
WLS | W | ADF	performs weighted least squares parameter estimation. If no INWGT= data set is specified, the WLS method uses the inverse matrix of estimated

asymptotic covariances of the sample covariance or correlation matrix as the weight matrix **W**. In this case, the WLS estimation method is equivalent to Browne's asymptotically distribution-free estimation (Browne 1982, 1984). The WLS method requires a nonsingular weight matrix.

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

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

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

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

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

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

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

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

MODIFICATION | MOD

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

The Lagrange multiplier test (Bentler 1986; Lee 1985; Buse 1982) provides an estimate of the 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 1277 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 × *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 1259 for more discussion on the issue.

NOBS=*nobs*

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

NOINDEXTYPE

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

NOMEANSTR

deactivates the inherited MEANSTR option for the analysis of mean structures. You can specify the NOMEANSTR option in both the PROC CALIS and the MODEL statements. When this option is specified in the PROC CALIS statement, it does not have any apparent effect because by default the mean structures are not analyzed. When this option is specified in the MODEL statement, it deactivates the inherited MEANSTR option from the PROC CALIS statement. In other words, this option is mainly used for resetting the default behavior in the local model that is specified within the scope of a particular MODEL statement. If you specify both the MEANSTR and NOMEANSTR options in the same statement, the NOMEANSTR option is ignored.

CAUTION: This option does not remove the mean structure specifications from the model. It only deactivates the MEANSTR option inherited from the PROC CALIS statement. The mean structures of the model are analyzed as long as there are mean structure specifications in the model (for example, when you specify the means or intercepts in any of the main or subsidiary model specification statements).

NOMISSPAT

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

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

NOPARMNAME

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

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

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

OMETHOD | 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 CALIS procedure. The optimization method or technique is specified by using one of the following *names* in the OMETHOD= option:

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

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

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

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

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

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

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

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

For fewer than 500 parameters ($t < 500$), OMETHOD=NRRIDG (Newton-Raphson Ridge) is the default optimization technique for the FIML estimation, and OMETHOD=LEVMAR (Levenberg-Marquardt) is the default optimization technique for the all other estimation methods. For $500 \leq t < 1,000$, OMETHOD=QUANEW (quasi-Newton) is the default method, and for $t \geq 1,000$, OMETHOD=CONGRA (conjugate gradient) is the default method. Each optimization method or technique can be modified in various ways. See the section "Use of Optimization Techniques" on page 1283 for more details.

ORDERALL

prints the model and group results in the order of the model or group numbers, starting from the smallest number. It also 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. You can specify the ORDERSPEC option in both the PROC CALIS and the MODEL statements. When this option is specified in the PROC CALIS statement, it propagates to all models. When this option is specified in the MODEL statement, it applies only to the local model.

OUTEST=*SAS-data-set*

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

See the section "OUTEST= SAS-data-set" on page 1176 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 1190 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 CALIS. The OUTMODEL= data set also contains a set of fit indices; the section "OUTMODEL= SAS-data-set" on page 1180 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 CALIS. See the section "OUTSTAT= SAS-data-set" on page 1186 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 1047). 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 | OUTWEIGHT=*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 CALIS. See the section "OUTWGT= SAS-data-set" on page 1190 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 printing behavior. In contrast, the NOPARMNAME option suppresses the printing of the parameter names in the model results. You can specify the PARMNAME option in both the PROC CALIS and the MODEL statements. When this option is specified in the PROC CALIS statement, it does not have any apparent effect because by default model results show the parameter names. When this option is specified in the MODEL statement, it deactivates the inherited NOPARMNAME option from the PROC CALIS statement. In other words, this option is mainly used for resetting the default behavior in the local model that is specified within the scope of a particular MODEL statement. If you specify both the PARMNAME and NOPARMNAME options in the same statement, the PARMNAME option is ignored.

PCORR | CORR

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

PCOVES | PCE

displays the following:

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

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

PDETERM | PDE

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

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

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

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

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

PLATCOV | PLATMOM | PLC
displays the following:

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

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

PLOTS | PLOT < = *plot-request* >

PLOTS | PLOT < = (*plot-request* < ... *plot-request* >) >
specifies the ODS graphical plots. Currently, the only available ODS graphical plots in PROC CALIS 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

ODS Graphics must be enabled before requesting plots. For example:

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

For more information about enabling and disabling ODS Graphics, see the section “Enabling and Disabling ODS Graphics” on page 612 in Chapter 21, “Statistical Graphics Using ODS.”

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

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 $\mathbf{W}$ 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 $0 \le r \le 1$. 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 1282 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 CALIS to compute a regression model, you can specify RDF= *number-of-regressor-variables* to get approximate standard errors equal to those computed by PROC REG.

READADDPARM | READADD

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

RESIDUAL | 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 1260 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 CALIS 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 CALIS models, you can supply initial estimates individually as parenthesized values after each parameter name. Unspecified initial estimates are usually computed by various reasonable initial estimation methods in PROC CALIS. If none of the initialization methods is able to compute all the unspecified initial estimates, then the remaining unspecified initial estimates are set to r, $10|r|$, 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 1282 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). In contrast, the NOSTDERR option suppresses the printing of the standard error estimates. If you specify neither the STDERR nor the NOSTDERR option, the standard errors are computed for the OUTMODEL= data set. This information is displayed by default when you specify the PRINT option.

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

TMISSPAT | THRESHOLDMISSPAT | THRESMISSPAT=*n*

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

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

UPDATE | 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.

VARDEF= DF | N | WDF | WEIGHT | WGT

specifies the divisor used in the calculation of covariances and standard deviations. The default value is VARDEF=N for the METHOD=FIML, and VARDEF=DF for other estimation methods. The values and associated divisors are displayed in the following table, where k is the number of partial variables specified in the PARTIAL statement. When a WEIGHT statement is used, w_j is the value of the WEIGHT variable in the 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 $\mathbf{W}$. 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 $\mathbf{W}$, and it is effective only with the ASYCOV=CORR option, which is the default for correlation analyses. The often used value $r = 1$ seems to be too small in many cases to fit the diagonal elements of a correlation matrix properly. The default WPENALTY= value emphasizes the importance of the fit of the diagonal elements in the correlation matrix. You can decrease or increase the value of *r* if you want to decrease or increase the importance of the diagonal elements fit. This option is effective only with the WLS or DWLS estimation method and the analysis of a correlation matrix.

See the section "Estimation Criteria" on page 1246 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).

BOUNDS Statement

BOUNDS *constraint* < *, constraint . . .* > **;**

where *constraint* represents

< *number operator* > *parameter-list* < *operator number* >

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

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

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

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

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

```
BOUNDS x > 0;
```

Specify the following statement instead:

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

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

BY Statement

BY *variables* **;**

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

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

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

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

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

COSAN Statement

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

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

matrix_definition < ∗ *matrix_definition* . . . > < *mean_definition* >

or

mean_definition

where *matrix_definition* is of the following form:

matrix_name < (*number_of_columns* < , *matrix_type* < ,*transformation* > >) >

and *mean_definition* is one of the following forms:

[/ *mean_vector***]**

or

[MEAN=*mean_vector***]**

where *mean_vector* is a vector name.

COSAN stands for covariance structure analysis (McDonald 1978, 1980). The COSAN model in PROC CALIS is a generalized version of the original COSAN model. See the section "The COSAN Model" on page 1193 for details of the generalized COSAN model. You can analyze a very wide class of mean and covariance structures with the COSAN modeling language, which consists of the COSAN statement as the main model specification statement and the MATRIX statement as the subsidiary model specification statement. Use the following syntax to specify a COSAN model:

COSAN < **VAR=***variable-list,* > *term* < + *term. . .* > **;**
  MATRIX *matrix-name parameters-in-matrix* **;**
  /* Repeat the MATRIX statement as needed */ **;**
  VARNAMES *name_assignments* **;**

The PROC CALIS statement invokes the COSAN modeling language. You can specify at most one COSAN statement in a model within the scope of either the PROC CALIS statement or a MODEL statement. To complete the COSAN model specification, you might need to add as many MATRIX statements as needed. Optionally, you can provide the variable names for the COSAN model matrices in the VARNAMES statement.

In the COSAN statement, you specify the list of observed variables for analysis in the VAR= option and the formulas for covariance and mean structures in the *terms*. If specified at all, the VAR= option must be specified at the very beginning of the COSAN statement. The order of the variables in the VAR= option is important. It is the same order assumed for the row and column variables in the mean and covariance structures defined in the *terms*. If you do not specify the VAR= option, PROC CALIS selects all the numerical variables in the associated groups for analysis. To avoid confusion about the variables being analyzed in the model, it is recommended that you set the VAR= list explicitly in the COSAN statement.

To define the matrix formulas for the covariance and mean structures, you specify the *terms*, *matrix_definitions*, and *mean_vector* in the COSAN statement. The forms of the covariance and mean structures that are supported in PROC CALIS are mentioned in the section "The COSAN Model" on page 1193. In each *term*, you specify the covariance structures by listing the matrices in the *matrix_definitions*. These matrices must be in the proper order such that their matrix product produces the intended covariance structures. If you want to analyze the corresponding mean structures, specify the trailing *mean_vectors* in the *terms* whenever needed.

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

$$\mathbf{\Sigma} = \mathbf{FPF}' + \mathbf{U}$$

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

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

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

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

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

In the current example, `F(2,GEN)` represents a general rectangular (GEN) matrix $\mathbf{F}$ with two columns. Implicitly, it has six rows because it is the first matrix of the first term in the covariance structure formula. `P(2,SYM)` represents a symmetric (SYM) matrix $\mathbf{P}$ with two columns. Implicitly, it has two rows because it is premultiplied with $\mathbf{F}$, which has two columns. In the second term, `U(6,DIA)` represents a diagonal (DIA) matrix $\mathbf{U}$ with six rows and six columns. Because you do not specify the third argument in these *matrix_definitions*, no transformation is applied to any of the matrices in the covariance structure formula.

PROC CALIS supports the following keywords for *matrix_type*:

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

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

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

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

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

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

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

GEN specifies a general rectangular matrix (default).

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

```
A(3,GEN)
A(3)
A(3,)
A(3, ,)
```

PROC CALIS supports the following two keywords for *transformation*:

INV uses the inverse of the matrix.

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

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

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

```
cosan var = var1-var6,
      B(3,GEN) * R(3,SYM) + H(3,DIA) * R(3,SYM,INV);
```

Mean and Covariance Structures

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

$$\boldsymbol{\mu} = \mathbf{F}\mathbf{v} + \mathbf{a}$$

where $\boldsymbol{\mu}$ is a 6×1 vector for the observed variable means, $\mathbf{v}$ is a 2×1 vector for the factor means, and $\mathbf{a}$ is a 6×1 vector for the intercepts of the observed variables. To include the mean structures in the COSAN model, you need to specify the mean vector at the end of the *terms*, as shown in the following statement:

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

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

$$\boldsymbol{\Sigma} = \mathbf{F}\mathbf{P}\mathbf{F}' + \mathbf{U}$$

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

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

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

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

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

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

The covariance structure formula for this specification is

$$\boldsymbol{\Sigma} = \mathbf{F}\mathbf{P}\mathbf{F}' + \mathbf{U} + \mathbf{0}$$

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

$$\boldsymbol{\mu} = \mathbf{F}\mathbf{v} + \mathbf{0} + \mathbf{a}$$

where $\mathbf{0}$ in the second term represents a zero vector.

Specifying Models with No Explicit Central Covariance Matrices

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

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

Again, assuming that $\mathbf{F}$ is a 6×2 factor loading matrix and $\mathbf{U}$ is a 6×6 diagonal matrix for unique variances, you can specify the covariance structure formula as in the following COSAN statement:

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

In determining the proper formula for the covariance structures, PROC CALIS detects whether the last matrix specified in each *term* is symmetric. If you specify this last matrix explicitly with the SYM, IDE (with the same number of rows and columns), or DIA type, it is certainly a symmetric matrix. If you specify this last matrix without an explicit *matrix_type* and it has the same number of rows and columns, it is also treated as a symmetric matrix for the central covariance matrix of the *term*. Otherwise, this last matrix is not symmetric and PROC CALIS treats the *term* as if an identity matrix has been inserted for the central covariance matrix. For example, for the orthogonal factor model specified in the preceding statement, PROC CALIS correctly generates the first term as $\mathbf{F}\mathbf{F}' = \mathbf{F}\mathbf{I}\mathbf{F}'$ and the second term as $\mathbf{U}$.

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

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

Specifying Mean Structures for Models with No Central Covariance Matrices

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

```
cosan var = var1-var6,
      A(6,GEN) [ / v];
```

The generated formulas for the covariance and mean structures are

$$\begin{aligned}\mathbf{\Sigma} &= \mathbf{A}\mathbf{A}' \\ \boldsymbol{\mu} &= \mathbf{v}\end{aligned}$$

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

$$\begin{aligned}\mathbf{\Sigma} &= \mathbf{A}\mathbf{A}' \\ \boldsymbol{\mu} &= \mathbf{A}\mathbf{v}\end{aligned}$$

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

```
cosan var = var1-var6,
      A(6,GEN) * I(6,IDE) [ / v];
```

Specifying Parameters in Matrices

By specifying the COSAN statement, you define the covariance and mean structures in matrix formulas for the observed variables. To specify the parameters in the model matrices, you need to use the MATRIX statements.

For example, for an orthogonal factor model with six variables (var1–var6) and two factors, the 6×2 factor loading matrix $\mathbf{F}$ might take the following form:

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

The 6×6 unique variance matrix $\mathbf{U}$ might take the following form:

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

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

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

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

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

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

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

See the MATRIX statement on page 1111 for various ways to specify the parameters in matrices.

Matrix Names versus Parameter Names

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

Consider the following two-group analysis example:

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

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

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

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

- Matrices with the same name in the same model are treated as identical.
- Matrices with the same name in different models are not treated as identical.
- Parameters with the same name are identical throughout the entire PROC CALIS specification.

- Cross-model constraints on matrix elements are set by using the same parameter names, but not the same matrix names.

Row and Column Variable Names for Matrices

You can use the VARNAMES statement to define the column variable names for the model matrices of a COSAN model. However, you do not specify the row variable names for the model matrices directly because they are determined by the column variable names of the related matrices in the covariance and mean structure formulas. For example, the following specification names the column variables of matrices **F** and **I**:

```
cosan var = var1-var6,
      F(2,GEN) * I(2,IDE) + U(6,DIA);
varnames
      F = [Factor1 Factor2],
      I = F;
```

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

Default Parameters

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

Modifying a COSAN Model from a Reference Model

In this section, it is assumed that you use a REFMODEL statement within the scope of a MODEL statement and the reference model (or base model) is also a COSAN model. The reference model is referred to as the old model, while the model that makes reference to this old model is referred to as the new model. If the new model is not intended to be an exact copy of the old model, you can use the following extended COSAN modeling language to make modifications within the scope of the MODEL statement for the new model. The syntax is similar to, but not exactly the same as, the ordinary COSAN modeling language. (See the section "COSAN Statement" on page 1055.) The respecification syntax for a COSAN model is as follows:

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

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

You can optionally use the VARNAMES statement in the respecification. If the variable names for a COSAN matrix are defined in the old model but not redefined the new model, all variable names for that matrix are duplicated in the new model. However, specification of variable names for a COSAN matrix in the new model overrides the corresponding specification in the old model.

You can respecify or modify the elements of the COSAN model matrices by using the MATRIX *matrix-name* statements. The syntax of the MATRIX statements for respecifications is the same as that in the ordinary COSAN modeling language, but with one more feature. In the respecification syntax, you can use the missing value '.' to drop a parameter specification from the old model.

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

Duplication: If you do not specify in the new model a parameter location that exists in the old model, the old parameter specification is duplicated in the new model.

Addition: If you specify in the new model a parameter location that does not exist in the old model, the new parameter specification is used in the new model.

Deletion: If you specify in the new model a parameter location that also exists in the old model and the new parameter is denoted by the missing value '.', the old parameter specification is not copied into the new model.

Replacement: If you specify in the new model a parameter location that also exists in the old model and the new parameter is not denoted by the missing value '.', the new parameter specification replaces the old one in the new model.

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

```
proc calis;
   group 1 / data=d1;
   group 2 / data=d2;
   model 1 / group=1;
      cosan
         var = x1-x6,
         F(2,GEN) * PHI(2,SYM)  + PSI(6,SYM);
      matrix F            [1,1] = 1.,
                          [2,1] = load2,
                          [3,1] = load3,
                          [4,2] = 1.,
                          [5,2] = load5,
                          [6,2] = load6;
      matrix PHI          [1,1] = phi1,
                          [2,2] = phi2,
                          [2,1] = phi21;
      matrix PSI          [1,1] = psi1,
                          [2,2] = psi2,
                          [3,3] = psi3,
                          [4,4] = psi4,
                          [5,5] = psi5,
                          [6,6] = psi6;
      varnames F   = [Factor1 Factor2],
               PHI = F;
   model 2 / group=2;
      refmodel 1;
      matrix F            [3,1] = load2;    /* replacement */
      matrix PHI          [2,1] = .;        /* deletion */
      matrix PSI          [3,1] = psi31;    /* addition */
      varnames F   = [FF1 FF2],
run;
```

In this example, Model 2 is the new model which refers to the old model, Model 1. It illustrates the four types of model integration by using the MATRIX statements:

- Duplication: Except for the **F**[3, 1] and **PHI**[2, 1] elements, all parameter specifications in the old model are duplicated in the new model.
- Addition: The **PSI**[3, 1] element is added with a new parameter psi31 in the new model. This indicates the presence of a correlated error in Model 2, but not in Model 1.
- Deletion: The **PHI**[2, 1] element is no longer a free parameter in the new model. This means that the two latent factors are correlated in Model 1, but not in Model 2.
- Replacement: The **F**[3, 1] element defined in Model 2 replaces the definition in the old model. This element is now a free parameter named load2. Because the **F**[2, 1] element (via duplication from the old model) is also a free parameter with this same name, **F**[3, 1] and **F**[2, 1] are constrained to be the same in Model 2, but not in Model 1.

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

COSAN Models and Other Models

Because the COSAN model is a more general model than any other model considered in PROC CALIS, you can virtually fit any other type of model in PROC CALIS by using the COSAN modeling language. See the section "Special Cases of the Generalized COSAN Model" on page 1195, Example 26.28, and Example 26.29 for illustrations and discussions.

In general, it is recommended that you use the more specific modeling languages such as FACTOR, LINEQS, LISMOD, MSTRUCT, PATH, and RAM. Because the COSAN model is very general in its formulation, PROC CALIS cannot exploit the specific model structures to generate reasonable initial estimates the way it does with other specific models such as FACTOR and PATH. If you do not provide initial estimates for a COSAN model, PROC CALIS uses some default starting values such as 0.5. See the START= option for controlling the starting value. See the RANDOM= option for setting random starting values. There are other reasons for preferring specific modeling languages whenever possible. The section "Which Modeling Language?" on page 1012 discusses these various reasons. However, when the covariance structures are complicated and are difficult to specify otherwise, the COSAN modeling language is a very useful tool. See Example 26.30 and Example 26.32 for illustrations.

COV Statement

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

where *assignment* represents

var_list < ∗ *var_list2* > < = *parameter-spec* >

The COV statement is a subsidiary model specification statement for the confirmatory FACTOR and LINEQS models. In the LINEQS model, the COV statement defines the covariances among the exogenous variables, including errors and disturbances. In the confirmatory FACTOR model, the COV statement defines the factor covariances. In each *assignment* of the COV statement, you specify variables in the *var_list* and the *var_list2* lists, followed by the covariance parameter specification in the *parameter-spec* list. The latter two specifications are optional.

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

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

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

```
cov V2 V1 ,
    V3 V1 = (0.3),
    V3 V2 = 1.0,
    V4 V1 = phi1,
    V4 V2 = phi2(0.2);
```

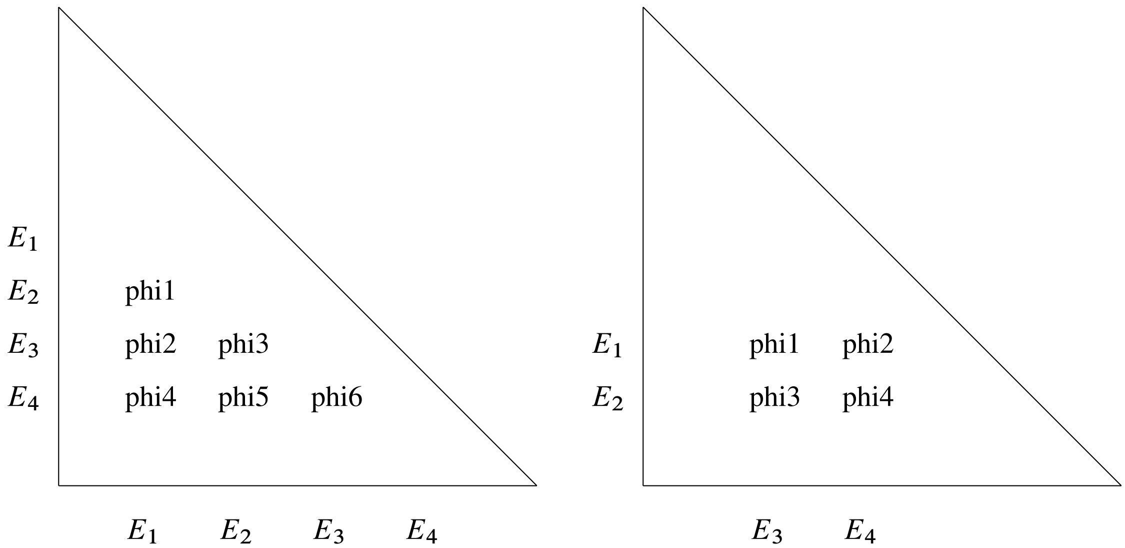
In this statement, **cov(V2,V1)** is specified as an unnamed free parameter. For this covariance, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1). **cov(V3,V1)** is an unnamed free parameter but with an initial value of 0.3. PROC CALIS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

Note that the *var_list* and *var_list2* lists on the left-hand side of the equal sign of the COV statement should contain only names of *exogenous* variables. Hence, the COV statement is different from the PCOV statement in which you can list both exogenous and endogenous variables, although the COV and PCOV statements share the same syntax.

You can use the COV statement for specifying covariance parameters in the FACTOR and LINEQS models. In the FACTOR model, the COV statement specifies the covariances among latent factors. In the LINEQS model, the COV statement specifies the covariances among all observed or latent exogenous variables, including error and disturbance terms.

If you specify only the *var_list* list, then you are specifying the so-called within-list covariances. If you specify both of the *var_list* and *var_list2* lists, then you are specifying the so-called between-list covariances. An asterisk is used to separate the two variable lists. You can use one of these two alternatives to specify the covariance parameters. Figure 26.2 illustrates the within-list and between-list covariance specifications.

Figure 26.2 Within-List and Between-List Covariances



Within-List Covariances   Between-List Covariances

Within-List Covariances

The left panel of the figure shows that the same set of four variables are used in both the rows and columns. This yields six nonredundant covariances to specify. In general, with a *var_list* list with k variables in the COV statement, there are $k(k-1)/2$ distinct covariance parameters you can specify. The variable order of the *var_list* list is important. For example, the left panel of Figure 26.2 corresponds to the following COV statement specification:

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

This specification is equivalent to the following specification:

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

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

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

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

The root of the prefix-name should have few characters so that the generated parameter name is not longer than 32 characters. To avoid unintentional equality constraints, the prefix-names should not coincide with other parameter names.

You can also specify the within-list covariances as unnamed free parameters, as shown in the following statement:

```
cov E1-E4;
```

This specification is equivalent to the following specification:

```
cov E2 E1,
    E3 E1, E3 E2,
    E4 E1, E4 E2, E4 E3;
```

Between-List Covariances

The right panel of Figure 26.2 illustrates the application of the between-list covariance specification. The set of row variables is different from the set of column variables. You intend to specify the cross covariances of the two sets of variables. There are four of these covariances in the figure. In general, with k_1 and k_2 variable names in the two variable lists (separated by an asterisk) in a COV statement, there are $k_1 \times k_2$ distinct covariances to specify. Again, variable order is very important. For example, the right panel of Figure 26.2 corresponds to the following between-list covariance 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;
```

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

```
cov  E1 E2 * E3 E4 = 4*phi__ ; /* phi with two trailing underscores */
```

Mixed Parameter Lists

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

```
cov E1-E4 = phi1(0.1)  0.2  phi3  phi4(0.4) (0.5) phi6;
```

This specification is equivalent to the following specification:

```
cov E2 E1 = phi1(0.1) ,
    E3 E1 = 0.2       , E3 E2 = phi3,
    E4 E1 = phi4(0.4) , E4 E2 = (0.5), E4 E3 = phi6;
```

Notice that an initial value that follows a parameter name is associated with the free parameter. Therefore, in the original mixed list specification, 0.1 is interpreted as the initial value for the parameter phi1, but not as the initial estimate for the covariance between E3 and E1. Similarly, 0.4 is the initial value for the parameter phi4, but not the initial estimate for the covariance between E4 and E2.

However, if you indeed want to specify that phi1 is a free parameter *without* an initial value and 0.1 is an initial estimate for the covariance between E3 and E1 (while keeping all other things the same), you can use a null initial value specification for the parameter phi1, as shown in the following statement:

```
cov E1-E4 = phi1() (0.1)  phi3  phi4(0.4) (0.5) phi6;
```

This way 0.1 becomes the initial estimate for the covariance between E3 and E1. Because a parameter list with mixed types might be confusing, you can break down the specifications into separate *assignments* to remove ambiguities. For example, you can use the following equivalent specification:

```
cov E2 E1 = phi1 ,
    E3 E1 = (0.1)     , E3 E2 = phi3,
    E4 E1 = phi4(0.4) , E4 E2 = (0.5), E4 E3 = phi6;
```

Shorter and Longer Parameter Lists

If you provide fewer parameters than the number of covariances in the variable lists, all the remaining parameters are treated as unnamed free parameters. For example, the following specification assigns a fixed value to **cov(E1,E3)** while treating all the other three covariances as unnamed free parameters:

```
cov  E1 E2 * E3 E4 = 1.0;
```

This specification is equivalent to the following specification:

```
cov  E1 E3 = 1.0, E1 E4, E2 E3, E2 E4;
```

If you intend to fill up all values by the last parameter specification in the list, you can use the continuation syntax **[...]**, **[..]**, or **[.]**, as shown in the following example:

```
cov  E1 E2 * E3 E4 = 1.0 phi [...];
```

This means that **cov(E1,E3)** is a fixed value of 1 and all the remaining three covariances are free parameter named phi. The last three covariances are thus constrained to be equal by using the same parameter name.

However, you must be careful not to provide too many parameters. For example, the following specification results in an error:

```
cov  E1 E2 * E3 E4 = 1.0 phi2(2.0) phi3 phi4 phi5 phi6;
```

The parameters after phi4 are excessive.

Default Covariance Parameters

In the confirmatory FACTOR model, by default all factor covariances are free parameters. In the LINEQS model, by default all covariances among exogenous manifest and latent variables (excluding error or disturbance variables) are also free parameters. For these default free parameters, PROC CALIS generate the parameter names with the _Add prefix and appended with unique integer suffixes. You can also use the COV statement specification to override these default covariance parameters in situations where you want to set parameter constraints, provide initial or fixed values, or make parameter references.

Another type of default covariances are fixed zeros. In the LINEQS model, covariances among errors or disturbances are all fixed zeros by default. Again, you can override the default fixed values by providing explicit specification of these covariances in the COV statement.

Modifying a Covariance Parameter Specification from a Reference Model

If you define a new model by using a reference (old) model in the REFMODEL statement, you might want to modify some parameter specifications from the COV statement of the reference model before transferring the specifications to the new model. To change a particular covariance specification from the reference model, you can simply respecify the same covariance with the desired parameter specification in the COV statement of the new model. To delete a particular covariance parameter from the reference model, you can specify the desired covariance with a missing value specification in the COV statement of the new model.

For example, suppose that the covariance between variables V1 and V2 is specified in the reference model but you do not want this covariance specification be transferred to the new model. You can use the following COV statement specification in the new model:

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

Note that the missing value syntax is valid only when you use it with the REFMODEL statement. See the section "Modifying a LINEQS Model from a Reference Model" on page 1094 for a more detailed example of the LINEQS model respecification with the REFMODEL statement. See the section "Modifying a FACTOR Model from a Reference Model" on page 1080 for a more detailed example of the FACTOR model respecification with the REFMODEL statement.

As discussed in a preceding section, PROC CALIS generates some default free covariance parameters for the LINEQS and FACTOR models if you do not specify them explicitly in the COV statement. When you use the REFMODEL statement for defining a reference model, these default free covariance parameters in the old (reference) model are not transferred to the new model. Instead, the new model generates its own set of default free covariance parameters *after* it is resolved from the reference model, the REFMODEL statement options, the RENAMEPARM statement, and the COV statement specifications in the new model. This also implies that if you want any of the covariance parameters to be constrained across the models by means of the REFMODEL specification, you must specify them explicitly in the COV statement of the reference model so that the same covariance specification is transferred to the new model.

DETERM Statement

DETERM | STRUCTEQ *variables* < */ option* > **;**

where *option* represents:

LABEL | NAME = *name*

The DETERM statement is used to compute the determination coefficient of the listed dependent *variables* in the model. The precursor of the DETERM statement is the STRUCTEQ statement, which enables you to define the list of the dependent variables of the structural equations. Because the term *structural equation* is not defined in a unique way, a more generic concept of determination coefficients is revealed by the DETERM statement.

You can specify the DETERM statement as many times as you want for computing determination coefficients for the sets of dependent *variables* of interest. You can label each set of dependent variables by using the LABEL= option. Note that you cannot use the DETERM statement in an MSTRUCT model because there are no dependent variables in this type of model.

EFFPART Statement

EFFPART *effect* < *, effect* > **;**

where *effect* represents:

var_list < *direction var_list2* >

and *direction* is the direction of the effect, as indicated by one of the following: --->, -->, ->, >, <---, <--, <-, 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 CALIS statement also enables you to analyze effects. The difference is that the EFFPART or TOTEFF option displays effects on *all* endogenous variables, while the EFFPART statement shows only the effects of interest. In addition, the EFFPART statement enables you to arrange the effects in any way you like. Hence, the EFFPART statement offers a more precise and organized way to present various results of effects.

The EFFPART statement supports the following three types of effect specifications:

- > , ->, -->, or ---> direction

  Example:

  ```
  effpart X1 X3-X5 ---> Y1 Y2;
  ```

  This will display *four* separate tables, respectively for the effects of X1, X3, X4, and X5 on Y1 and Y2. Each table contains the total, direct, and indirect effects of an X-variable on the two Y-variables.

- < , <-, <--, or <--- direction

  Example:

  ```
  effpart Y1 Y2 <--- X1 X3-X5;
  ```

  This will display *two* separate tables, respectively for the effects on Y1 and Y2, by X1, X3, X4, and X5. Each table contains the total, direct, and indirect effects of the four X-variables on a Y-variable. Certainly, the results produced from this statement are essentially the same as the previous statement. The difference is about the organization of the effects in the tables.

- no direction

  Example:

  ```
  effpart Y1 Y2 X1-X3;
  ```

  In this case, variables on the list are analyzed one by one to determine the nature of the effects. If a variable has nonzero effects on any other variables in the model, a table of the total, direct, and indirect effects of the variable on those variables is displayed. If a variable is endogenous, a table of total, direct, and indirect effects of those variables that have nonzero effects on the variable is displayed. Note that an endogenous variable in a model might also have effects on other endogenous variables. Therefore, the two cases mentioned are not mutually exclusive—a variable listed in the EFFPART statement might yield two tables for effect analysis.

FACTOR Statement

FACTOR < *EFA_options* | *CFA_spec* > **;**

where *EFA_options* are options for the exploratory factor analysis that are described in the section “Exploratory Factor Analysis” on page 1072 and *CFA_spec* is a specification of confirmatory factor analysis that is described in the section “Confirmatory Factor Analysis” on page 1076.

In the FACTOR statement, you can specify either *EFA_options*, *CFA_spec*, or neither of these. However, you cannot specify both *EFA_options* and *CFA_spec* at the same time. If no option is specified or there is at least one *EFA_option* (exploratory factor analysis option) specified in the FACTOR statement, an exploratory factor model is analyzed. Otherwise, a confirmatory factor model is analyzed with the *CFA_spec*. These two types of models are discussed in the next two sections.

Exploratory Factor Analysis

FACTOR < *EFA_options* > **;**

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

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

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 $\mathbf{\Sigma}$, and n denotes the number of factors (or components, if the COMPONENT option is specified in the FACTOR statement) corresponding to the columns of the factor loading matrix $\mathbf{F}$. 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 $\tilde{\mathbf{F}} = \mathbf{F}\mathbf{\Theta}$ satisfies the same model structures as $\mathbf{F}$ does, where $\mathbf{\Theta}$ is any orthogonal matrix so that $\mathbf{\Theta}'\mathbf{\Theta} = \mathbf{\Theta}\mathbf{\Theta}' = \mathbf{I}$. Mathematically, the covariance or correlation structures can be expressed as:

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

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 CALIS. Initial estimates for factor loadings and unique variances are computed by an algebraic method of approximate factor analysis. Given the initial estimates, final estimates are obtained through the iterative optimization of an objective function, which depends on the estimation method specified in the METHOD= option (default with ML—maximum likelihood) of the PROC CALIS statement.

To make the factor solution more interpretable, you can use the ROTATE= option in the FACTOR statement to obtain a rotated factor loading matrix with a "simple" pattern. Rotation can be orthogonal or oblique. The rotated factors remain uncorrelated after an orthogonal rotation but would be correlated after an oblique rotation. The model structures of an oblique solution are expressed in the following equation:

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

where $\tilde{\mathbf{F}}$ is the rotated factor loading matrix and $\mathbf{P}$ is a symmetric matrix for factor correlations. See the sections "The FACTOR Model" on page 1197 and "Exploratory Factor Analysis Models" on page 1199 for more details about exploratory factor models.

You can also do exploratory factor analysis by the more dedicated FACTOR procedure. Even though extensive comparisons of the factor analysis capabilities between the FACTOR and CALIS procedures are not attempted here, some general points can be made here. In general, the FACTOR procedure provides more factor analysis options than the CALIS procedure does, although both procedures have some unique factor analysis features that are not shared by the other. PROC CALIS requires more computing time and memory than PROC FACTOR because it is designed for more general structural estimation problems and is not able to exploit all the special properties of the unconstrained factor analysis model. For maximum likelihood analysis, you can use either PROC FACTOR (with METHOD=ML, which is not the default method in PROC FACTOR) or PROC CALIS. Because the initial unrotated factor solution obtained by PROC FACTOR uses a different set of identification constraints than that of PROC CALIS, you would observe different initial ML factor solutions for the procedures. Nonetheless, the initial solutions by both procedures are statistically equivalent.

The following *EFA_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 $\mathbf{FF}'$ is equal to the number n of components in $\mathbf{F}$. If n is smaller than the number of variables in the moment matrix $\mathbf{\Sigma}$, the matrix of predicted model values is singular and maximum likelihood estimates for $\mathbf{F}$ cannot be computed. You should compute ULS estimates in this case.

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 $\mathbf{\Sigma} = \mathbf{U}$, 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 34, "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 2161 in Chapter 34, "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 2161 in Chapter 34, "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 *name*s 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 2161 in Chapter 34, "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 2161 in Chapter 34, "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 2161 in Chapter 34, "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-variables-relation* < , *factor-variables-relation* . . . > **;**

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

factor right-arrow var_list < **=** *parameter-spec* >

where *right-arrow* is one of the following: --->, -->, ->, or > .
To complete the specification of a confirmatory factor model, you might need to use the PVAR, COV, and MEAN statements to specify the variance, partial variance, covariance, and mean parameters in the model, as shown in the following syntax:

FACTOR *factor-variable-relation* < , *factor-variables-relation* . . . > **;**
  PVAR *partial-variance-parameters* **;**
  COV *covariance-parameters* **;**
  MEAN *mean-parameters* **;**

The model structures for the covariance matrix $\boldsymbol{\Sigma}$ of the confirmatory factor model are described in the equation

$$\boldsymbol{\Sigma} = \mathbf{FPF}' + \mathbf{U}$$

where $\mathbf{F}$ is the factor loading matrix, $\mathbf{P}$ is a symmetric matrix for factor correlations, and $\mathbf{U}$ 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 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 1197 and "Confirmatory Factor Analysis Models" on page 1201 for more details about confirmatory factor models.

The FACTOR statement is the main model specification statement for the confirmatory factor model. The specifications in the FACTOR statement concern the factor loading pattern in the **F** matrix. More details follow after a brief description of the subsidiary model specification statements: PVAR, COV, and MEAN.

By default, the factor variance parameters in the diagonal of matrix **P** and the error variances in the diagonal of matrix **U** are free parameters in the confirmatory factor model. However, you can override these default parameters by specifying them explicitly in the PVAR statement. For example, in some confirmatory factor models, you might want to set some of these variances to fixed constants, or you might want to set equality constraints by using the same parameter name at different parameter locations in your model.

By default, factor covariances, which are the off-diagonal elements of matrix **P**, are free parameters in the confirmatory factor model. However, you can override these default covariance parameters by specifying them explicitly in the COV statement. Note that you cannot use the COV statement to specify the error covariances—they are always fixed zeros in the confirmatory factor analysis model.

By default, all factor means are fixed zeros and all intercepts are free parameters if the mean structures are analyzed. You can override these defaults by explicitly specifying the means of the factors in vector $\boldsymbol{\nu}$ and the intercepts of the manifest variables in vector $\boldsymbol{\alpha}$ in the MEAN statement.

Because the default parameterization of the confirmatory FACTOR model already covers most commonly used parameters in matrices **P**, **U**, $\boldsymbol{\alpha}$, and $\boldsymbol{\nu}$, the specifications in the PVAR, COV, and MEAN statements are secondary to the specifications in the FACTOR statement, which specifies the factor pattern of the **F** matrix. The following example statement introduces the syntax of the confirmatory FACTOR statement. Suppose that there are nine manifest variables V1-V9 in your sample and you want to fit a model with four factors, as shown in the following FACTOR statement:

```
factor
   g_factor   --->   V1-V9 ,
   factor_a   --->   V1-V3 ,
   factor_b   --->   V4-V6 ,
   factor_c   --->   V7-V9 ;
```

In this factor model, you assume a general factor g_factor and three group-factors: factor_a, factor_b, and factor_c. The general factor g_factor is related to all manifest variables in the sample, while each group-factor is related only to three manifest variables. This example fits the following pattern of factor pattern of **F**:

	g_factor	factor_a	factor_b	factor_c
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

where an x represents an unnamed free parameter and all other cells that are blank are fixed zeros. For each of these unnamed parameters, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1, _Parm2 and so on).

An unnamed free parameter is only one of the following five types of parameters (*parameter-spec*) you can specify at the end of each *factor-variables-relation*:

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

To illustrate these different types of parameter specifications, consider the following factor pattern for **F**:

```
          g_factor   factor_a   factor_b   factor_c

    V1     g_load1    1.
    V2     g_load2    x
    V3     g_load3    x
    V4     g_load4               1.
    V5     g_load5               load_a
    V6     g_load6               load_b
    V7     g_load7                          1.
    V8     g_load8                          load_c
    V9     g_load9                          load_c
```

where an x represents an unnamed free parameter, a constant 1 represents a fixed value, and each name in a cell represents a name for a free parameter. You can specify this factor pattern by using the following FACTOR statement:

```
factor
   g_factor   --->   V1-V9    = g_load1-g_load9 (9*0.6),
   factor_a   --->   V1-V3    = 1. (.7  .8),
   factor_b   --->   V4-V6    = 1. load_a (.9) load_b,
   factor_c   --->   V7-V9    = 1. 2*load_c ;
```

In the first entry of the FACTOR statement, you specify that the loadings of V1–V9 on g_factor are free parameters g_load1–g_load9 with all given an initial estimate of 0.6. The syntax `9*0.6` means that `0.6` is repeated nine times. Because they are enclosed in a pair parentheses, all these values are treated as initial estimates, but not fixed values.

The second entry of the FACTOR statement can be split into the following specification:

```
   factor_a   --->   V1    = 1.  ,
   factor_a   --->   V2    = (.7),
   factor_a   --->   V3    = (.8),
```

This means that the first loading is a fixed value of 1, while the other loadings are unnamed free parameters with initial estimates 0.7 and 0.8, respectively. For each of these unnamed parameters with initial values, PROC CALIS also generates a parameter name with the _Parm prefix and appended with a unique integer.

The third entry of the FACTOR statement can be split into the following specification:

```
factor_b   --->   V4    = 1.  ,
factor_b   --->   V5    = load_a (.9),
factor_b   --->   V6    = load_b,
```

This means that the first loading is a fixed value of 1, the second loading is a free parameter named load_a with an initial estimate of 0.9, and the third loading is a free parameter named load_b without an initial estimate. PROC CALIS generates the initial value for this free parameter.

The fourth entry of the FACTOR statement states that the first loading is a fixed 1 and the remaining two loadings are free parameters named load_c. No initial estimate is given. But because the two loadings have the same parameter name, they are constrained to be equal in the estimation.

Notice that an initial value that follows after a parameter name is associated with the free parameter. For example, in the third entry of the FACTOR statement, the specification `(.9)` after load_a is interpreted as the initial value for the parameter load_a, but not as the initial estimate for the next loading for V6.

However, if you indeed want to specify that load_a is a free parameter *without* an initial value and `(0.9)` is an initial estimate for the loading for V6, you can use a null initial value specification for the parameter load_a, as shown in the following specification:

```
factor_b   --->   V4-V6    = 1. load_a() (.9),
```

This way 0.9 becomes the initial estimate of the loading for V6. Because a parameter list with mixed parameter types might be confusing, you can split the specification into separate entries to remove ambiguities. For example, you can use the following equivalent specification:

```
factor_b   --->   V4    = 1.,
factor_b   --->   V5    = load_a,
factor_b   --->   V6    = (.9),
```

Shorter and Longer Parameter Lists

If you provide fewer parameters than the number of loadings that are specified in the corresponding *factor-variable-relation*, all the remaining parameters are treated as unnamed free parameters. For example, the following specification assigns a fixed value of 1.0 to the first loading, while treating the remaining two loadings as unnamed free parameters:

```
factor
   factor_a   --->   V1-V3    = 1.;
```

This specification is equivalent to the following specification:

```
factor
   factor_a   --->   V1      = 1.,
   factor_a   --->   V2 V3      ;
```

If you intend to fill up all values with the last parameter specification in the list, you can use the continuation syntax `[...]`, `[..]`, or `[.]`, as shown in the following example:

```
factor
   g_factor   --->   V1-V30    = 1.   (.5) [...];
```

This means that the loading of V1 on g_factor is a fixed value of 1.0, while the remaining 29 loadings are unnamed free parameters with all given an initial estimate of 0.5.

However, you must be careful not to provide too many parameters. For example, the following specification results in an error:

```
factor
   g_factor   --->   V1-V3    = load1-load6;
```

The parameter list has six parameters for three loadings. Parameters after load3 are excessive.

Default Parameters

It is important to understand the default parameters in the FACTOR model. First, if you know which parameters are default free parameters, you can make your specification more efficient by omitting the specifications of those parameters that can be set by default. For example, because all error variances in the confirmatory FACTOR model are free parameters by default, you do not need to specify them with the PVAR statement if these error variances are not constrained. Second, if you know which parameters are default free parameters, you can specify your model accurately. For example, because all factor variance and covariances in the confirmatory FACTOR model are free parameters by default, you must use the COV statement to restrict the covariances among the factors if you want to fit an orthogonal factor model. See the section "Default Parameters in the FACTOR Model" on page 1204 for details about the default parameters of the FACTOR model.

Modifying a FACTOR Model from a Reference Model

This section assumes that you use a REFMODEL statement within the scope of a MODEL statement and that the reference model (or base model) is a factor model, either exploratory or confirmatory. The reference model is called the old model, and the model that refers to the old model is called the new model. If the new model is not intended to be an exact copy of the old FACTOR model, you can use the extended FACTOR modeling language described in this section to make modifications from the old model before transferring the specifications to the new model.

Using the REFMODEL statement for defining new factor models is not recommended in the following cases:

- If your old model is an exploratory factor analysis model, then specification by using the FACTOR modeling language in the new model replaces the old model completely. In this case, the use of the REFMODEL statement is superfluous and should be avoided.
- If your old model is a confirmatory factor analysis model, then specification of an exploratory factor model by using the FACTOR statement in the new model also replaces the old model completely. Again, the use of the REFMODEL statement is superfluous and should be avoided.

The nontrivial case where you might find the REFMODEL statement useful is when you modify an old confirmatory factor model to form a new confirmatory factor model. This nontrivial case is the focus of discussion in the remaining of the section.

The extended FACTOR modeling language for modifying model specification bears the same syntax as that of the ordinary FACTOR modeling language (see the section “Confirmatory Factor Analysis” on page 1076). 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 calis;
   group 1 / data=d1;
   group 2 / data=d2;
   model 1 / group=1;
      factor
         F1 ---> V1-V3    = 1. load1 load2,
         F2 ---> V4-V6    = 1. load3 load4,
         F3 ---> V7-V9    = 1. load5 load6;
      cov
         F1 F2 = c12,
         F2 F3 = c23;
      pvar
         F1-F3 = c1-c3,
         V1-V9 = ev1-ev9;
   model 2 / group=2;
      refmodel 1;
      factor
         F1 ---> V1    = loada,
         F2 ---> V4    = loadb,
         F3 ---> V7    = loadc;
      cov
         F1 F2 = .,
         F1 F3 = c13;
   run;
```

In this specification, you specify Model 2 by referring to Model 1 in the REFMODEL statement; Model 2 is the new model which refers to the old model, Model 1. Because the PVAR statement is not used in new model, all variance and partial variance parameter specifications in the PVAR statement of the old model are duplicated in the new model. The covariance parameter c23 for covariance between F2 and F3 in the COV statement of the old model is also duplicated in the new model. Similarly, loading parameters load1–load6 for some specific factor matrix locations are duplicated from the old model to the new model.

The new model has an additional parameter specification that the old model does not have. In the COV statement of the new model, covariance parameter c13 for the covariance between F1 and F3 is added.

In the same statement, the covariance between F1 and F2 is denoted by the missing value '.'. The missing value indicates that this parameter location in the old model should not be included in the new model. The consequence of this deletion from the old model is that the covariance between F1 and F2 is a fixed zero in the new model.

Finally, the three new loading specifications in the FACTOR statement of the new model replace the fixed ones in the old model. They are now free parameters loada, loadb, and loadc in the new model.

FITINDEX Statement

FITINDEX *option* < *option* . . . > **;**

You can use the FITINDEX statement to set the options for computing and displaying the fit indices, or to output the fit indices. All but the OFF= and ON= options of the FITINDEX statement are also available in the PROC CALIS statement. The options set in the FITINDEX statement will overwrite those set in the PROC CALIS statement.

For the listing of fit indices and their definitions, see the section "Overall Model Fit Indices" on page 1263. Note that not all fit indices are available with all estimation methods, which is specified by the METHOD= option of the PROC CALIS statement. See the section "Fit Indices and Estimation Methods" on page 1270 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 CALIS statement on page 1025.

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 CALIS statement on page 1025.

CHICORRECT | CHICORR = *name* | *c*

specifies a correction factor *c* for the chi-square statistics for model fit. See the CHICORRECT= option of the PROC CALIS statement on page 1026.

CLOSEFIT=*p*

defines the criterion value *p* for indicating a close fit. See the CLOSEFIT= option of the PROC CALIS statement on page 1027.

DFREDUCE=*i*

reduces the degrees of freedom of the χ^2 test by *i*. See the DFREDUCE= option of the PROC CALIS statement on page 1031.

NOADJDF

turns off the automatic adjustment of degrees of freedom when there are active constraints in the analysis. See the NOADJDF option of the PROC CALIS statement on page 1041.

NOINDEXTYPE

disables the display of index types in the fit summary table. See the NOINDEXTYPE option of the PROC CALIS statement on page 1041.

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 CALIS:

ABSOLUTE	Absolute or stand-alone fit indices that measures the model fit without using a baseline model.
ALL	All fit indices available in PROC CALIS.
INCREMENTAL	Incremental fit indices that measure model fit by comparing with a baseline model.
MODELINFO	General modeling information including sample size, number of variables, number of variables, and so on.
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 CALIS:

BASECHISQ Chi-square statistic for the baseline model.

BASEDF Degrees of freedom of the chi-square statistic for the baseline model.

BASEFUNC Baseline model function value.

BASELOGLIKE Baseline model −2 log-likelihood function value for METHOD=FIML.

BASEPROBCHI P-value of the chi-square statistic for the baseline model fit.

BASESTATUS Status of the baseline model fitting for METHOD=FIML.

NACTCON Number of active constraints.

NIOBS Number of incomplete observations for METHOD=FIML.

NMOMENTS Number of elements in the moment matrices being modeled.

NOBS Number of observations assumed in the analysis.

NPARM | NPARMS Number of independent parameters.

NVAR Number of variables.

SATFUNC Saturated model function value for METHOD=FIML.

SATLOGLIKE Saturated model −2 log-likelihood function value for METHOD=FIML.

SATSTATUS Status of the saturated model fitting for METHOD=FIML.

Output Control of Absolute Fit Indices

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

CHISQ Chi-square statistic for model fit.

CN | CRITICAL_N Hoelter's critical N.

CONTLIKE Percentage contribution to the Log-likelihood function value of each group in multiple-group analyses with METHOD=FIML.

CONTRIBUTION | CONTCHI Percentage contribution to the chi-square value for multiple-group analyses.

DF Degrees of freedom for the chi-square test for model fit.

ELLIPTIC Elliptical chi-square statistic for ML and GLS methods in single-group analyses without mean structures. This index is computed only when you input the raw data with the KURTOSIS option specified.

FUNCVAL Optimized function value.

GFI Goodness-of-fit index by Jöreskog and Sörbom.

LOGLIKE Fitted model −2 log-likelihood function value for METHOD=FIML.

PROBCHI P-value of the chi-square statistic for model fit.

PROBELLIPTIC P-value of the elliptical chi-square statistic.

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 CALIS:

AGFI   Adjusted GFI.

AIC   Akaike information criterion.

CAIC   Bozdogan corrected AIC.

CENTRALITY   McDonald centrality measure.

ECVI   Expected cross-validation index.

ECVI_LL | LL_ECVI   Lower confidence limit for ECVI.

ECVI_UL | UL_ECVI   Upper confidence limit for ECVI .

PGFI   Parsimonious GFI.

PROBCLFIT   Probability of close fit.

RMSEA   Root mean squares of error approximation.

RMSEA_LL | LL_RMSEA   Lower confidence limit for RMSEA.

RMSEA_UL | UL_RMSEA   Upper confidence limit for RMSEA.

SBC   Schwarz Bayesian criterion.

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

BENTLERCFI | CFI   Bentler comparative fit index.

BENTLERNFI   Bentler-Bonett normed fit index.

BENTLERNNFI   Bentler-Bonett nonnormed fit index.

BOLLENNFI   Bollen normed fit index (Rho1).

BOLLENNNFI   Bollen nonnormed fit index (Delta2).

PNFI   James et al. parsimonious normed fit index.

OUTFIT=*SAS-data-set*
creates an output data set containing the values of the fit indices. This is the same as the OUTFIT= option of the PROC CALIS statement on page 1045. See the section "OUTFIT= SAS-data-set" on page 1190 for details.

FREQ Statement

FREQ *variable* **;**

If one variable in your data set represents the frequency of occurrence for the other values in the observation, specify the variable's name in a FREQ statement. PROC CALIS then treats the data set as if each observation appears n_i times, where n_i is the value of the FREQ variable for observation i. Only the integer portion of the value is used. If the value of the FREQ variable is less than 1 or is missing, that observation is not included in the analysis. The total number of observations is considered to be the sum of the FREQ values. You can use only one FREQ statement within the scope of each GROUP or the PROC CALIS statement.

GROUP Statement

GROUP *i* < / *options*> **;**

where *i* is an assigned group number between 1 and 9999, inclusively.

The GROUP statement signifies the beginning of a group specification block and designates a group number for the group. All subsidiary group specification statements after a GROUP statement belong in that group until another MODEL or GROUP statement is used. The subsidiary group specification statements refer to one of the following four statements:

- FREQ statement on page 1086
- PARTIAL statement on page 1136
- VAR statement on page 1164
- WEIGHT statement on page 1172

For example, consider the following statements:

```
proc calis;
   var X1-X4;
   group 1 / label='Women' data=women_data;
      freq  Z;
   group 2 / label='Men' data=men_data;
      partial P;
   model 1 / group = 1-2;
      factor N=1; /* One factor exploratory factor analysis */
   run;
```

In the GROUP statements, two groups are defined. Group 1, labeled as 'Women', refers to the data set women_data. Group 2, labeled as 'Men', refers to the data set men_data. Both groups are fitted by an exploratory factor model defined in Model 1, as indicated in the GROUP= option of the MODEL statement. While the frequency variable Z defined in the FREQ statement is applicable only to Group 1, the partial variable P defined in the PARTIAL statement is applicable only to Group 2. However, the VAR statement, which appears before the definitions of both groups, applies globally to both Group 1 and Group 2. Therefore, variables X1–X4 are the analysis variables in the two groups.

You can set group-specific *options* in each GROUP statement. All but one (that is, the LABEL= option) of these *options* are also available in the MODEL and PROC CALIS statements. If you set these group-specific *options* in the PROC CALIS statement, they will apply to all groups unless you respecify them in the GROUP statement. If you set these group-specific *options* in the MODEL statement, they will apply to all groups that are fitted by the associated model. In general, the group-specific options are transfered from the PROC CALIS statement to the MODEL statements (if present) and then to the fitted groups. In the transferring process, options are overwritten by the newer ones. If you want to apply some group-specific options to a particular group only, you should set those options in the GROUP statement corresponding to that group.

Option Available in the GROUP Statement Only

LABEL | NAME=*name*

specifies a label for the current group. You can use any valid SAS names up to 256 characters for labels. You can also use quote strings for the labels. This option can be specified only in the GROUP statement, not the PROC CALIS statement.

Options Available in the GROUP and PROC CALIS Statements

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

Option	**Description**
DATA= on page 1031	Specifies the input data set
INWGT= on page 1034	Specifies the data set that contains the weight matrix
OUTSTAT= on page 1045	Specifies the data set for storing the statistical results
OUTWGT= on page 1045	Specifies the data set for storing the weight matrix

See the section "Listing of PROC CALIS Statement Options" on page 1025 for more details about these options. If you specify these options in the PROC CALIS statement, they are transferred to *all* GROUP statements. They might be overwritten by the respecifications in the individual GROUP statements.

Options Available in GROUP, MODEL, and PROC CALIS Statements

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

Option	Description
BIASKUR on page 1025	Computes the skewness and kurtosis without bias corrections
EDF= on page 1031	Defines nobs by the number of error df
INWGTINV on page 1034	Specifies that the INWGT= data set contains the inverse of the weight matrix
KURTOSIS on page 1034	Computes and displays kurtosis
MAXMISSPAT= on page 1036	Specifies the maximum number of missing patterns to display
NOBS= on page 1041	Defines the number of observations (nobs)
NOMISSPAT on page 1041	Suppresses the display of missing pattern analysis
PCORR on page 1046	Displays analyzed and estimated moment matrix
PLOTS= on page 1047	Specifies ODS Graphics selection
PWEIGHT on page 1172	Displays the weight matrix
RDF | DFR= on page 1048	Defines nobs by the number of regression df
RESIDUAL | RES on page 1049	Computes the default residuals
RESIDUAL | RES= on page 1049	Specifies the type of residuals
RIDGE on page 1049	Specifies the ridge factor for covariance matrix
SIMPLE on page 1050	Prints univariate statistics
TMISSPAT= on page 1050	Specifies the data proportion threshold for displaying the missing patterns
VARDEF= on page 1051	Specifies variance divisor
WPENALTY= on page 1052	Specifies the penalty weight to fit correlations
WRIDGE= on page 1052	Specifies the ridge factor for the weight matrix

If you specify these options in the PROC CALIS statement, they are transferred to all MODEL statements. These options are overwritten by the respecifications in the individual MODEL statements. After these options are resolved in a given MODEL statement, they are transferred further to the GROUP statements of which the associated groups are fitted by the model. Again, these options might be overwritten by the respecifications in the individual GROUP statements.

LINCON Statement

LINCON *constraint* < , *constraint* . . . > **;**

where *constraint* represents one of the following:

- *number operator linear-term*
- *linear-term operator number*

and *linear-term* is

< +|− >< *coefficient* ∗ > *parameter* << +|− >< *coefficient* ∗ > *parameter* . . . >

The LINCON statement specifies a set of linear equality or inequality constraints of the following form:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, \ldots, m$$

The constraints must be separated by commas. Each linear constraint i in the statement consists of a linear combination $\sum_j a_{ij} x_j$ of a subset of the n parameters x_j, $j = 1, \ldots, n$, and a constant value b_i separated by a comparison operator. Valid operators are <=, <, >=, >, and = or, equivalently, LE, LT, GE, GT, and EQ. PROC CALIS cannot enforce the strict inequalities < or >. Note that the coefficients a_{ij} in the linear combination must be constant numbers and must be followed by an asterisk and the name of a parameter (that is, listed in the PARMS, main, or subsidiary model specification statements). The following is an example of the LINCON statement that sets a linear constraint on parameters x1 and x2:

```
lincon        x1 + 3 * x2 <= 1;
```

Although you can 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 CALIS 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 CALIS statement or a MODEL statement. To completely specify a LINEQS model, you might need to add some subsidiary model specification statements such as the VARIANCE, COV, and MEAN statements. The syntax for the LINEQS modeling language is as follows:

LINEQS < *equation* < , *equation* . . . >> **;**
  VARIANCE *partial-variance-parameters* **;**
  COV *covariance-parameters* **;**
  MEAN *mean-parameters* **;**

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

In the VARIANCE statement, you specify the variance parameters. In the COV statement, you specify the covariance parameters. In the MEAN statement, you specify the mean parameters. For details of these subsidiary model specification statements, see the syntax of these statements.

In the LINEQS statement, in addition to the functional relations among variables, you specify the coefficient parameters of interest in the equations. There are five types of parameters you can specify in equations, as shown in the following example:

```
lineqs
   V1 =          * F1 + E1,
   V2 = (.5)     * F1 + E2,
   V3 = 1.       * F1 + E3,
   V4 = b4       * F1 + E4;
   V5 = b5 (.4) * F1 + E5;
```

In this example, you have manifest variables V1–V5, which are related to a latent factor, denoted by F1, as specified in the equations. In each equation, you have one outcome variable (V-variable), one predictor variable (F1, which is assumed to be a latent factor, the so-called F-variable), and one error variable (E-variable). The following four types of parameters have been specified:

- an unnamed free parameter

  The effect of F1 on V1 in the first equation is an unnamed free parameter. Although you specify nothing before the asterisk sign, the effect parameter is effectively specified. For an unnamed free parameter, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1, _Parm2, and so on).

- an initial value

  The effect of F1 on V2 in the second equation is an unnamed free parameter with an initial estimate of 0.5. PROC CALIS also generates a parameter name for this specification. Notice that you must use a pair of parentheses for the initial value specification because it is interpreted as a fixed value otherwise, as described in the next case.

- a fixed value

  The effect of F1 on V3 in the third equation is an unnamed free parameter with a fixed value of 0.5. A fixed value remains the same in the estimation. There is no parameter name for a fixed constant in the model.

- a free parameter with a name

  The effect of F1 on V4 in the fourth equation is a free parameter named b4. You do not provide an initial estimate for this free parameter.

- a free parameter with a name and an initial estimate

  The effect of F1 on V5 in the fifth equation is a free parameter named b5 with an initial estimate of 0.4. Parameters with no starting values are initialized by various heuristic and effective methods in PROC CALIS. See the section "Initial Estimates" on page 1282 for details.

Notice that there must be an error term in each equation. The error terms in equation must start with the prefix 'E', 'e', 'D', or 'd'. See the section "Representing Latent Variables in the LINEQS Model" on page 1092 for details about naming the factors and error terms. The effect or the path coefficient attached to an error term must be 1.0. This is implicitly specified as in the preceding example. For example, there is no parameter specification nor an asterisk sign before the error term E1 in the first equation, as shown in the following:

```
V1 =          * F1 + E1,
```

This specification is the same as the following explicit specification with a fixed constant 1.0 for the effect of the error term E1:

```
V1 =          * F1 + 1. * E1,
```

The equivalence shown here implies that you can also specify the third equation in the following equivalent way:

```
V3 =  F1 + E3,
```

This implicitly specifies a constant 1.0 for the effect of F1 on V3. You must be very careful about the distinction between this specification and the following one with an asterisk before F1:

```
V3 =  * F1 + E3,
```

With an asterisk sign, the effect of F1 on V3 becomes an unnamed free parameter in the current specification. This interpretation is very different from the preceding one without an asterisk sign before F1, which assumes a fixed constant of 1.0.

Except for the unnamed free parameter specification, you can omit the asterisk signs in all other types of parameter specifications. That is, you can use the following equivalent statement for the preceding LINEQS specification:

```
lineqs
   V1 =          * F1 + E1,
   V2 = (.5)       F1 + E2,
   V3 = 1.         F1 + E3,
   V4 = b4         F1 + E4;
   V5 = b5  (.4)   F1 + E5;
```

Again, you cannot omit the asterisk in the first equation because it is intended to denote an unnamed free parameter.

If your model contains many unconstrained parameters and it is too cumbersome to find different parameter names, you can specify all those parameters by the same prefix-name. A prefix name is a short name called "root" followed by two underscores __. Whenever a prefix-name is encountered, the CALIS procedure generates a parameter name by appending a unique integer to the root. Hence, the prefix-name should have few characters so that the generated parameter name is not longer than thirty-two characters. To avoid unintentional equality constraints, the prefix names should not coincide with explicitly defined parameter names. The following statement illustrates the uses prefix-names:

```
lineqs
   V1 = 1.   * F1 + E1,
   V2 = b__  * F1 + E2,
   V3 = b__  * F1 + E3,
   V4 = b__  * F1 + E4;
   V5 = b__  * F1 + E5;
```

In the five equations, only the first equation has a fixed constant 1.0 for the effect of F1 on V1. For all the remaining equations, the effects of F1 on the variables are all free parameters with the prefix b. The generated parameter names for these effects have unique integers appended to this prefix. For example, b1, b2, b3, and b4 are the parameter names for these effects.

Representing Latent Variables in the LINEQS Model

Because latent variables are widely used in structural equation modeling, PROC CALIS needs a way to identify different types of latent variables that are specified in the LINEQS model. This is accomplished by following some naming conventions for the latent variables. See the section "Naming Variables in the LINEQS Model" on page 1206 for details about these naming rules. Essentially, latent factors (systematic sources) must start with the letter 'F' or 'f'. Error terms must start with the letter 'E', 'e', 'D', or 'd'. Prefix 'E' or 'e' represents the error term of an endogenous *manifest* variable. Prefix 'D' or 'd' represents the disturbance (or error) term of an endogenous *latent* variable. Although D- and E- variables are conceptually different, for modeling purposes 'D' and 'E' prefixes are interchangeable in the LINEQS modeling language. Essentially, only the distinction between latent factors (systematic sources) and errors or disturbances (unsystematic sources) is critical in specifying a proper LINEQS model. Manifest variables in the

LINEQS model do not need to follow additional naming rules beyond those required by the general SAS System—they are recognized by PROC CALIS by referring to the variables in the input data sets.

Types of Variables and Semantic Rules of Equations

Depending on their roles in the system of equations, variables in a LINEQS model can be classified into endogenous or exogenous. An endogenous variable is a variable that serves as an outcome variable (left-hand side of an equation) in one of the equations. All other variables are exogenous variables, including those manifest variables that do not appear in any equations but are included in the model because they are specified in the VAR statement for the analysis.

Merely following the syntactic rules described so far is not sufficient to define a proper system of equations that PROC CALIS can analyze. You also need to observe the following semantic rules:

- Only manifest or latent variables can be endogenous. This means that you cannot specify any error or disturbances variables on the left-hand side of the equations. This also means that error and disturbance variables are always exogenous in the LINEQS model.
- An endogenous variable that appears on the left-hand side of an equation cannot appear on the left-hand side of another equation. In other words, you have to specify all the predictors for an endogenous variable in a single equation.
- An endogenous variable that appears on the left-hand side of an equation cannot appear on the right-hand side of the same equation. This prevents a variable to have a direct effect on itself (but indirect effect on itself is possible).
- Each equation must contain one and only one *unique* error term, be it an E-variable for a manifest outcome variable or a D-variable for a latent outcome variable. If, indeed, you want to specify an equation without an error term, you can equivalently set the variance of the error term to a fixed zero in the VARIANCE statement.

Mean Structures in Equations

To fit a LINEQS model with mean structures, you can specify the MEANSTR option in the PROC CALIS or the associated MODEL statement. This generates the default mean and intercept parameters for the model (see the section “Default Parameters” on page 1094). Alternatively, you can specify the intercept parameters with the Intercept variable in the equations or the mean parameters in the MEAN statement. The Intercept variable in the LINEQS model is a special “variable” that contains the value 1 for each observation. You do not need to have this variable in your input data set, nor do you need to generate it in the DATA step. It serves as a notational convenience in the LINEQS modeling language. The actual intercept parameter is expressed as a coefficient parameter with the intercept variable. For example, consider the following LINEQS model specification:

```
lineqs
   V1 = a1 (10) * Intercept +  1.0     * F1 + E1,
   V2 =         * Intercept +          * F1 + E2,
   V3 =                     +  b2      * F1 + E3,
   V4 = a2      * Intercept +  b2      * F1 + E4,
```

```
    V5 = a2      * Intercept +  b4 (.4) * F1 + E5;
```

In the first equation, a1, with a starting value at 10, is the intercept parameter of V1. In the second equation, the intercept parameter of V2 is an unnamed free parameter. In the third equation, although you do not specify the Intercept variable, the intercept parameter of manifest variable V3 is assumed to be a free parameter by default. See the section "Default Parameters" on page 1094 for more details about default parameters. In the fourth and the fifth equations, the intercept parameters are both named a2. This means that these intercepts are constrained to be the same in the estimation.

In some cases, you might need to set the intercepts to fixed constants such as zeros. You can use the following syntax:

```
lineqs
   V1 = 0 * Intercept +  F_intercept + a2 * F_slope + E1;
```

This sets the intercept parameter of V1 to a fixed zero. An example of this application is the analysis of latent growth curve model in which you define the intercept as a random variable represented by a latent factor (for example, F_intercept in the specification). See Example 26.24 for a detailed example.

To complete the specification of the mean structures in the LINEQS model, you might want to use the MEAN statement to specify the mean parameters. For example, the following statements specify the means of F_intercept and F_slope as unnamed free parameters in the LINEQS model:

```
lineqs
   V1 = 0 * Intercept +  F_intercept + 1 * F_slope + E1;
mean
   F_intercept F_slope;
```

See the MEAN statement for details.

Default Parameters

It is important to understand the default parameters in the LINEQS model. First, if you know which parameters are default free parameters, you can make your specification more efficient by omitting the specifications of those parameters that can be set by default. For example, because all variances and covariances among exogenous variables (excluding error terms) are free parameters by default, you do not need to specify them with the COV and VARIANCE statements if these variances and covariances are not constrained. Second, if you know which parameters are default fixed zero parameters, you can specify your model accurately. For example, because all error covariances in the LINEQS model are fixed zeros by default, you must use the COV statement to specify the covariances among the errors if you want to fit a model with correlated errors. See the section "Default Parameters in the LINEQS Model" on page 1211 for details about the default parameters of the LINEQS model.

Modifying a LINEQS Model from a Reference Model

This section assumes that you use a REFMODEL statement within the scope of a MODEL statement and that the reference model (or base model) is a LINEQS model. The reference model is called the old model, and the model being defined is called the new model. If the new model is not intended to be an exact copy of the old model, you can use the extended LINEQS modeling language described in this section to make modifications within the scope of the MODEL statement for the new model.

The syntax of the extended LINEQS modeling language is the same as that of the ordinary LINEQS modeling language (see the section “LINEQS Statement” on page 1090):

> **LINEQS** < *equation* < , *equation* . . . > > **;**
> **VARIANCE** *partial-variance-parameters* **;**
> **COV** *covariance-parameters* **;**
> **MEAN** *mean-parameters* **;**

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

Duplication: If you do not specify in the new model an equation with an outcome variable (that is, a variable on the left side of the equal sign) that exists in the old model, the equation with that outcome variable in the old model is duplicated in the new model. For specifications other than the LINEQS statement, if you do not specify in the new model a parameter location that exists in the old model, the old parameter specification is duplicated in the new model.

Addition: If you specify in the new model an equation with an outcome variable that does not exist as an outcome variable in the equations of the old model, the equation is added in the new model. For specifications other than the LINEQS statement, if you specify in the new model a parameter location that does not exist in the old model, the new parameter specification is added in the new model.

Deletion: If you specify in the new model an equation with an outcome variable that also exists as an outcome variable in an equation of the 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, the following two-group analysis specifies Model 2 by referring to Model 1 in the REFMODEL statement:

```
proc calis;
   group 1 / data=d1;
   group 2 / data=d2;
   model 1 / group=1;
      lineqs
         V1   =       1 * F1   + E1,
         V2   =   load1 * F1   + E2,
         V3   =   load2 * F1   + E3,
         F1   =      b1 * V4   + b2 * V5 + b3 * V6 + D1;
      variance
         E1-E3  = ve1-ve3,
         D1     = vd1,
         V4-V6  = phi4-phi6;
      cov
         E1 E2 = cve12;
   model 2 / group=2;
      refmodel 1;
      lineqs
         V3   = load1 * F1 + E3;
      cov
         E1 E2 = .,
         E2 E3 = cve23;
   run;
```

Model 2 is the new model which refers to the old model, Model 1. This example illustrates the four types of model integration:

- Duplication: All equations, except the one with outcome variable V3, in the old model are duplicated in the new model. All specifications in the VARIANCE 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** = *var_list*

- **XVAR** | **XV** | **X** = *var_list*

- **ETAVAR** | **ETAV** | **ETA** = *var_list*

- **XIVAR** | **XIV** | **XI** | **KSIVAR** | **KSIV** | **KSI** = *var_list*

LISMOD stands for LISREL modeling, where LISREL is the program developed by Jöreskog and Sörbom (1988). Like the original implementation of LISREL, LISMOD uses a matrix specification interface. To complete the LISMOD specification, you might need to add as many MATRIX statements as needed, as shown in the following statement structure for the LISMOD model:

LISMOD *var_lists* **;**
  MATRIX *matrix-name parameters-in-matrix* **;**
  Repeat the MATRIX statement as needed **;**

The *matrix-name* in the MATRIX statement should be one of the twelve model matrices in LISMOD, as listed in the following:

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

See the section "Model Matrices in the LISMOD Model" on page 1214 for definitions of these matrices and their roles in the LISMOD modeling language. See the MATRIX statement on page 1111 for the details of parameter specification.

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

- YVAR= list is for manifest variables **y** that are directly related to the endogenous latent variables $\boldsymbol{\eta}$ (eta). Variables in the list are called **y**-variables.
- XVAR= list is for manifest variables **x** that are directly related to the exogenous latent variables $\boldsymbol{\xi}$ (xi or ksi). Variables in the list are called **x**-variables.
- ETAVAR= list is for endogenous latent variables $\boldsymbol{\eta}$. Variables in the list are called $\boldsymbol{\eta}$-variables.
- XIVAR= list is for exogenous latent variables $\boldsymbol{\xi}$. Variables in the list are called $\boldsymbol{\xi}$-variables.

The order of variables in the lists of the LISMOD statement is used to define the variable order in rows and columns of the LISMOD model matrices.

Depending on the model of interest, you might not need to specify all the lists of variables. When some variable lists are not specified, the full model reduces to specialized submodels. However, to be a proper submodel in the LISMOD modeling language, it is necessary (but not sufficient) that at least one of the YVAR= or XVAR= lists is defined. See the section "LISMOD Submodels" on page 1216 for the details about LISMOD submodels that PROC CALIS can handle.

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

```
proc calis;
   lismod xvar=x1-x3 yvar=y1-y6 xivar=xi etavar=eta1-eta2;
   matrix _LAMBDAY_   [,1]  = 1. load3 load4,
                      [,2]  = 0. 0. 0. 1. load5 load6;
   matrix _THETAY_    [1,1] = ey1-ey3,
                      [2,1] = cey;
   matrix _LAMBDAX_   [,]   = 1. load1 load2;
   matrix _THETAX_    [1,1] = 3*ex;
   matrix _GAMMA_     [,1]  = beta1 beta2;
   matrix _PHI_       [1,1] = phi;
   matrix _PSI_       [1,1] = psi1-psi2;
run;
```

In this example, you have three $\mathbf{x}$-variables x1–x3, six $\mathbf{y}$-variables y1–y6, one $\boldsymbol{\xi}$-variable xi, and two $\boldsymbol{\eta}$-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 square brackets '[' and ']' the starting row and column numbers of the first element to be parameterized. After the equal sign, you specify fixed or free parameters for the matrix elements.

Depending on how you specify the starting row and column numbers, the parameter specification might proceed differently. See the MATRIX statement on page 1111 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 proceeds 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 use similar notation to specify the parameters of a row. For example, with the notation [2,] for the starting row and column numbers, specification proceeds to the left with the same second row in the matrix.

If you have specified both starting row and column numbers, such as those in the first specification in matrix _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 [,], no starting row and column numbers are specified. Specification starts from the first valid element in the matrix and proceeds row-wise for all valid elements in the matrix. For example, in the matrix _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.

Default Parameters

It is important to understand the default parameters in the LISMOD model. First, if you know which parameters are default free parameters, you can make your specification more efficient by omitting some specifications. For example, because all variances and covariances among the exogenous $\boldsymbol{\xi}$-variables (excluding error terms) are free parameters by default, you do not need to specify them with MATRIX statement if these variances and covariances are not constrained. Second, if you know which parameters are default fixed zero parameters, you can specify your model accurately. For example, because all measurement errors in the LISMOD model are fixed zeros by default, you must use the MATRIX statement to specify the covariances among the errors in the $\boldsymbol{\Theta}_x$ (_THETAX_) or $\boldsymbol{\Theta}_y$ (_THETAY_) matrices if you want to fit a model with some correlated measurement errors. See the section "Default Parameters in the LISMOD Model" on page 1220 for details about the default parameters of the LISMOD model.

Modifying a LISMOD Model from a Reference Model

This section assumes that you use a REFMODEL statement within the scope of a MODEL statement and that the reference model (or base model) is also a LISMOD model. The reference model is called the old model, and the model that refers to this old model is called the new model. If the new model is not intended to be an exact copy of the old model, you can use the extended LISMOD modeling language described in this section to make modifications within the scope of the MODEL statement for the new model. The syntax is similar to, but not exactly the same as, the ordinary LISMOD modeling language (see the section "LISMOD Statement" on page 1097). The respecification syntax for a LISMOD model is shown as follows:

> **LISMOD** ;
>   **MATRIX** *matrix-name parameters-in-matrix* ;
>   ***Repeat the MATRIX statement as needed*** ;

First, in the respecification you should not put any variable lists in the LISMOD statement. The reason is that the parameter respecifications in the new model refer to the variable lists of the old model. Therefore, the variable lists in the new model are implicitly assumed to be exactly the same as those in the old model. Because of this, the LISMOD statement is entirely optional for the respecification in the new model.

Second, you can use MATRIX *matrix-name* statements to modify the old model by using the same syntax as in the LISMOD modeling language. The *matrix-name* can be one of the twelve possible LISMOD matrices. In addition, in the respecification syntax you can use the missing value '.' to drop a parameter specification from the old model.

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

Duplication: If you do not specify in the new model a parameter location that exists in the old model, the old parameter specification is duplicated in the new model.

Addition: If you specify in the new model a parameter location that does not exist in the old model, the new parameter specification is used in the new model.

Deletion: If you specify in the new model a parameter location that also exists in the old model and the new parameter is denoted by the missing value '.', the old parameter specification is not copied into the new model.

Replacement: If you specify in the new model a parameter location that also exists in the old model and the new parameter is not denoted by the missing value '.', the new parameter specification replaces the old one in the new model.

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

```
proc calis;
   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 for the [2,1] element in matrix _THETAY_ and the diagonal of matrix _THETAX_, which are modified in the new model.
- Addition: The _BETA_[2,1] parameter location is added with a new parameter beta in the new model. This indicates that *eta1* is a predictor variable of *eta2* in the new model, but not in the old model.
- Deletion: Because the missing value '.' is used for the parameter value, the _THETAY_[2,1] parameter location is no longer defined as a free parameter in the new model. In the old model, the same location is defined by the free parameter cey.
- Replacement: The diagonal elements of the _THETAX_ matrix in the new model are now defined by three distinct parameters ex1–ex3. This replaces the old specification where a single constrained parameter ex is applied to all the diagonal elements in the _THETAX_ matrix.

LMTESTS Statement

LMTESTS | LMTEST *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 CALIS statement so that the options will apply to all models. It can also be used within the scope of each MODEL statement so that the options will apply only locally. Therefore, different models within a CALIS run can have very different LMTESTS *options*.

The LM Tests Display Options

The following are the *display-options* for the LM tests:

DEFAULT

conducts the default sets of LM tests for freeing fixed parameters in the model. This option is used when you need to reset the default sets of LM tests in the local model. For example, you might have turned off the default LM tests by using the NODEFAULT option in the LMTESTS statement within the scope of PROC CALIS statement. However, for the model under the scope of a particular MODEL statement, you can use this DEFAULT option in the local LMTESTS statement to turn on the default LM tests again.

MAXRANK

sets the maximum number of rankings within a set of LM tests. The actual number of test rankings might be smaller because the number of possible LM tests within a set might be smaller than the maximum number requested.

NODEFAULT

turns off the default sets of LM tests for freeing fixed parameters in the model. As a result, only the customized LM tests defined in the *test-sets* of the LMTESTS statement are conducted and displayed. Note that the LM tests for equality and active boundary constraints are not turned off by this option. If you specify this option in the LMTESTS statement within the scope of the PROC CALIS statement, it will propagate to all models.

NORANK

turns off the ranking of the LM tests. Ranking of the LM tests is done automatically when the model modification indices are requested. The NORANK option is ignored if you also set the MAXRANK option.

LMMAT

prints the sets of LM tests in matrix form, in addition to the normal LM test results.

The Customized Sets of LM Tests: Syntax of the Test-sets

In addition to the *display-options*, you can define customized sets of LM tests as *test-sets* in the LMTESTS statement. You can define as many *test-sets* as you like. Ranking of the LM tests will be done individually for each *test-set*. For example, the following LMTESTS statement requests that the default sets of LM tests not be conducted by the NODEFAULT option. Instead, two customized *test-sets* are defined.

```
lmtests nodefault MyFirstSet=[ALL] MySecondSet=[COVEXOG COVERR];
```

The first customized set MyFirstSet pulls all possible parameter locations together for the LM test ranking (ALL keyword). The second customized set MySecondSet pulls only the covariances among exogenous variables (COVEXOG keyword) and among errors (COVERR keyword) together for the LM test ranking.

Two different kinds of *regions* for LM tests are supported in PROC CALIS: matrix-based or non-matrix-based.

The matrix-based *regions* can be used if you are familiar with the matrix representations of various types of models. Note that defining *test-sets* by using matrix-based *regions* does not mean that LM tests are printed in matrix format. It means only that the parameter locations within the specified matrices are included into the specific *test-sets* for LM test ranking. For matrix output of LM tests, use the LMMAT option in the LMTESTS statement.

Non-matrix-based *regions* do not assume the knowledge of the model matrices. They are easier to use in most situations. In addition, non-matrix-based *regions* can cover special subsets of parameter locations that cannot be defined by model matrices and submatrices. For example, because of the compartmentalization according to independent and dependent variables in the LINEQS model matrices, the sets of LM tests defined by the LINEQS matrix-based *regions* are limited. For example, you cannot use any matrix-based *regions* to request LM tests for new paths to existing independent variables in the LINEQS model. Such a matrix does not exist in the original specification. However, you can use the non-matrix based *region* NEWENDO to refer to these new paths.

The *regions* for parameter locations are specified by keywords in the LMTESTS statement. Because the *regions* are specific to the types of models, they are described separately for each model type in the following.

The LM Test Regions for COSAN Models

ALLMAT
specifies all parameter locations in all matrices.

CENTRAL
specifies all parameter locations in the central covariance matrices in all terms.

MATRIX | MAT | MATSET = [*set-of-matrices***] | {***set-of-matrices***}**
specifies the parameter locations in the matrices specified in *set-of-matrices*.

MEANVEC
specifies all parameter locations in the central mean vectors in all terms.

OUTER
specifies all parameter locations in all matrices except for the central covariance matrices and central mean vectors in all terms.

The LM Test Regions for FACTOR Models

The keywords for the matrix-based regions are associated with the FACTOR model matrices. See the section "Summary of Matrices in the FACTOR Model" on page 1202 for the definitions and properties of these matrices.

Keywords for Matrix-Based Regions

FACTERRV | FACTERRV
specifies the error variances.

FACTFCOV | FACTFCOV
specifies the covariances among factors.

FACTINTE | FACTINTE
specifies the intercepts.

FACTLOAD | FACTLOAD
specifies the factor loadings.

FACTMEAN | FACTMEAN
specifies the factor means.

Keywords for Non-Matrix-Based Regions

ALL
specifies all parameter locations.

COV
specifies the covariances among factors.

COVERR
specifies the covariances among errors.

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

The keywords for the matrix-based regions are associated with the LINEQS model matrices. See the section "Matrix Representation of the LINEQS Model" on page 1207 for definitions of these model matrices and see the section "Summary of Matrices and Submatrices in the LINEQS Model" on page 1209 for the names and properties and the model matrices and submatrices.

EQSALPHA | EQSALPHA
specifies the intercepts of dependent variables.

EQSBETA | EQSBETA
specifies effects of dependent variables on dependent variables.

EQSGAMMA | _EQSGAMMA_SUB_ | EQSGAMMA | EQSGAMMASUB
specifies the effects of independent variables (excluding errors) on dependent variables. Because effects of errors on dependent variables are restricted to ones in the LINEQS model, LM tests on _EQSGAMMA_ and _EQSGAMMA_SUB_ (submatrix of _EQSGAMMA_) are the same.

EQSNU | _EQSNU_SUB_ | EQSNU | EQSNUSUB
specifies the means of independent variables (excluding errors). Because means of errors are restricted to zero in the LINEQS model, LM tests on _EQSNU_ and _EQSNU_SUB_ (submatrix of _EQSNU_) are the same.

EQSPHI | EQSPHI
specifies variances and covariances among all independent variables, including errors.

EQSPHI11 | EQSPHI11
specifies variances and covariances among independent variables, excluding errors.

EQSPHI21 | EQSPHI21
specifies covariances between errors and disturbances with other independent variables.

EQSPHI22 | EQSPHI22
specifies variances and covariances among errors and disturbances.

Keywords for Non-Matrix-Based Regions

ALL
specifies all possible parameter locations.

COV
specifies all covariances among independent variables, including errors and disturbances.

COVERR
specifies covariances among errors or disturbances.

COVEXOG
specifies covariances among independent variables, excluding errors and disturbances.

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

The keywords for the matrix-based regions are associated with the LISMOD model matrices. See the section "Model Matrices in the LISMOD Model" on page 1214 for the definitions and properties of these matrices.

Keywords for Matrix-Based Regions

ALPHA | ALPHA
specifies the _ALPHA_ matrix.

BETA | BETA
specifies the _BETA_ matrix.

GAMMA | GAMMA
specifies the _GAMMA_ matrix.

KAPPA | KAPPA
specifies the _KAPPA_ matrix.

LAMBDA | LAMBDA
specifies the _LAMBDAX_ and _LAMBDAY_ matrices.

LAMBDAX | LAMBDAX
specifies the _LAMBDAX_ matrix.

LAMBDAY | LAMBDAY
specifies the _LAMBDAY_ matrix.

_NU_ | NU
specifies the _NUX_ and _NUY_ matrices.

NUX | NUX
specifies the _NUX_ matrix.

NUY | NUY
specifies the _NUY_ matrix.

PHI | PHI
specifies the _PHI_ matrix.

PSI | PSI
specifies the _PSI_ matrix.

THETA | THETA
specifies the _THETAX_ and _THETAY_ matrices.

THETAX | THETAX
specifies the _THETAX_ matrix.

THETAY | THETAY
specifies the _THETAY_ matrix.

Keywords for Non-Matrix-Based Regions

ALL
specifies all model matrices.

COV
specifies all covariance parameters in _THETAY_, _THETAX_, _PHI_, and _PSI_.

COVERR
specifies all covariances for errors or disturbances in _THETAY_, _THETAX_, and _PSI_.

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

The keywords for the matrix-based regions are associated with the MSTRUCT model matrices. See the section "Model Matrices in the MSTRUCT Model" on page 1221 for the definitions and properties of these matrices.

Keywords for Matrix-Based Regions

MSTRUCTCOV | _COV_ | MSTRUCTCOV
specifies the _MSTRUCTCOV_ or _COV_ matrix.

MSTRUCTMEAN | _MEAN_ | MSTRUCTMEAN
specifies the _MSTRUCTMEAN_ or _MEAN_vector.

Keywords for Non-Matrix-Based Regions

ALL
specifies the _MSTRUCTCOV_ (or _COV_) and _MSTRUCTMEAN_ (or _MEAN_) matrices.

COV
specifies the _MSTRUCTCOV_ or _COV_ matrix.

MEANS | MEAN
specifies the _MSTRUCTMEAN_ or _MEAN_ matrix.

The LM Test Regions for PATH and RAM Models

The keywords for the matrix-based regions are associated with the submatrices of the RAM model matrices. See the section "Partitions of the RAM Model Matrices and Some Restrictions" on page 1232 for the definitions of these submatrices and the section "Summary of Matrices and Submatrices in the RAM Model" on page 1233 for the summary of the names and properties of these submatrices.

Keywords for Matrix-Based Regions

RAMA | _A_ | RAMA
specifies the _RAMA_ matrix.

_RAMA_LEFT_ | _A_LEFT_ | RAMALEFT
specifies the left portion of the _RAMA_ matrix.

_RAMA_LL_ | _A_LL_ | RAMALL
specifies the lower left portion of the _RAMA_ matrix.

_RAMA_LR_ | _A_LR_ | RAMALR
specifies the lower right portion of the _RAMA_ matrix.

_RAMA_LOWER_ | _A_LOWER_ | RAMALOWER
specifies the lower portion of the _RAMA_ matrix. This is equivalent to the region specified by the NEWENDO keyword.

_RAMA_RIGHT_ | _A_RIGHT_ | RAMARIGHT
specifies the right portion of the _RAMA_ matrix.

_RAMA_UPPER_ | _A_UPPER_ | RAMAUPPER
specifies the upper portion of 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 . . .* >>> **;**

MATRIX statement specifies the matrix elements (locations) and their parameters. Parameters can be fixed or free, with or without initial estimates. The *matrix-name* indicates the matrix to specify in the MATRIX statement. The *location* indicates the starting row and column numbers of the matrix being specified and the *parameter-spec* is a list of free or fixed parameters for the elements that are indicated by the *location*.

The MATRIX statement is a subsidiary model specification statement of the COSAN, LISMOD, and MSTRUCT modeling languages. You might need to use the MATRIX statements as many times as needed for specifying your model. However, you can use the MATRIX statement at most once for each distinct model matrix.

Valid Matrix Names for the COSAN Model

The valid *matrix-names* depend on the your specification in the COSAN statement in which you define the COSAN model matrices and their properties. Except for those fixed matrices with the IDE or ZID type, you can use the MATRIX statement to specify any COSAN model matrices you define in the COSAN statement.

Valid Matrix Names for the LISMOD Model

There are 12 model matrices in the LISMOD model, and they correspond to the following valid *matrix-names* :

- matrices and their types in the measurement model for the **y**-variables

  LAMBDAY the matrix of regression coefficients of the **y**-variables on the $\boldsymbol{\eta}$-variables (general, GEN)

  NUY the vector of intercept terms of the **y**-variables (general, GEN)

  THETAY the error covariance matrix for the **y**-variables (symmetric, SYM)

- matrices and their types in the measurement model for the **x**-variables

  LAMBDAX the matrix of regression coefficients of the **x**-variables on the $\boldsymbol{\xi}$-variables (general, GEN)

  NUX the vector of intercept terms of the **x**-variables (general, GEN)

  THETAX the error covariance matrix for the **x**-variables (symmetric, SYM)

- matrices and their types in the structural model

  ALPHA the vector of intercept terms of the $\boldsymbol{\eta}$-variables (general, GEN)

  BETA the matrix of regression coefficients of the $\boldsymbol{\eta}$-variables on the $\boldsymbol{\eta}$-variables (general, GEN)

GAMMA	the matrix of regression coefficients of the η-variables on the ξ-variables (general, GEN)
KAPPA	the mean vector for the ξ-variables (general, GEN)
PHI	the covariance matrix for the ξ-variables (symmetric, SYM)
PSI	the error covariance matrix for the η-variables (symmetric, SYM)

Valid Matrix Names for the MSTRUCT Modeling Language

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

COV	the covariance matrix (symmetric, SYM)
MEAN	the mean vector (general, GEN)

Specifying Locations in Model Matrices

The five main types of matrix *locations* (elements) specification in the MATRIX statement are briefly described in the following:

- Unspecified *location*: Blank or `[ , ]`
  Use this notation to specify the `[1,1]` element of the matrix, and to specify the remaining *valid* elements of the matrix in a prescribed order until all the parameters in the *parameter-spec* list are assigned.

- Row-and-column *location*: `[i,j]`, `[@i,j]`, `[i,@j]`, or `[@i,@j]`
  Use this notation to specify the `[i,j]` element of a matrix, and to specify the remaining elements of the matrix in the order indicated by the *location* notation until all the parameters in the *parameter-spec* list are assigned.

- Row *location* only: `[i,]`, `[@i,]`, or `[`*iset*`,]`
  Use this notation to specify the first *valid* matrix element in the `[i]`-th row (for the first two notations) or the `[i1]`-th row (for the `[`*iset*`,]` notation, where *iset*=(`i1`, `i2`, . . .) is a set of row numbers), and to specify the remaining elements of the matrix in the order indicated by the *location* notation until all the parameters in the *parameter-spec* list are assigned.

- Column *location* only: `[,j]`, `[,@j]`, or `[,`*jset*`]`
  Use this notation to specify the first *valid* matrix element in the `[j]`-th column (for the first two notations) or the `[j1]`-th column (for the `[,`*jset*`]` notation, where *jset*=(`j1`, `j2`, . . .) is a set of columns), and to specify the remaining elements of the matrix in the order indicated by the *location* notation until all the parameters in the *parameter-spec* list are assigned.

- Row-and-column-sets *location*: `[`*iset*`,`*jset*`]`, `[`*iset*`,j]`, or `[i,`*jset*`]`
  Use this notation to specify the `[i1,j1]` element of the matrix, where `i1` is either the same as `i` or the first row number specified in *iset*, and `j1` is either the same as `j` or the first column number specified in *jset*, and to specify the remaining elements of the matrix in the order indicated by the *location* notation until all the parameters in the *parameter-spec* list are assigned.

Consider the following points about the *location* specifications:

- In the description of the various *location* specifications, the starting matrix element for parameter assignment is relatively well-defined. However, if the *parameter-spec* list has more than one parameter, there are more matrix elements to assign with the parameters in the *parameter-spec* list. If there is no *parameter-spec* list, a set of matrix elements are specified as unnamed free parameters. Hence, the actual number of elements specified by these *location* specifications depends on the length of the *parameter-spec* list.
- Because more than one matrix element could be specified in any of these *location* specifications, it is important to understand the order that PROC CALIS uses to assign the matrix elements.
- In some of the *location* specifications, either the row or column is unspecified and the assignment of the matrix element starts with the first *valid* element given the column or the row number. This first valid element depends on the type of the matrix in question.

The next few sections describe the parameter assignments in more detail for each of these *location* specifications in the MATRIX statement.

Unspecified Location: Blank or `[ , ]`

This notation means that all valid elements started with the **[1,1]** element of the matrix specified in the model. If no *parameter-spec* list is specified, all valid elements in the matrix are unnamed free parameters. For these elements, PROC CALIS generates parameter names with the _Parm prefix followed by a unique integer (for example, _Parm1, _Parm2, and so on). If a *parameter-spec* list is specified, the assignment of parameters starts with the **[1,1]** element and proceeds to the next *valid* elements in the same row. If the entire row of valid elements is assigned with parameters, it proceeds to the next row and so on, until all the parameters in the *parameter-spec* list are assigned. The valid element given the row or column number depends on the type of matrix in question. The following examples illustrate the usage of the unspecified *location* notation.

Suppose that _GAMMA_ is a general 3×3 matrix. The following statement specifies four elements of this matrix:

```
matrix _GAMMA_ [,] = gg1-gg4;
```

Equivalently, you can use the following blank *location* specification:

```
matrix _GAMMA_ = gg1-gg4;
```

Both specifications are equivalent to the following elementwise specification:

```
matrix _GAMMA_ [1,1] = gg1,
               [1,2] = gg2,
               [1,3] = gg3,
               [2,1] = gg4;
```

With the unspecified *location* for the matrix _GAMMA_, the first row is filled up with the parameters first. Then it proceeds to the next row and so on until all parameters in the *parameter-spec* list are assigned. Because there are four parameters and _GAMMA_ has three columns, the parameter gg4 is assigned to the _GAMMA_[2, 1] element.

However, if the preceding specification is for a 3×3 matrix symmetric matrix _PHI_, the parameters are assigned differently. That is, the following specification has different matrix elements assigned with the parameters:

```
matrix _PHI_ = gg1-gg4;
```

Because symmetric matrices contain redundant elements, parameters are assigned only to the lower triangular elements (including the diagonal elements). As a result, the following elementwise specification reflects the preceding specification of matrix _PHI_:

```
matrix _PHI_ [1,1] = gg1,
             [2,1] = gg2,
             [2,2] = gg3,
             [3,1] = gg4;
```

The case for lower triangular matrices is the same as the case for symmetric matrices. That is, only the lower triangular elements are valid elements for the parameter assignments.

For upper triangular matrices, only the upper triangular elements (including the diagonal elements) are valid for the parameter assignments. For example, consider the following specification of a 3×3 upper triangular matrix UPP:

```
matrix UPP = gg1-gg4;
```

The matrix elements assigned with the parameters are the same as the following elementwise specification:

```
matrix UPP [1,1] = gg1,
           [1,2] = gg2,
           [1,3] = gg3,
           [2,2] = gg4;
```

If a 4×4 diagonal matrix is specified by the preceding MATRIX statement, the parameters are assigned to the following elements: `[1,1]`, `[2,2]`, `[3,3]`, and `[4,4]`.

Lastly, if there is no *parameter-spec* list for the unspecified *location* notation, all valid parameters in the matrix being specified are unnamed free parameters. For example, if **A** is a 4×4 general rectangular matrix, the following specification assigns 16 unnamed free parameters to all of the elements in **A**:

```
matrix A [,];
```

PROC CALIS generates parameters _Parm1, _Parm2, ..., _Parm16 to the elements `[1,1]`, `[1,2]`, `[1,3]`, ..., `[4,3]`, `[4,4]`, respectively.

However, if **S** is a 4×4 *symmetric* matrix, the following specification assigns only 10 unnamed free parameters to the lower triangular elements of **S**:

```
matrix S;
```

PROC CALIS generates parameters _Parm1, _Parm2, ..., _Parm10 to the elements `[1,1]`, `[2,1]`, `[2,2]`, ..., `[4,3]`, `[4,4]`, respectively.

Row-and-Column Location: [i,j], [@i,j], [i,@j], or [@i,@j]

All these notations provide the starting row (i) and column (j numbers for the assignment of the parameters in the *parameter-spec* list. The notations are different in the way they proceed to the next element in the matrix. If no *parameter-spec* list is specified, only the single element [i,j] is an unnamed free parameter. For this [i,j] element, PROC CALIS generates a parameter name with the _Parm prefix followed by a unique integer (for example, _Parm1). If a *parameter-spec* list is specified, the assignment of parameters starts with the [i,j] element and proceeds to next element until all the parameters in the *parameter-spec* list are assigned. The following summarizes how the assignment of parameter proceeds, depending on the uses of the @ sign before the starting row or column number:

- [i,j] specifies the [i,j] element, and proceeds to [i+1,j+1], [i+2,j+2], and so on.
- [@i,j] specifies the [i,j] element, and proceeds to [i,j+1], [i,j+2], [i,j+3], and so on.
- [i,@j] specifies the [i,j] element, and proceeds to [i+1,j], [i+2,j], [i+3,j], and so on.
- [@i,@j] specifies the [i,j] element only.

The following examples illustrates the usage of the row-and-column *location* notation.

The simplest case is the specification of a single element as an unnamed free parameter. For example, the following statement specifies that [1,4] in matrix **A** is an unnamed free parameter:

```
matrix A [1,4];
```

PROC CALIS generates a parameter name with the _Parm prefix for this element. In this case, using the @ sign before the row or column number is optional. That is, the following statements are all the same specification:

```
matrix A [1,4];
matrix A [@1,4];
matrix A [1,@4];
matrix A [@1,@4];
```

You can specify more than one unnamed free parameter by using multiple *location* specifications, as shown in the following example:

```
matrix A [1,4],[3,5];
```

Elements [1,4] and [3,5] of matrix **A** are both unnamed free parameters. However, when a *parameter-spec* list is specified after the *location*, more than one parameters might be specified. The use of the @ determines how the elements in the matrix are assigned with the parameters in the *parameter-spec* list. The following examples illustrate this under various situations.

For example, consider the following specification of a 4×4 matrix general matrix **A**:

```
matrix A
   [1,1] = a b c;
```

The three parameters a, b, and c, are assigned to the matrix elements [1,1], [2,2], and [3,3], respectively. That is, this specification is equivalent to the following elementwise specification:

```
matrix A
   [1,1] = a ,
   [2,2] = b ,
   [3,3] = c ;
```

However, with the @ sign, the assignment is different. For example, consider the @ sign attached to the row number in the following specification:

```
matrix A
   [@1,1] = a b c;
```

The @ sign fixes the row number to 1. As a result, this specification is equivalent to the following elementwise specification:

```
matrix A
   [1,1] = a ,
   [1,2] = b ,
   [1,3] = c ;
```

Using the @ sign before the column number fixes the column number. For example, consider the following specification of matrix **A**:

```
matrix A
   [2,@2] = a b c;
```

The @ sign fixes the column number to 2. As a result, this specification is equivalent to the following elementwise specification:

```
matrix A
   [2,2] = a ,
   [3,2] = b ,
   [4,2] = c ;
```

If you put the @ sign in both of the row and column numbers, only one element is intended to be assigned. For example, the following specification means that only $\mathbf{A}[2,3]$ is assigned with the parameter a:

```
matrix A
   [@2,@3] = a;
```

But you could specify this simply as the statement without the @ sign:

```
matrix A
   [2,3] = a;
```

Notice that the matrix type does not play a role in determining the elements for the parameter assignments in the row-and-column *location* specification. You have to make sure that the parameters are assigned in the valid elements of the matrix. For example, suppose that $\mathbf{S}$ is a 4×4 symmetric matrix and you specify the following statement for its elements:

```
matrix A
   [@3,2] = a b c;
```

The elements to be assigned with the parameters a, b, and c, are `[3,2]`, `[3,3]`, and `[3,4]`, respectively. However, because **S** is symmetric, you can specify only the nonredundant elements in the lower triangular of **S**. Hence, the specification of the `[3,4]` element is not valid and it generates an error.

Row Location Only: `[i,]`, `[@i,]`, or `[`*iset*`,]`

All these notations provide the starting row `[i1,]` for the assignment of the parameters in *parameter-spec*, where `i1` is `i` for the first two *location* notations or `i1` is the first row specified in *iset*, where *iset* = (`i1`, `i2`, ...) is a set of row numbers. Because no column location is specified, the starting element is the first valid element in the `i1`-th row of the matrix.

If no *parameter-spec* list is specified, all the valid elements in the entire `i1`-th row of the matrix are unnamed free parameters. If a set of row numbers is specified in *iset*, all the valid elements in the all the rows specified in *iset* are unnamed free parameters.

If a *parameter-spec* list is specified, the assignment of parameters starts with the first valid elements of the `i1`-th row. The assignment proceeds to next valid elements in the same row. The `[i,]` specification proceeds row by row for parameter assignment while the `[@i,]` specification stays at the same `i`-th row. The `[`*iset*`,]` specification indicates and limits the sequence of rows to be assigned with the parameter in the *parameter-spec* list. The assignment stops when all the parameters in the *parameter-spec* list are assigned. The following summarizes how the assignment of parameters proceeds in more precise terms:

- `[i,]` specifies the first valid element in row `i` and proceeds to the valid elements in rows `i`, `i+1`, `i+2`, ..., until all parameters in the *parameter-spec* list are assigned.
- `[@i,]` specifies the first valid elements in row `i` and proceeds to the valid elements in the *same* row until all parameters in the *parameter-spec* list are assigned.
- `[`*iset*`,]` specifies the first valid elements in row `i1`, where `i1`, `i2`, ... are the rows specified in *iset*. It proceeds to the valid elements in rows `i1`, `i2`, ..., until all parameters in the *parameter-spec* list are assigned.

The following examples illustrates the usage of the row *locations*.

The simplest case is the specification of all valid elements of a single row as unnamed free parameters. For example, the following specification of a 3×3 rectangular matrix **A** assigns unnamed free parameters to all the elements in the second row of matrix **A**:

```
matrix A [2,];
```

PROC CALIS generates parameter names with the _Parm prefix for these elements. For example, the `[2,1]`, `[2,2]`, and `[2,3]` elements are named with _Parm1, _Parm2, and _Parm3, respectively.

Using the `@` sign before the row number in this case is optional. That is, the following statement is the same specification:

```
matrix A [@2,];
```

If you specify a set of row numbers without the *parameter-spec* list, all valid elements of the specified rows are unnamed free parameters. For example, consider the following specification of a 6×6 symmetric matrix **S**:

```
matrix S [1 3 5,];
```

This specification specifies unnamed free parameters for the lower triangular elements in the first, third, and fifth rows of matrix **S**. It is equivalent to the following specification:

```
matrix S [1,],
         [3,],
         [5,];
```

As a result, this means that the following elements in matrix **S** are free parameters: `[1,1]`, `[3,1]`, `[3,2]`, `[3,3]`, `[5,1]`, `[5,2]`, `[5,3]`, `[5,4]`, and `[5,5]`. Notice that only the elements in the lower triangular of those specified rows in **S** are free parameters. This shows that parameter assignment with the row *location* notation depends on the matrix type.

With the use of the *parameter-spec* list, the parameter assignment with the row *location* notation stops when all the parameters are assigned. For example, consider the following specification of a 4×4 *general* (rectangular) matrix **A**:

```
matrix A
   [2,] = a b c;
```

The three parameters a, b, and c, are assigned to the matrix elements `[2,1]`, `[2,2]`, and `[2,3]`, respectively. However, a different assignment of the parameters applies if you use the same specification for a 4×4 *symmetric* matrix **S**, as shown in the following statement:

```
matrix S
   [2,] = a b c;
```

Because there are redundant elements in a symmetric matrix, you can specify only the lower triangular elements. Therefore, the row *location* specification is equivalent to the following elementwise specification:

```
matrix S
   [2,1] = a ,
   [2,2] = b ,
   [3,1] = c ;
```

When all the valid row elements are assigned with the parameters, the assignment proceeds to the next row. This is why the last parameter assignment is for **S**[3, 1]. The same assignment sequence applies to matrices with the lower triangular type (LOW).

For matrices with the upper triangular matrix type (UPP), only the elements in the upper triangular are assigned. For example, consider a 4×4 upper triangular matrix **U** with the following row *location* specification:

```
matrix U
   [2,] = a b c d;
```

The assignment of parameters is the same as the following elementwise specification:

```
matrix U
   [2,2] = a ,
   [2,3] = b ,
   [2,4] = c ,
   [3,3] = d;
```

The first valid element in the second row of the **U** matrix is **U**[2, 2]. Because all the valid elements in the second row are assigned with parameters, the last element has to go to the valid element in the next row. Hence, the parameter d is assigned to **U**[3, 3].

For matrices with the diagonal matrix type (DIA), only the diagonal elements are assigned. For example, consider a 4 × 4 upper diagonal matrix **D** with the following row *location* specification:

```
matrix D
   [2,] = a b c;
```

The assignment of parameters is the same as the following elementwise specification:

```
matrix D
   [2,2] = a ,
   [3,3] = b ,
   [4,4] = c ;
```

If you use an @ sign before the row number in the row *location* specification, the row number cannot move—it cannot proceed to the next row even if the valid elements in that row are already filled with the parameters in *parameter-spec*. All other behavior of the **[@i,]** specification is the same as that of the **[i,]** specification. For example, consider the following specification of a 4 × 4 *general* (rectangular) matrix **A**:

```
matrix A
   [@2,] = a b c d;
```

The four parameters a, b, c, and d, are assigned to the matrix elements **[2,1]**, **[2,2]**, **[2,3]**, and **[2,4]**, respectively. This is exactly the same result as the following specification without the @ sign:

```
matrix A
   [2,] = a b c d;
```

Here, all the elements of the second row of matrix **A** are assigned with elements. However, if one more parameter is specified in the *parameter-spec* list, the behavior for the two types of row *location* specifications are different. The following specification without the @ sign proceeds to the next row for the last parameter:

```
matrix A
   [2,] = a b c d e;
```

That is, the parameter e is assigned to the **A**[3, 1] element. However, the following specification *with* the @ sign results in an out-of-bound error:

```
matrix A
   [@2,] = a b c d e;
```

The out-of-bound error is due to the fact that the row number must be fixed so that the parameter e is forced to be assigned to **A**[2, 5], which does not exist.

However, the distinction between the row *location* specifications with and without the @ sign is not very important in common practice because in most cases you do not want the parameter assignment to proceed row after row automatically with a long list of parameters. For example, consider the following specification of a 4 × 4 *symmetric* matrix **S**:

```
matrix S
   [2,] = s21 s22 s31 s32 s33 s41 s42 s43;
```

This specification is equivalent to the following specification:

```
matrix S
   [2,] = s21 s22,
   [3,] = s31 s32 s33,
   [4,] = s41 s42 s43;
```

Although this specification is not as concise as the preceding one, it specifies more clearly about how parameters are assigned to each of the three rows of the **S** matrix. In this specification, you make sure that each of the three row *location* specifications has just enough parameters for the given row without proceeding to the next row for additional parameter assignments. With this kind of "careful" row *location* specifications, you do not need to use the `@` sign before the row numbers at all.

The last type of row *location* specification is the `[`*iset*`,]` notation, where *iset* means a set of row numbers. This specification type provides the set of row numbers for the assignment of the parameters in the *parameter-spec* list. For example, consider the following specification of a 4×4 *general* matrix **A**:

```
matrix A
   [2 4,] = a21 a22 a23 a24 a41 a42 a43 a44;
```

This specification is equivalent to the following statement with two row *location* specifications:

```
matrix A
   [2,] = a21 a22 a23 a24,
   [4,] = a41 a42 a43 a44;
```

In other words, the assignment of parameters follows the order of rows provided in the *iset*. Notice that the *iset* notation merely provides the order of rows to be assigned with the parameters in the *parameter-spec* list; it is not an error if you provide a shorter parameter list than that of the total number of elements in the rows. For example, the following specification of a 4×4 *general* matrix **A** is valid:

```
matrix A
   [2 4,] = a21 a22 a23 a24;
```

This specification has the same results as the following statement with one row *location*:

```
matrix A
   [2,] = a21 a22 a23 a24;
```

However, a valid specification does not mean it is a good representation of the problem. Providing more rows in the *iset* specification than intended is simply not a good practice.

Although a shorter *parameter-spec* list is acceptable, a longer list results in an error. For example, the following specification of a 4 × 4 *symmetric* matrix **S** results in an error:

```
matrix S
   [2 to 3,] = s21 s22 s31 s32 s33 extra1 extra2;
```

The `[2 to 3,]` not only gives the sequence of the rows for the parameter assignment, it also limits the set of rows to assign. Because matrix **S** is symmetric and because only the second and the third rows are supposed to be assigned with the *iset* specification, the parameters extra1 and extra2 are excessive.

Column Location Only: `[,j]`, `[,@j]`, or `[,`*jset*`]`

These notations mirror that of the row *location* notations. Instead of the rows being specified, the columns are specified by these notations. Therefore, you can understand the column *location* notations the same way as the row *location* notations.

All these column *location* notations provide the starting column `[,j1]` for the assignment of the parameters in *parameter-spec*, where `j1` is `j` for the first two *location* notations or `j1` is the first column specified in *jset*, where *jset* = (`j1`, `j2`, ...) is a set of column numbers. Because no row location is specified, the starting element is the first valid element in the `j1`-th column of the matrix.

If no *parameter-spec* list is specified, all the valid elements in the entire `j1`-th column of the matrix are unnamed free parameters. If a set of column numbers is specified in *jset*, all the valid elements in the all the columns specified in *jset* are unnamed free parameters.

If a *parameter-spec* list is specified, the assignment of parameters starts with the first valid elements of the `j1`-th column. The assignment proceeds to next valid elements in the same column. The `[,j]` specification proceeds column by column for parameter assignment while the `[,@j]` specification stays at the same `j`-th column. The `[,jset]` specification indicates and limits the sequence of columns to be assigned with the parameter in the *parameter-spec* list. The assignment stops when all the parameters in the *parameter-spec* list are assigned. The following list summarizes how the assignment of parameters proceeds:

- `[,j]` specifies the first valid element in column `j` and proceeds to the valid elements in columns `j`, `j+1`, `j+2`, ..., until all parameters in the *parameter-spec* list are assigned.
- `[,@j]` specifies the first valid elements in column `j` and proceeds to the valid elements in the *same* column until all parameters in the *parameter-spec* list are assigned.
- `[,`*jset*`]` specifies the first valid elements in column `j1`, where `j1`, `j2`, ... are the columns specified in *jset*. It proceeds to the valid elements in columns `j1`, `j2`, ..., until all parameters in the *parameter-spec* list are assigned.

See the section "Row Location Only: `[i,]`, `[@i,]`, or `[`*iset*`,]`" on page 1117 for examples, which are applicable to the usage of the column *locations*.

Row-and-Column-Sets Location: [iset,jset], [iset,j], or [i,jset]

These notations specify the sets of row and column elements for the assignment of the parameters in the *parameter-spec* list. In the first notation, you specify the set of row numbers in *iset* = (**i1**, **i2**, ...), and the set of column numbers in *jset* = (**j1**, **j2**, ...). The last two notations are special cases of the first notation. The **[iset,j]** notation specifies only one column with *jset* = **j1** = **j**. The **[i,jset]** notation specifies only one row with *iset* = **i1** = **i**. For the last two notations, adding the **@** sign before **j** or **i** is optional. In general, the row-and-column-sets *locations* specify the matrix elements in the following order:

[i1,j1], **[i1,j2]**, ...,
[i2,j1], **[i2,j2]**, ...,
[i3,j1], **[i3,j2]**, ...,
..., ..., ...,
[ir,j1], **[ir,j2]**, ..., **[ir,js]**

where **r** represents the number of rows in the *iset* and **s** represents the number of columns in the *jset*. Note that this ordering of elements does not necessarily mean that all these elements are specified. The number of elements specified depends on the length of the *parameter-spec* list.

If no *parameter-spec* list is specified after the *location* notation, all the **r** × **s** elements specified in the *iset* and *jset* are unnamed free parameters. PROC CALIS generates parameter names with the _Parm prefix for these elements.

If a *parameter-spec* list is specified after the *location* notation, the total number of matrix elements that are assigned with the parameters is the same as the number of parameter specifications in the *parameter-spec* list.

The following examples illustrates the usage of the row-and-column-sets *locations*.

The simplest case is the specification of all elements in the *iset* and *jset* as free parameters, as shown in the following statement:

```
matrix _Gamma_ [2 3,4 1];
```

This means that _Gamma_[2, 4], _Gamma_[2, 1], _Gamma_[3, 4], and _Gamma_[3, 1] are all free parameters in the matrix. For these elements, PROC CALIS generates parameter names with the _Parm prefix followed by a unique integer (for example, _Parm1, _Parm2, ...). This row-and-column-sets *location* specification is the same as the following specification:

```
matrix _Gamma_ [2,4 1],[3,4 1];
```

It is also equivalent to the following elementwise specification:

```
matrix _Gamma_ [2,4],[2,1],[3,4],[3,1];
```

If you provide a *parameter-spec* list after the row-and-column-sets *location*, the parameters in the list are assigned to the matrix elements. For example, consider the following specification:

```
matrix _Gamma_ [2 3,4 1] = gamma1-gamma4;
```

This specification is equivalent to the following elementwise specification:

```
matrix _Gamma_ [2,4] = gamma1,
               [2,1] = gamma2,
               [3,4] = gamma3,
               [3,1] = gamma4;
```

It is not necessary for all the elements specified in the row-and-column-sets *location* to be assigned with the parameters in the *parameter-spec* list. For example, the following *iset* and *jset* specify a maximum of six elements, but only five parameters are assigned as a result of a shorter *parameter-spec* list:

```
matrix _Gamma_ [2 to 4,1 5] = gamma1-gamma4;
```

This specification is equivalent to the following elementwise specification:

```
matrix _Gamma_ [2,1] = gamma1,
               [2,5] = gamma2,
               [3,1] = gamma3,
               [3,5] = gamma4,
               [4,1] = gamma5;
```

In this case, _Gamma_[4, 5] is not specified and is fixed at zero by default.

With the row-and-column-sets *location* specifications, you need to be aware of the matrix type being specified. For example, the following specification of the symmetric matrix **S** results in an out-of-bounds error:

```
matrix S [1 2,1 2] = s1-s4;
```

This specification is equivalent to the following elementwise specification:

```
matrix S [1,1] = s1,
         [1,2] = s2,
         [2,1] = s3,
         [2,2] = s4;
```

The specification of the $\mathbf{S}[1,2]$ element is not valid because you can specify only the lower triangular elements of a symmetric matrix in PROC CALIS. The upper triangular elements are redundant and are taken into account by PROC CALIS during computations.

Specifying Fixed and Free Parameters in Model Matrices

For clarity in describing various *location* notations, the *parameter-spec* list contains only free parameters in the examples. In general, you can specify fixed values, free parameters, and initial values in the *parameter-spec* list. The syntax for the *parameter-spec* list is the same as the *parameter-spec* list for the VARIANCE statement. You can specify the following five types of the parameters in the MATRIX statement:

- an unnamed free parameter
- an initial value
- a fixed value

- a free parameter with a name provided
- a free parameter with a name and initial value provided

The following example demonstrates these five types of specifications:

```
matrix A  [1,2],
          [1,3] = (.2),
          [1,4] = .3,
          [2,3] = a1,
          [2,4] = a2(.5);
```

In this statement, $\mathbf{A}[1,2]$ is an unnamed free parameter. For this element, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1). $\mathbf{A}[1,3]$ is an unnamed free parameter with an initial value of 0.2. PROC CALIS also generates a parameter name for this element. $\mathbf{A}[1,4]$ is fixed at 0.3. This value does not change in estimation. $\mathbf{A}[2,3]$ is a free parameter named a1. No initial value is given for this element. $\mathbf{A}[2,4]$ is a free parameter named a2 with an initial value of 0.5.

You can also specify different types of parameters in the *parameter-spec* list. The preceding specification is equivalent to the following specification:

```
matrix A  [1,2],
          [1 2,3 4] = (.2) .3  a1-a2 (.5);
```

Notice that 0.5 is the initial value for a2 but not for a1 because this specification is the same as:

```
matrix A  [1,2],
          [1 2,3 4] = (.2) .3  a1 a2(.5);
```

When you use *parameter-spec* lists with mixed parameters, you must be careful about how the initial value syntax is interpreted with and without a parameter name before it. With a parameter before the initial value, the initial value is for the parameter, as shown in the following statement:

```
matrix S  [1,1] = s1 s2 (.2);
```

This specification is the same as the following elementwise specification:

```
matrix S  [1,1] = s1,
          [2,2] = s2(.2);
```

This means that 0.2 is the initial value of parameter s2, but not interpreted as an unnamed free parameter for $\mathbf{S}[3,3]$. If you do intend to set the free parameter s2 for $\mathbf{S}[2,2]$ without an initial value and set the initial value 0.2 for $\mathbf{S}[3,3]$, you can use a null initial value for the s2 parameter, as shown in the following:

```
matrix S  [1,1] = s1 s2() (.2);
```

This specification is the same as the following elementwise specification:

```
matrix S  [1,1] = s1,
          [2,2] = s2,
          [3,3] = (.2);
```

Modifying a Parameter Specification from a Reference Model

If you define a new COSAN, LISMOD, or MSTRUCT model by using a reference (old) model in the REFMODEL statement, you might want to modify some parameter specifications from the MATRIX statement of the reference model before transferring the specifications to the new model. To change a particular matrix element specification from the reference model, you can simply respecify the same matrix element with the desired parameter specification in the MATRIX statement of the new model. To delete a particular matrix parameter from the reference model, you can specify the desired matrix element with a missing value specification in the MATRIX statement of the new model.

For example, suppose that _PHI_[1, 2] is a free parameter in the reference model but you do not want this matrix element be a free parameter in the new model, you can use the following specification in the new model:

```
matrix _PHI_  [1,2] = .;
```

Notice that the missing value syntax is valid only when you use the REFMODEL statement. See the section "Modifying a COSAN Model from a Reference Model" on page 1062 for a more detailed example of COSAN model respecification. See the section "Modifying a LISMOD Model from a Reference Model" on page 1099 for a more detailed example of LISMOD model respecification. See the section "Modifying an MSTRUCT Model from a Reference Model" on page 1130 for a more detailed example of MSTRUCT model respecification.

MEAN Statement

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

where *assignment* represents:

var_list < **=** *parameter-spec* >

The MEAN statement specifies the mean or intercept parameters in connection with the FACTOR, LINEQS, and PATH modeling languages. With the MEAN statement specification, PROC CALIS analyzes the mean structures in addition to the covariance structures.

In each *assignment* of the MEAN statement, you list the *var_list* that you want to specify for their means or intercepts. Optionally, you can provide a list of parameter specifications in a *parameter-spec* after an equal sign for each *var_list*. The syntax of the MEAN statement is exactly the same as that of the VARIANCE statement. See the VARIANCE statement on page 1167 for details about the syntax.

For the confirmatory FACTOR or PATH model, the variables in a *var_list* can be exogenous or endogenous. You specify the mean of a variable if the variable is exogenous. You specify the intercept of a variable if the variable is endogenous. However, for the LINEQS model, you can specify only the means of exogenous variables whose type is not error (that is, not the E- or D- variables) in the MEAN statement. You cannot specify the intercept parameters in the MEAN statement for the LINEQS model. Instead, you must specify the intercepts in the equations of the LINEQS statement.

You can specify the following five types of the parameters for the means or intercepts in the MEAN statement:

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

For example, consider a PATH model with exogenous variables x1, x2, and x3 and endogenous variables y4 and y5. The following MEAN statement illustrates the five types of specifications in five *assignments*:

```
mean x1 ,
     x2 = (3.0),
     x3 = 1.5,
     y4 = intercept1,
     y5 = intercept2(0.6);
```

In this statement, the mean of x1 is specified as an unnamed free parameter. For this mean, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1). The mean of x2 is an unnamed free parameter with an initial value of 3.0. PROC CALIS also generates a parameter name for this mean. The mean of x3 is a fixed value of 1.5. This value stays the same during the estimation. The intercept of endogenous variable y4 is a free parameter named intercept1. The intercept of endogenous variable y5 is a free parameter named intercept2 with an initial value of 0.6.

The syntax of the MEAN statement is the same as the syntax of the VARIANCE statement. See the VARIANCE statement for more illustrations about the usage.

Default Mean and Intercept Parameters

If the mean structures are analyzed, all the means and intercepts of the *manifest* variables in the confirmatory FACTOR, LINEQS, or PATH model are free parameters by default. For these default free mean or intercept parameters, PROC CALIS generate the parameter names with the _Add prefix and appended with unique integer suffixes. For the FACTOR and PATH model, you can use the MEAN statement specification to override these default mean or intercept parameters in situations where you want to set parameter constraints, provide initial or fixed values, or make parameter references. For the LINEQS model, you can use the MEAN statement specification to override only the default mean parameters. The intercept parameters of the LINEQS model must be specified in the equations of the LINEQS statement.

Fixed zero is another type of default mean or intercept parameters for the FACTOR, LINEQS, or PATH model. All the intercepts and means of the *latent* variables in these models are fixed zeros by default. For the FACTOR and PATH models, you can override these default fixed zeros by using the MEAN statement specifications. However, for the LINEQS model, you can override only the default fixed zeros of the latent variables whose type is not error. That is, you can use the MEAN statement to override the default zero mean for the exogenous latent factors (excluding the error or disturbance variables) or use the LINEQS statement to override the default zero intercept for the endogenous latent factors. The fixed zero means for the error or disturbance variables in the LINEQS model reflects the model restrictions. There is no way you can override these default zero means.

Modifying a Mean or Intercept Parameter Specification from a Reference Model

If you define a new FACTOR, LINEQS, or PATH model by using a reference (old) model in the REFMODEL statement, you might want to modify some parameter specifications from the MEAN statement of the reference model before transferring the specifications to the new model. To change a particular mean or intercept specification from the reference model, you can simply respecify the same mean or intercept with the desired parameter specification in the MEAN statement of the new model. To delete a particular mean or intercept parameter from the reference model, you can specify the desired mean or intercept with a missing value specification in the MEAN statement of the new model.

For example, suppose that the mean of F1 is specified in the reference model, but you do not want this mean specification be transferred to the new model. You can use the following MEAN statement specification in the new model:

```
mean F1 = .;
```

Note that the missing value syntax is valid only when you use with the REFMODEL statement. See the section "Modifying a FACTOR Model from a Reference Model" on page 1080 for a more detailed example of FACTOR model respecification. See the section "Modifying a LINEQS Model from a Reference Model" on page 1094 for a more detailed example of LINEQS model respecification. See the section "Modifying a PATH Model from a Reference Model" on page 1145 for a more detailed example of PATH model respecification.

As discussed in a preceding section, PROC CALIS generates default free mean or intercept parameters for manifest variables in the FACTOR, LINEQS, or PATH model if you do not specify them explicitly in the MEAN statement (and the LINEQS statement for the LINEQS model). When you use the REFMODEL statement for defining a reference model, these default free mean or intercept parameters in the old (reference) model are not transferred to the new model. Instead, the new model generates its own set of default free mean or intercept parameters *after* the new model is resolved from the reference model, the REFMODEL statement options, the RENAMEPARM statement, and the MEAN statement (and the LINEQS statement for the LINEQS model) specifications in the new model. This also implies that if you want any of the mean or intercept parameters to be constrained across the models by means of the REFMODEL specification, you must specify them explicitly in the MEAN statement (or the LINEQS statement for the LINEQS model) of the reference model so that the same mean or intercept specification is transferred to the new model.

MODEL Statement

MODEL *i* < / *options* > **;**

where *i* is an assigned model number between 1 and 9999, inclusively.

A MODEL statement signifies the beginning of a model specification block and designates a model number for the model. All main and subsidiary model specification statements after a MODEL statement belong in that model until another MODEL or GROUP statement is encountered.

The MODEL statement itself does not serve the purpose of model specification, which is actually done by the main and subsidiary model specification statements that follow it. The MODEL statement serves as a "place-holder" of specification of a single model. It also makes the reference to a model easier with an assigned model number. For example, consider the following statements:

```
proc calis;
   group 1 / data=women_data;
   group 2 / data=men_data;
   model 1 / group=1 label='Women Model';
      {model 1 specification here}
   model 2 / group=2 label='Men Model';
      {model 2 specification here}
   run;
```

This example illustrates a two-group analysis with two models. One is model 1 labeled as 'Women Model' in a MODEL statement. Another is model 2 labeled as 'Men Model' in another MODEL statement. The two groups, group 1 and group 2, as defined in two separate GROUP statements, are fitted by model 1 and model 2, respectively, as indicated by the GROUP= option of the MODEL statements. Within the scope of model 1, you provide model specification statements by using the main and subsidiary model specification statements. Usually, one of the following main model specification statements is used: FACTOR, LINEQS, LISMOD, MSTRUCT, PATH, RAM, or REFMODEL. Similarly, you provide another set of model specification statements within the scope of model 2.

Hence, for an analysis with a single group, the use of the MODEL statement is not necessary because the model that fits the group is unambiguous.

You can set model-specific *options* in each MODEL statement. All but two of these *options* are also available in the PROC CALIS statement. If you set these *options* in the PROC CALIS statement, they apply to all models, unless you respecify them in the local MODEL statements. If you want to apply some *options* only to a particular model, specify these *options* in the MODEL statement that corresponds to that model.

You can also set group-specific *options* in the MODEL statement. These group *options* apply to the groups that are specified in GROUP= option of the MODEL statement. See the section "Options Available in the GROUP and PROC CALIS Statements" on page 1087 for a detailed descriptions of these group *options*.

Options Available Only in the MODEL Statement

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 quoted strings for labels.

GROUP | GROUPS=*int-list*

specifies a list of integers which represent the groups to be fitted by the model.

Options Available in the MODEL and PROC CALIS Statements

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

Option	Description
DEMPHAS= on page 1031	Emphasizes the diagonal entries
EFFPART | TOTEFF on page 1031	Displays total, direct, and indirect effects
EXTENDPATH | GENPATH on page 1032	Displays the extended path estimates that include variances and covariances
INEST= on page 1032	Specifies the data set that contains the initial values and constraints
INMODEL | INRAM= on page 1033	Specifies the data set that contains the model specifications
MEANSTR on page 1039	Analyzes the mean structures
MODIFICATION on page 1040	Computes modification indices
NOMEANSTR on page 1041	Deactivates the inherited MEANSTR option
NOMOD on page 1041	Suppresses modification indices
NOORDERSPEC on page 1042	Displays model specifications and results according to the input order
NOPARMNAME on page 1042	Suppresses the printing of parameter names in results
NOSTAND on page 1042	Suppresses the standardized output
NOSTDERR on page 1042	Suppresses standard error computations
ORDERSPEC on page 1044	Orders the model output displays according to the parameter types within each model
OUTEST= on page 1044	Specifies the data set that outputs the estimates and their covariance matrix
OUTMODEL | OUTRAM= on page 1045	Specifies the output data set for storing the model specification and results
PARMNAME on page 1046	Displays parameter names in model specifications and results
PDETERM on page 1046	Computes the determination coefficients
PESTIM on page 1047	Prints parameter estimates
PINITIAL on page 1047	Prints initial pattern and values
PLATCOV on page 1047	Computes the latent variable covariances and scoring coefficients
PRIMAT on page 1048	Displays results in matrix forms
READADDPARM on page 1049	Instructs generated default parameters be read in the INMODEL= data set
STDERR on page 1050	Computes the standard errors

Options Available in MODEL, GROUP, and PROC CALIS Statements

Some options in the GROUP statement can also be specified in the MODEL statements. Group options that are specified the MODEL statements are transferred to the GROUP statements that define the groups that are fitted by the associated models in the MODEL statements. This is a little more convenient than setting the common group options individually in the GROUP statements for all fitted groups by a model. See the section "Options Available in GROUP, MODEL, and PROC CALIS Statements" on page 1088 for a reference of these options.

MSTRUCT Statement

MSTRUCT < **VAR=***var_list* > **;**

MSTRUCT stands for matrix structures. As opposed to other modeling languages, in which the mean and covariance structures are implied from paths, equations, or complicated model matrix computations, the MSTRUCT language is for direct structured mean and covariance models.

In the MSTRUCT statement, you define the list of variables. You can use MATRIX statements to specify the parameters in the mean and covariance structures:

MSTRUCT < **VAR=***var_list* > **;**
  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 1111 for details. If you do not use any MATRIX statement for specifying parameters, a saturated model is assumed. This means that all elements in the covariance and mean (if modeled) matrices are free parameters in the model.

The order of variables in the *var_list* of the MSTRUCT statement is important; it is used to refer to the row and column variables of the _COV_ and the _MEAN_ matrices. The variables specified in the list should be present in the input data set that is intended for the MSTRUCT model. With direct mean and covariance structures on the observed variables, no latent variables are explicitly involved in the MSTRUCT modeling language. However, this does not mean that the MSTRUCT modeling language cannot handle latent variable models. With additional specifications in the PARAMETERS and the SAS programming statements, it is possible to fit certain latent variable models by using the MSTRUCT modeling language. Despite this, the code might get too complicated and error-prone. Hence, using the MSTRUCT modeling language for latent variable modeling is not recommended for novice users. The LINEQS, LISMOD, PATH, or RAM modeling language should be considered first for latent variable modeling.

Default Parameters

It is important to understand the default parameters in the MSTRUCT model. If you know which parameters are default free parameters, you can make your specification more efficient by omitting the specifications of those parameters that can be set by default. For example, you do not need to specify any elements of the _COV_ matrix if all elements are supposed to free parameters. See the section “Default Parameters in the MSTRUCT Model” on page 1222 for details about the default parameters of the FACTOR model.

Modifying an MSTRUCT Model from a Reference Model

This section assumes that you use a REFMODEL statement within the scope of a MODEL statement and that the reference model (or base model) is also an MSTRUCT model. The reference model is called the old model, and the model that refers to the old model is called the new model. If the new model is not intended to be an exact copy of the old model, you can use the following extended MSTRUCT modeling language to make modifications within the scope of the MODEL statement for the new model. The syntax is similar to, but not exactly the same as, the ordinary MSTRUCT modeling language, as described in the section "MSTRUCT Statement" on page 1130. The syntax for respecifying or modifying an MSTRUCT model takes the following form:

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

In the respecification, you should not put any VAR= list in the MSTRUCT statement, as you would do for specifying the original base model. The reason is that parameter respecifications in the new model refer to the variables in the VAR= list of the old model. Therefore, the VAR= list in the new model is implicitly assumed to be exactly the same as that in the old model. This renders the specification of a VAR= list of the MSTRUCT statement of the new model unnecessary. Because the VAR= option is the only possible option in the MSTRUCT statement, it also implies that the entire MSTRUCT statement is optional for the new model.

You can use the MATRIX _COV_ and MATRIX _MEAN_ statements to modify from the old model by using the same syntax as in ordinary MSTRUCT modeling language. In addition, in the respecification syntax, you can use the missing value '.' to drop a parameter location from the old model.

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

Duplication: If you do not specify in the new model a parameter location that exists in the old model, the old parameter specification is duplicated in the new model.

Addition: If you specify in the new model a parameter location that does not exist in the old model, the new parameter specification is used in the new model.

Deletion: If you specify in the new model a parameter location that also exists in the old model and the new parameter is denoted by the missing value '.', the old parameter specification is not copied into the new model.

Replacement: If you specify in the new model a parameter location that also exists in the old model and the new parameter is not denoted by the missing value '.', the new parameter specification replaces the old one in the new model.

For example, consider the following statements for a two-group analysis:

```
proc calis;
   group 1 / data=d1;
   group 2 / data=d2;
   model 1 / group=1;
      mstruct var=V1-V6;
      matrix _COV_ [1,1] = 6*vparm (8.),
                   [2,]  = cv21,
                   [3,]  = cv31,
                   [4,]  = cv41 cv42 cv43,
                   [5,]  = cv51 cv52 cv53 cv54,
                   [6,]  = cv61 cv62 cv63 cv64 cv65;
   model 2 / group=2;
      refmodel 1;
      matrix _COV_ [2,]  = 3.,
                   [3,2] = cv32,
                   [4,]  = . . . ,
                   [5,]  = . . . ,
                   [6,]  = . . . ;
   run;
```

In these statements, you specify Model 2 by referring to Model 1 in the REFMODEL statement. Hence, Model 2 is called the new model that refers to the old model, Model 1. Because they are not respecified in the new model, all parameters on the diagonal of the covariance matrix are duplicated from the old model for the new model. Similarly, parameter locations associated with the cv54, cv64, and cv65 parameters are also duplicated in the new model.

An added parameter in the new model is cv32 for the covariance between V3 and V2. This parameter location is not specified in the old model.

In the new model, parameters for the covariances between the variable sets {V1 V2 V3} and {V4 V5 V6} are all deleted from the old model. The corresponding parameter locations for these covariances are given missing values '.' in the new model, indicating that they are no longer free parameters as in the old model. Deleting these parameters amounts to setting the corresponding covariances to fixed zeros in the new model.

Finally, covariance between V2 and V1 is changed from a free parameter 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 number operator*
- *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 CALIS 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 CALIS by using the NLOPTIONS statement. The NLOPTIONS statement provides more displayed and file output control on the results of the optimization process, and it permits the same set of termination criteria as in PROC NLP. These are more technical options that you might not need to specify in most cases.

Several statistical procedures support the use of NLOPTIONS statement. The syntax of NLOPTIONS statement is common to all these procedures and can be found in the section "NLOPTIONS Statement" on page 496 in Chapter 19, "Shared Concepts and Topics."

See the section "Use of Optimization Techniques" on page 1283 for more information about the use of optimization techniques in PROC CALIS.

OUTFILES Statement

OUTFILES | OUTFILE *file_option* < *file_option* ... > **;**

where *file_option* represents one of the following:

- OUTMODEL | OUTRAM= *file_name* [MODEL= *int_list* < , *int_list* >]
- OUTSTAT= *file_name* [GROUP= *int_list* < , *int_list* >]
- OUTWGT= *file_name* [GROUP= *int_list* < , *int_list* >]

with *file_name* representing an output file name and *int_list* representing list of model or group numbers

Use the OUTFILES statement when you need to output multiple-group or multiple-model information to output files in a complex way. In each OUTFILES statement, each possible *file_option* should appear no more than once. However, as needed, you can use the OUTFILES statement more than once. For example, suppose you want to create two OUTWGT= files for different sets of groups. You can specify the OUTFILES statement twice, as shown in the following specification:

```
outfiles outwgt=file1 [group=1,2];
outfiles outwgt=file2 [group=3,4];
```

In the first OUTFILES statement, the weights for groups 1 and 2 are output to the file file1. In the second OUTFILES statement, the weights for groups 3 and 4 are output to the file file2.

When the OUTMODEL=, OUTSTAT=, or OUTWGT= option is intended for *all* groups or models, you can simply specify the option in the PROC CALIS statement. Only when you need to output the group (model) information from *more than one* group (model), *but not all* groups (models), to a single output file does the use the OUTFILES statement become necessary. For example, consider the following specification:

```
proc calis method=gls;
   outfiles outmodel=outmodel [model=1,3]
            outwgt=outwgt [group=1,2]
            outstat=outstat [group=2,3];
   group 1 / data=g1;
   group 2 / data=g2;
   group 3 / data=g3 outwgt=outwgt3;
   model 1 / group=1;
      factor N=3;
   model 2 / group=2;
      factor N=2;
   model 3 / group=3;
      factor N=3;
   run;
```

You fit three different factor models to three groups: Model 1 for Group 1, Model 2 for Group 2, and Model 3 for group 3. In the OUTFILES statement, you output model information from models 1 and 3 to an output file named outmodel, weight matrices from groups 1 and 2 to outwgt, and statistics from groups 2 and 3 to outstat. In each of these output files, you have information from more than one (but not all) groups or models. In the GROUP statement for group 3, you have another OUTWGT= file named outwgt3 for group 3 alone.

Note that you cannot specify the preceding output file organization by using the following statements:

```
proc calis method=gls;
   group 1 / data=g1 outwgt=outwgt;
   group 2 / data=g2 outwgt=outwgt outstat=outstat;
   group 3 / data=g3 outwgt=outwgt3 outstat=outstat;
   model 1 / group=1 outmodel=outmodel;
      factor N=3;
   model 2 / group=2;
      factor N=2;
   model 3 / group=3 outmodel=outmodel;
      factor N=3;
   run;
```

This specification will not work because SAS forbids the repeated specification of the same output file in the same PROC CALIS run. That is, you cannot specify OUTWGT=outwgt, OUTSTAT=outstat, or OUTMODEL=outmodel more than once in the PROC CALIS run without causing file invocation problems (however, multiple specification of the same input file is not a problem).

If you specify any of the output files for a group (or a model) in both of the OUTFILES and the GROUP (or MODEL) statements, the destination specified in the more specific GROUP (or MODEL) statement will be used. For example, for the following specification PROC CALIS will save the Model 2 information in the OUTMODEL=outmodel2 data set, but not in the OUTMODEL=outfile1 data set:

```
proc calis method=gls;
   outfiles outmodel=outfile1 [model=1,2];
   group 1 / data=g1;
   group 2 / data=g2;
   model 1 / group=1;
      factor N=3;
   model 2 / group=2 outmodel=outmodel2;
      factor N=2;
   run;
```

The OUTFILES statement is intended for arranging output files in a complex way. The use of the OUTFILES statement is unnecessary in the following situations:

- If you have a single-sample analysis, you do not need to use the GROUP statement. As a result, you can simply use the OUTSTAT= or OUTWGT= options in the PROC CALIS statement for specifying the output destinations. Therefore, the OUTFILES statement is not needed.
- If you have a single model in your analysis, you do not need to use the MODEL statement. As a result, you can simply use the OUTMODEL= options in the PROC CALIS statement for specifying the output destination. Therefore, the OUTFILES statement is not needed.
- If you have multiple groups or multiple models in your analysis and information for all groups or models is output to the same file, you do not need to use the OUTFILES statement. You can simply use the OUTSTAT=, OUTWGT=, or OUTMODEL= options in the PROC CALIS statement because the output file information is automatically propagated from the PROC CALIS statement to the groups or models.
- If you have multiple groups or multiple models in your analysis and each group or model has a unique output data file destination (including cases where some groups or models might not have any output

files), you do not need to use the OUTFILES statement. You can simply specify the OUTSTAT=, OUTWGT=, or OUTMODEL= options in the GROUP or MODEL statements.

PARAMETERS Statement

PARAMETERS | PARMS *parameters* << = > *numbers* >
<< , > *parameters* << = > *numbers* > . . . > **;**

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 CALIS 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 1164 for more details.

CAUTION: The OUTMODEL= or OUTRAM= data sets do not contain any information about the PARAMETERS statement or the SAS programming statements.

PARTIAL Statement

PARTIAL *variables* **;**

If you want the analysis to be based on a partial correlation or covariance matrix, use the PARTIAL statement to list the variables used to partial out the variables in the analysis. You can specify only one PARTIAL statement within the scope of each GROUP or PROC CALIS statement.

PATH Statement

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

where *path* represents any of the following specifications:

- single-headed path for defining functional relationship
- double-headed path for specifying variances or covariances
- 1-path for specifying means or intercepts

For example, the following PATH statement contains only the single-headed paths:

```
PATH
   V1    <---  V2,
   V2    <---  V4 V5,     /* same as: V2 <--- V4 and V2 <--- V5 */
   V3    --->  V5,        /* same as: V5 <--- V3 */
   V4 V5 <---  V6 V7;     /* same as: V4 <--- V6, V4 <--- V7,
                             V5 <--- V6, and V5 <--- V7 */
```

Although the most common definition of paths refer to these single-headed paths, PROC CALIS extends the definition of paths to include the so-called "variance-paths," "covariance-paths," and "1-paths" that refer to the variance, covariance, and the mean or intercept parameters, respectively. Corresponding to these extended path definitions, PROC CALIS provides the double-headed path and 1-path syntax. For example, the following PATH statement contains single-headed paths for specifying functional relationships and double-headed paths for specifying variances and covariances:

```
PATH
   V1    <---  V3-V5,    /* same as: V1 <--- V3, V1 <--- V4, and V1 <--- V5 */
   V2    <---  V4 V5,
   V3    <---  V5,
   V1    <-->  V1,       /* error variance of V1 */
   <--> V2 V3,           /* error variances of V2 and V3 */
   V2    <-->  V3,       /* error covariance between V2 and V3 */
   <--> [V4 V5];         /* variances and covariance for V4 and V5 */
```

The following PATH statement contains single-headed paths for specifying functional relationships and 1-paths for specifying means and intercepts:

```
PATH
   V1    <---  V3-V5,
   V2    <---  V4 V5,
   V3    <---  V5,
   1     --->  V1,       /* intercepts for V1 */
   1     --->  V2-V3,    /* intercepts for V2 and V3 */
   1     --->  V4 V5;    /* means of V4 and V5 */
```

Details about the syntax of these three different types of *paths* are described later. Instead of using double-headed paths and 1-paths, you can also specify these parameters by the subsidiary model specification statements such as the PVAR, PCOV, and the MEAN statements, as shown in the following syntactic structure of the PATH modeling language:

> **PATH** *path* < *, path . . .* > **;**
>   **PVAR** *partial-variance-parameters* **;**
>   **PCOV** *partial-covariance-parameters* **;**
>   **MEAN** *mean-parameters* **;**

Typically, in this syntactic structure the *paths* contains only single-headed paths for representing the functional relationships among variables, which could be observed or latent. The *paths* are separated by commas. You can specify at most one PATH statement in a model within the scope of either the PROC CALIS statement or a MODEL statement.

Next, the PVAR statement specifies the parameters for the variances or error (partial) variances. The PCOV statement specifies the parameters for the covariances or error (partial) covariances. The MEAN statement specifies the parameters for the means or intercepts. For details about these subsidiary model specification statements, see the syntax of the individual statements.

A natural question now arises. For the specification of variances, covariances, intercepts, and means, should you use the extended path syntax that includes double-headed paths and 1-paths or the subsidiary model specification statements such as the PVAR, PCOV, and MEAN statements? If you want to specify all parameters in a single statement and hence output and view all the parameter estimates in a single output table, then the extended path syntax would be your choice. If you want to use more common language for specifying and viewing the parameters or the estimates of variances, covariances, means, and intercepts, then the subsidiary model specification statements serve the purpose better.

You are not restricted to using extended path syntax or the subsidiary model statements exclusively in a PATH model specification. For example, you might specify the variance of V1 by using the double-headed path syntax and the variance of V2 by using the PVAR statement. The only restriction is that you cannot specify the same parameter twice. In addition, even if you specify your PATH model without using double-headed paths or 1-paths, you can include the estimation results associated with these extended paths in the same output table for the single-headed paths by using the EXTENDPATH or GENPATH option. This way all the estimates of the PATH model can be shown in a single output table.

The Single-Headed Path Syntax for Specifying Functional Relationships

> *var_list arrow var_list2* < = *parameter-spec* >

where *var_list* and *var_list2* are lists of variables, *parameter-spec* is an optional specification of parameters, and *arrow* represents either a *left-arrow* is one of the following forms:

    <---, <--, <-, or <

or a *right-arrow* is one of the following forms:

    --->, -->, ->, or >

In each single-headed path, you specify two lists of variables: *var_list* and *var_list2*. Depending on the direction of the *arrow* specification, one group of variables contains the outcome variables and the other group contains the predictor variables. Optionally, you can specify the *parameter-spec* at the end of each path entry. You can specify the following five types of the parameters for the path entries:

- unnamed free parameters
- initial values
- fixed values
- free parameters with names provided
- free parameters with names and initial values provided

For example, in the following statement you specify a model with five paths:

```
PATH
   V1  <---  F1   ,
   V2  <---  F1   = (0.5),
   V3  <---  F1   =  1.,
   V4  <---  F1   =  b1,
   V5  <---  F1   =  b2 (.4);
```

The first path entry specifies a path from F1 to V1. The effect of F1 (or the path coefficient) on V1 is an unnamed free parameter. For this path effect parameter, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1). The second path entry specifies a path from F1 to V2. The effect of F1 is also an unnamed free parameter with an initial estimate of 0.5. PROC CALIS also generates a parameter name for effect parameter. The third path entry specifies a path from F1 to V3. The effect of F1 is also a fixed value of 1.0. This value stays the same in the model estimation. The fourth path entry specifies a path from F1 to V4. The effect of F1 is a free parameter named b1. The fifth path entry specifies a path from F1 to V5. The effect of F1 is a free parameter named b2, with an initial value of 0.4.

You can specify multiple variables in the *var_list* and *var_list2* lists. For example, the following statement specifies five paths from F1 to V1–V5:

```
PATH
   F1 ---> V1-V5;
```

All the five effects of F1 on the five variables are unnamed free parameters. If both *var_list* and *var_list2* lists contain multiple variables, you must be careful about the order of the variables when you also specify parameters at the end of the path entry. For example, the following statement specifies the paths from the predictor variables x1–x2 to the outcome variables y1–y3:

```
PATH
   y1-y3 <--- x1-x2   = a1-a6;
```

The PATH statement specifies six paths in the path entry. These six paths have effect parameters a1–a6. This specification is equivalent to the following specification:

```
PATH
   y1 <--- x1   = a1;
   y1 <--- x2   = a2;
   y2 <--- x1   = a3;
   y2 <--- x2   = a4;
   y3 <--- x1   = a5;
   y3 <--- x2   = a6;
```

The following statement shows another example of multiple-path specification:

```
PATH
   x1-x2 ---> y1-y3   = b1-b6;
```

This specification is equivalent to the following specification with separate path specifications:

```
PATH
   x1 ---> y1   = b1;
   x1 ---> y2   = b2;
   x2 ---> y3   = b3;
   x2 ---> y1   = b4;
   x2 ---> y2   = b5;
   x2 ---> y3   = b6;
```

You can also specify parameter with mixed types in any path entry, as shown in the following specification:

```
PATH
   F1 ---> y1-y3    = 1.  b1(.5) (.3),
   F2 ---> y4-y6    = 1.  b2  b3(.7);
```

This specification is equivalent to the following expanded version:

```
PATH
   F1 ---> y1    = 1.,
   F1 ---> y2    = b1(.5),
   F1 ---> y3    = (.3),
   F2 ---> y4    = 1.,
   F2 ---> y5    = b2,
   F2 ---> y6    = b3(.7);
```

Notice that in the original specification with multiple-path entries, 0.5 is interpreted as the initial value for the parameter b1, but not as the initial estimate for the path from F1 to y3. In general, an initial value that follows a parameter name is associated with the free parameter.

If you indeed want to specify that b1 is a free parameter *without* an initial estimate and 0.5 is the initial estimate for the path from F1 to y3 (while keeping all other specification the same), you can use a null initial value specification, as shown in the following statement:

```
PATH
   F1 ---> y1-y3    = 1.  b1() (.5) ,
   F2 ---> y4-y6    = 1.  b2  b3(.7);
```

This way 0.5 becomes the initial value for the path from F1 to y3. Because a parameter list with mixed types might be confusing, you can break down the specifications into separate path entries to remove ambiguities. For example, you can use the following specification equivalently:

```
PATH
   F1 ---> y1       = 1.,
   F1 ---> y2       = b1,
   F1 ---> y3       = (.5) ,
   F2 ---> y4-y6    = 1.  b2  b3(.7);
```

The equal signs in the path entries are optional when the parameter lists do not start with a parameter name. For example, the preceding specification is the same as the following specification:

```
PATH
   F1 ---> y1         1.,
   F1 ---> y2       = b1,
   F1 ---> y3        (.5) ,
   F2 ---> y4-y6      1.  b2  b3(.7);
```

Notice that in the second path entry, you must retain the equal sign because b1 is a parameter name. Omitting the equal sign makes the specification erroneous because b1 is treated as a variable. This might cause serious estimation problems. Omitting the equal signs might be cosmetically appealing in specifying fixed values or initial values (for example, the first and the third path entries). However, the gain of doing that is not much as compared to the clarity of specification that results from using the equal signs consistently.

NOTE: You do not need to specify single-headed paths from the errors or disturbances (that is, error terms) in the PATH model specification, even though the functional relationships between variables are not assumed to be perfect. Essentially, the roles of error terms in the PATH model are in effect represented by the associated default error variances of the endogenous variables, making it unnecessary to specify any single-headed paths from error or disturbance variables.

The Double-Headed Path Syntax That Uses Two Variable Lists for Specifying Variances and Covariances

var_list two-headed-arrow var_list2 < = *parameter-spec* >

where a *two-headed-arrow* is one of the following forms:
<-->, <->, or <>

This syntax enables you to specify covariances between the variables in *var_list* and the variables in *var_list2*. Consider the following example:

```
PATH
   v1      <-->  v2,
   v3 v4   <-->  v5 v6 v7 = cv1-cv6;
```

The first double-headed path specifies the covariance between v1 and v2 as an unnamed free parameter. PROC CALIS generates a name for this unnamed free parameter. The second double-headed path specifies six covariances with parameters named cv1–cv6. This multiple-covariance specification is equivalent to the following elementwise covariance specification:

```
PATH
   v3  <-->  v5    = cv1,
   v3  <-->  v6    = cv2,
   v3  <-->  v7    = cv3,
   v4  <-->  v5    = cv4,
   v4  <-->  v6    = cv5,
   v4  <-->  v7    = cv6;
```

Note that the order of variables in the list is important for determining the assignment of the parameters in the *parameter-spec* list.

If the same variable appears in both of the *var_list* and *var_list2* lists, the “covariance” specification becomes a variance specification for that variable. For example, the following statement specifies two variances:

```
PATH
   v1      <-->  v1     = 1.0,
   v2      <-->  v2 v3 = sigma2  cv23;
```

The first double-headed path entry specifies the variance of v1 as a fixed value of 1.0. The second double-headed path entry specifies the variance of v2 as a free parameter named sigma2, and then the covariance between v2 and v3 as a free parameter named cv23.

It results in an error if you attempt to use this syntax to specify the variance and covariances among a set of variables. For example, suppose you intend to specify the variances and covariances among v1–v3 as unnamed free parameters by the following statement:

```
PATH
   v1-v3   <-->  v1-v3 ;
```

This specification expands to the following elementwise specification:

```
PATH
   v1  <-->  v1 ,
   v1  <-->  v2 ,
   v1  <-->  v3 ,
   v2  <-->  v1 ,
   v2  <-->  v2 ,
   v2  <-->  v3 ,
   v3  <-->  v1 ,
   v3  <-->  v2 ,
   v3  <-->  v3 ;
```

There are nine variance or covariance specifications, but all of the covariances are specified twice. This is treated as a duplication error. The correct way is to specify only the nonredundant covariances, as shown in the following elementwise specification:

```
PATH
   v1  <-->  v1 ,
   v2  <-->  v1 ,
   v2  <-->  v2 ,
   v3  <-->  v1 ,
   v3  <-->  v2 ,
   v3  <-->  v3 ;
```

However, the elementwise specification is quite tedious when the number of variables is large. Fortunately, there is another syntax for double-headed paths to deal with this situation. This syntax is discussed next.

The Double-Headed Path Syntax That Uses a Single Variable List for Specifying Variances

two-headed-arrow var_list < = parameter-spec >

This syntax enables you to specify variances among the variables in *var_list*. Consider the following example:

```
PATH
   <-->  v1    = (0.8),
   <-->  v2-v4 ;
```

The first double-headed path entry specifies the variance of v1 as an unnamed free parameter with an initial estimate of 0.8. The second double-headed path entry specifies the variances of v2–v4 as unnamed free parameters. No initial values are given for these three variances. PROC CALIS generates names for all these variance parameters. You can specify these variances equivalently by the elementwise covariance specification syntax, as shown in the following, but former syntax is much more efficient.

```
PATH
   v1 <-->  v1   = (0.8),
   v2 <-->  v2 ,
   v3 <-->  v3 ,
   v4 <-->  v4 ;
```

The Double-Headed Path Syntax That Uses a Single Variable List for Specifying Variances and Covariances

two-headed-arrow **[***var_list***]** < = *parameter-spec* >

This syntax enables you to specify all the variances and covariances among the variables in *var_list*. For example,the following statement specifies all the variances and covariances among v2–v4:

```
PATH
    <--> [v2-v4]  = 1.0  cv32  cv33(0.5) cv42 .7 cv44;
```

This specification is more efficient as compared with the following equivalent specification with elementwise variance or covariance definitions:

```
PATH
    v2 <--> v2    = 1.0,
    v3 <--> v2    = cv32 ,
    v3 <--> v3    = cv33(0.5),
    v4 <--> v2    = cv42,
    v4 <--> v3    = .7,
    v4 <--> v2    = cv44;
```

The double-headed path Syntax for Specifying Nonredundant Covariances

two-headed-arrow **(***var_list***)** < = *parameter-spec* >

This syntax enables you to specify all the nonredundant covariances among the variables in *var_list*. For example, the following statement specifies all the nonredundant covariances between v2–v4:

```
PATH
    <--> (v2-v5)  = cv1-cv6;
```

This specification is equivalent to the following elementwise specification:

```
PATH
    v3 <--> v2    = cv1 ,
    v4 <--> v2    = cv2 ,
    v4 <--> v3    = cv3 ,
    v5 <--> v2    = cv4 ,
    v5 <--> v3    = cv5 ,
    v5 <--> v4    = cv6 ;
```

The 1-path Syntax for Specifying Means and Intercepts

1 *right-arrow var_list* < = *parameter-spec*>

where a *right-arrow* is one of the following forms:
--->, -->, ->, or >

This syntax enables you to specify the means or intercepts of the variables in *var_list* as paths from the constant **1**. Consider the following example:

```
PATH
   v1 <---  v2-v4,
   1  --->  v1      =  alpha,
   1  --->  v2-v4   =  3*kappa;
```

The first single-headed path specifies that v1 is predicted by variables v2, v3, and v4. Next, the first 1-path entry specifies either the intercept of v1 as a free parameter named alpha. It is the intercept, rather than the mean, of v1 because endogenous in the PATH model. The second 1-path entry specifies the means of v2–v4 as constrained parameters. All these means or intercepts are named kappa so that they have the same estimate.

Therefore, whether the parameter is a mean or an intercept specified with the 1-path syntax depends on whether the associated variable is endogenous or exogenous in the model. The intercept is specified if the variable is endogenous. Otherwise, the mean of the variable is specified. Fortunately, any variable in the model can have either a mean or intercept (but not both) to specify. Therefore, the 1-path syntax is applicable to either the mean or intercept specification without causing conflicts.

Shorter and Longer Parameter Lists

If you provide fewer parameters in *parameter-spec* than the number of paths in a *path* entry, all the remaining parameters are treated as unnamed free parameters. For example, the following specification specifies the free parameter beta to the first path and assigns unnamed free parameters to the remaining four paths:

```
PATH
   F1 ---> y1 z1 z2 z3 z4   = beta;
```

This specification is equivalent to the following specification:

```
PATH
   F1 ---> y1   = beta,
   F1 ---> z1 z2 z3 z4;
```

If you intend to fill up all values with the last parameter specification in the list, you can use the continuation syntax **[...]**, **[..]**, or **[.]**, as shown in the following example:

```
PATH
   F1 ---> y1 z1 z2 z3 z4  = beta gamma [...];
```

This specification is equivalent to the following specification:

```
PATH
   F1 ---> y1 z1 z2 z3 z4  = beta 4*gamma;
```

The repetition factor **4*** means that gamma repeats 4 times.

However, you must be careful not to provide too many parameters. For example, the following specification results in an error:

```
PATH
   SES_Factor ---> y1 z1 z2 z3 z4  = beta gamma1-gamma6;
```

Because there are only five paths in the specification, parameters gamma5 and gamma6 are excessive.

Default Parameters

It is important to understand the default parameters in the PATH model. First, knowing which parameters are default free parameters makes your specification more efficient by omitting the specifications of those parameters that can be set by default. For example, because all variances and covariances among exogenous variables (excluding error terms) are free parameters by default, you do not need to specify them in the PATH model if these variances and covariances are not constrained. Second, knowing which parameters are default fixed zero parameters enables you to specify your model accurately. For example, because all error covariances in the PATH model are fixed zeros by default, you must use the PCOV statement or the double-headed path syntax to specify the partial (error) covariances among the endogenous variables if you want to fit a model with correlated errors. See the section "Default Parameters in the PATH Model" on page 1228 for details about the default parameters of the PATH model.

Modifying a PATH Model from a Reference Model

If you define a new model by using a reference (old) model in the REFMODEL statement, you might want to modify some path specifications from the PATH statement of the reference model before transferring the specifications to the new model. To change a particular path specification from the reference model, you can simply respecify the same path with the desired parameter specification in the PATH statement of the new model. To delete a particular path and its associated parameter from the reference model, you can specify the desired path with a missing value specification in the PATH statement of the new model.

PATH *path* < *, path ...* > **;**
  PVAR *partial-variance-parameters* **;**
  PCOV *partial-covariance-parameters* **;**
  MEAN *mean-parameters* **;**

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

Duplication: If you do not specify in the new model a parameter location that exists in the old model, the old parameter specification is duplicated in the new model.

Addition: If you specify in the new model a parameter location that does not exist in the old model, the new parameter specification is used in the new model.

Deletion: If you specify in the new model a parameter location that also exists in the old model and the new parameter is denoted by the missing value '.', the old parameter specification is not copied into the new model.

Replacement: If you specify in the new model a parameter location that also exists in the old model and the new parameter is not denoted by the missing value '.', the new parameter specification replaces the old one in the new model.

For example, consider the following specification of a two-group analysis:

```
proc calis;
   group 1 / data=d1;
   group 2 / data=d2;
   model 1 / group=1;
      path
         V1 <--- F1   = 1.,
         V2 <--- F1   = load1,
         V3 <--- F1   = load2,
         F1 <--- V4   = b1,
         F1 <--- V5   = b2,
         F1 <--- V6   = b3;
      pvar
         E1-E3     = ve1-ve3,
         F1        = vd1,
         V5-V6     = phi4-phi6;
      pcov
         V1 V2     = cve12;
   model 2 / group=2;
      refmodel 1;
      path
         V3 <--- F1   = load1,
      pcov
         V1 V2     =  .,
         V2 V3     = cve23;
   run;
```

You specify Model 2 by referring to Model 1 in the REFMODEL statement. Model 2 is the new model that refers to the old model, Model 1. This example illustrates the four types of model integration rules for the new model:

- Duplication: All parameter specifications, except for the partial covariance between V1 and V2 and the V3 <--- F1 path in the old model, are duplicated in the new model.
- Addition: The parameter cve23 for the partial covariance between V2 and V3 is added in the new model because there is no corresponding specification in the old model.
- Deletion: The specification of partial covariance between 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:

var_list < $*$ *var_list2* > < = *parameter-spec* >

The PCOV statement is a subsidiary model specification statement for the PATH model. You can use the PCOV statement only with the PATH modeling language. The PCOV statement specifies the covariances of exogenous variables, or the error covariances of endogenous variables in the PATH model. It can also specify the covariance between an exogenous variable and the error term of an endogenous variables, although this usage is rare in practice.

In each *assignment* of the COV statement, you specify variables in the *var_list* and the *var_list2* lists, followed by the covariance parameter specification in the *parameter-spec* list. The latter two specifications are optional. The syntax of the PCOV statement is the same as that of the COV statement. See the COV statement on page 1065 for details about specifying within- and between-list (partial) covariances.

The concept behind the PCOV statement is broader than that of the COV statement. The PCOV statement supports the partial covariance parameter specification in addition to the covariance parameter specification, which is the only type of parameter that the COV statement supports. This difference is also reflected from the sets of *var_list* and *var_list2* that you can use in the PCOV statement. In the COV statement, variables on the left-hand side of an *assignment* must be exogenous. However, in the PCOV statement, you can specify both exogenous and endogenous variables. If a pair of variables are both exogenous in a specification, you are defining a covariance parameter between the variables. If a pair of variables are both endogenous in a specification, you are defining a partial covariance parameter between of the variables. This partial covariance is usually interpreted as the error covariance between the two endogenous variables. If one variable is exogenous while the other is endogenous, you are defining a covariance parameter between the exogenous variable and the error term for the endogenous variable.

You can specify the following five types of the parameters for the partial covariances in the PCOV statement:

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

For example, consider a PATH model with exogenous variables x1, x2, and x3 and endogenous variables y4, y5 and y6. The following PCOV statement shows the five types of specifications in five *assignments*:

```
pcov x1 x2 ,
     x1 x3 = (0.5),
     x2 x3 = 2.0,
     y4 y5 = psi1,
     y5 y6 = psi2(0.4);
```

In this statement, the covariance between x1 and x2 is specified as an unnamed free parameter. For this covariance, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1). The covariance between x1 and x3 is an unnamed free parameter with an initial value of 0.5. PROC CALIS also generates a parameter name for this covariance. The covariance between x2 and x3 is a fixed value of 2.0. This value stays the same during the estimation. The error covariance between endogenous variables y4 and y5 is a free parameter named psi1. The error covariance between endogenous variables y5 and y6 is a free parameter named psi2 with an initial value of 0.4.

The syntax of the PCOV statement is the same as the syntax of the COV statement. See the COV statement for more illustrations about the usage.

Default Covariance Parameters

Although the PCOV statement specification is conceptually broader than the COV statement specification, their related default set of covariance parameters is the same—that is, all covariances among *exogenous* manifest and latent variables (excluding error or disturbance variables) are free parameters. Because the PCOV statement applies only to the PATH model, it is easy to understand why the covariances do not apply to the error or disturbance terms. The PATH model, as implemented in PROC CALIS, simply does not use any explicit error or disturbance terms. For the default free covariance parameters, PROC CALIS generate the parameter names with the _Add prefix and appended with unique integer suffixes. You can also use the PCOV statement specification to override these default covariance parameters in situations where you want to set parameter constraints, provide initial or fixed values, or make parameter references.

Another type of default partial covariances are fixed zeros. This default applies to the partial (error) covariances among all *endogenous* variables, and to the partial covariances between all *exogenous* variables and all *endogenous* variables in the path model. Again, you can override the default fixed values by providing explicit specification of these partial or error covariances in the PCOV statement.

Modifying a Covariance or Partial Covariance Parameter Specification from a Reference Model

If you define a new PATH model by using a reference (old) model in the REFMODEL statement, you might want to modify some parameter specifications from the PCOV statement of the reference model before transferring the specifications to the new model. To change a particular partial covariance specification from the reference model, you can simply respecify the same covariance with the desired parameter specification in the PCOV statement of the new model. To delete a particular partial covariance parameter from the reference model, you can specify the desired partial covariance with a missing value specification in the PCOV statement of the new model.

For example, suppose that you are defining a new PATH model by using the REFMODEL statement and that the covariance between variables F1 and V2 is defined as a fixed or free parameter in the reference model. If you do not want this fixed parameter specification to be copied into your new model, you can use the following specification in the new model:

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

Note that the missing value syntax is valid only when you use it with the REFMODEL statement. See the section "Modifying a PATH Model from a Reference Model" on page 1145 for a more detailed example of the PATH model respecification.

As discussed in the section "Default Covariance Parameters" on page 1148, PROC CALIS generates some default free covariance parameters for the PATH model if you do not specify them explicitly in the PCOV statement. When you use the REFMODEL statement for defining a reference model, these default free covariance parameters in the old (reference) model are not transferred to the new model. Instead, the new model generates its own set of default free covariance parameters *after* the new model is resolved from the reference model, the REFMODEL statement options, the RENAMEPARM statement, and the PCOV statement specifications in the new model. This also implies that if you want any of the (partial) covariance parameters to be constrained across the models by means of the REFMODEL specification, you must specify them explicitly in the PCOV statement of the reference model so that the same (partial) covariance specification is transferred to the new model.

PVAR Statement

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

where *assignment* represents:

var_list <= *parameter-spec*>

The PVAR statement specifies the variance or error (partial) variance parameters in connection with the confirmatory FACTOR and PATH models.

In each *assignment* of the PVAR statement, you list the *var_list* that you want to specify for their variances or error (partial) variances. Optionally, you can provide a list of parameter specifications (*parameter-spec*) after an equal sign for each *var_list* list. The syntax of the PVAR statement is exactly the same as that of the VARIANCE statement. See the VARIANCE statement on page 1167 for details about the syntax.

The concept behind the PVAR statement is broader than that of the VARIANCE statement. The PVAR statement supports the partial variance parameter specification in addition to the variance parameter specification, which is the only type of parameters that the VARIANCE statement supports. This difference is reflected from the set of *var_list* you can use in the PVAR statement. You can specify both exogenous variables and endogenous variables in the *var_list* list of the PVAR statement, but you can specify only exogenous variables in the *var_list* list of the VARIANCE statement. This conceptualization of the PVAR statement is needed in the FACTOR and PATH modeling languages because error variables are not explicitly defined in these models. You specify the variance of a variable if the variable in the *var_list* list of the PVAR statement is an exogenous (independent) variable in the FACTOR or PATH model. You specify the error (partial) variance of a variable if the variable in the *var_list* list of the PVAR statement is an endogenous (dependent) variable in the FACTOR or PATH model.

You can specify the following five types of the parameters for the partial variances in the PVAR statement:

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

For example, consider a PATH model with exogenous variables x1, x2, and x3 and endogenous variables y4 and y5. The following PVAR statement illustrates the five types of specifications in five *assignments*:

```
pvar x1 ,
     x2 = (2.0),
     x3 = 1.0,
     y4 = psi1,
     y5 = psi2(0.6);
```

In this statement, the variance of x1 is specified as an unnamed free parameter. For this variance, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1). The variance of x2 is an unnamed free parameter with an initial value of 2.0. PROC CALIS also generates a parameter name for this variance. The variance of x3 is a fixed value of 1.0. This value stays the same during the estimation. The error variance of endogenous variable y4 is a free parameter named psi1. The error variance of endogenous variable y5 is a free parameter named psi2 with an initial value of 0.6.

The syntax of the PVAR statement is the same as the syntax of the VARIANCE statement. See the VARIANCE statement for more illustrations about the usage.

Default Partial Variance Parameters

By default, all variances of the *exogenous* manifest and latent variables and all error (partial) variances of the *endogenous* manifest and latent variables are free parameters in the FACTOR or PATH model. For these default free variance parameters, PROC CALIS generates the parameter names with the _Add prefix and appended with unique integer suffixes. You can also use the PVAR statement specification to override these default variance parameters in situations where you want to specify parameter constraints, provide initial or fixed values, or make parameter references.

In the FACTOR or PATH model, a variable can either be exogenous or endogenous. Therefore, the default free parameters covers all the possible variance or partial variance parameters in the model. There are no default fixed zeros for any variances or partial variances in the model.

Modifying a Variance or Partial Variance Parameter Specification from a Reference Model

If you define a new FACTOR or PATH model by using a reference (old) model in the REFMODEL statement, you might want to modify some parameter specifications from the PVAR statement of the reference model before transferring the specifications to the new model. To change a particular variance or partial variance specification from the reference model, you can simply respecify the same variance or partial variance with the desired parameter specification in the PVAR statement of the new model. To delete a particular variance parameter from the reference model, you can specify the desired variance or partial variance with a missing value specification in the PVAR statement of the new model.

For example, suppose that the variance of V1 is specified in the reference PATH model but you do not want this variance specification to be transferred to the new model. You can use the following PVAR statement specification in the new model:

```
pvar
   V2 = .;
```

Note that the missing value syntax is valid only when you use the REFMODEL statement. See the section "Modifying a FACTOR Model from a Reference Model" on page 1080 for a more detailed example of FACTOR model respecification. See the section "Modifying a PATH Model from a Reference Model" on page 1145 for a more detailed example of PATH model respecification.

As discussed the section "Default Partial Variance Parameters" on page 1150, PROC CALIS generates default free variance parameters for the exogenous variables and default free error variance parameters for the endogenous variables in the confirmatory FACTOR or PATH model. When you use the REFMODEL statement for defining a reference model, these default free variance parameters in the old (reference) model are not transferred to the new model. Instead, the new model generates its own set of default free variance parameters *after* the new model is resolved from the reference model, the REFMODEL statement options, the RENAMEPARM statement, and the PVAR statement specifications in the new model. If you want any of the variance or error (partial) variance parameters to be constrained across the models by means of the REFMODEL specification, you must specify them explicitly in the PVAR statement of the reference model so that the same variance or error (partial) variance specification is transferred to the new model.

RAM Statement

RAM < **VAR=***variable-list* | **[** *variable-list=number-list* < *, variable-list=number-list . . .* > **]***,* > < *ram-entry* < *, ram-entry . . .* >> **;**

where *variable-list* is a list of variables for the rows and columns of the _A_ and _P_ matrices and the rows of the _W_ vector of the RAM model, *number-list* is a list of positive integers that denote the order of the specified variables, and *ram-entry* is a parameter specification for an element in one of the three RAM model matrices. You can specify latent variables in addition to observed variables in the VAR= option.

RAM stands for the reticular action model developed by McArdle (1980). The RAM model implemented in PROC CALIS extends the original RAM model with the specification of the mean vector in the _W_ vector. See the section "The RAM Model" on page 1229 for details about the model.

The RAM statement specification consists of the list of the variables in the model and the parameters and their locations the RAM model matrices. For example, consider the following simple RAM model specification:

```
ram  var= x1-x2 y3,
     _A_ 3 1,
     _A_ 3 2;
```

In this statement,variables x1, x2, and y3 are specified in the VAR= option. The variable order in the VAR= option is important. The same variable order applies to the rows and columns of the _A_ matrix. Next, there are two *ram-entries*. The first *ram-entry* specifies that the third variable (y3) has a path from the first variable (x1). Similarly, the second *ram-entry* specifies that y3 has a path from x2.

Specifying the VAR= Option

In the VAR= option, you specify the list of observed and latent variables in the RAM model. There are two ways to specify the VAR= list. The first way is a simple listing of variables. For example, you specify a total of 18 variables in the RAM model in the following statement:

```
ram  var= a b c x1-x5 y1-y10;
```

The order of the variables in this VAR= list is important. The same variable order applies to the rows and columns of the _A_ and _P_ matrices and the rows of the _W_ matrices. Although it is not required to arrange the variables according to whether they are observed or latent in the VAR= list, you can do so for your own convenience. PROC CALIS checks each variable in the VAR= list against the associated data sets to determine whether the variable is observed or latent.

When you specify the parameters in the *ram-entries*, you represent variables by the variable numbers that refer to the VAR= list. Therefore, it is important to make correct association of the variables and their order on the VAR= list. To this end, you can add some comments in your VAR= list to make the variable numbers explicit. For example,

```
ram  var= a      /* 1 */
          b      /* 2 */
          c      /* 3 */
          x1-x5  /* 4-8 */
          y1-y10 /* 9-18 */;
```

Another way to specify VAR= list is to provide the *variable-lists* together with explicit ordering indicated in the *number-lists*. For example, in the following statement you specify exactly the same variable list as that in the preceding example:

```
ram  var= [x1-x5 = 4 to 8, c = 3,  y1-y10 = 9 to 18, a = 1, b = 2];
```

Apart from showing how you can construct the VAR= list in a very general way with the *number-lists*, there is no particular reason why the *variable-lists* in the preceding specification are not in either an ascending or a descending order. Perhaps a more natural and useful way to use this type of explicit ordering specification is to place variables in exactly the same order as intended. For example, the following VAR= specification serves as a “key” of the variable numbers in the subsequent *ram-entries*:

```
ram  var= [x1 = 1, x2 = 2, y1 = 3, y2 = 4, y3 = 5],
     _A_   1   2 ,
     _P_   2   2 ;
```

With reference to the explicit variable numbers in the VAR= list, you can interpret the **_A_[1,2]** specification immediately as the effect from x2 to x1, and the **_P_[2,2]** specification as the variance of x2.

If the VAR= option is not specified in the RAM statement, the *n* observed variables in the VAR statement are used as the first *n* variables in the VAR= list. If you specify neither the VAR= option in the RAM statement nor the VAR statement, all *n* numerical variables in the associated data sets serve as the first *n* variables in the RAM model matrices. If there are more than *n* variables used in the *ram-entries*, the extra variables are all treated as latent variables in the RAM model.

Latent variables generated by PROC CALIS for the RAM model are named in two different ways, depending on whether your RAM model is specified under a MODEL statement. If you do not use the MODEL statement (for example, in situations with single-group analyses), latent variables are named _Factor1, _Factor2, and so on. If your RAM model is define within the scope of a MODEL statement, latent variables are named _Mdlk_F1, _Mdlk_F2, and so on, where k is substituted with the model number that is specified in the MODEL statement. For example, _Mdl2_F1 is a latent factor that is specified under a RAM model within the scope of the MODEL statement with 2 as its model number.

Because data sets might contain nonnumerical variables, implicit variable ordering deduced from the data sets is sometimes obscured. Therefore, it is highly recommended that you use the VAR= option to list all the variables in the RAM model.

Specifying a *ram-entry*

matrix-name | *matrix-number* *row-number* *column-number* < *parameter-spec* >

A *ram-entry* is a parameter specification of a matrix element of the RAM model. In each *ram-entry*, you first specify the matrix by using either the *matrix-name* or the *matrix-number*. Then you specify the *row-number* and the *column-number* of the element of the matrix. At the end of the *ram-entry*, you can optionally specify various kinds of parameters in *parameter-spec*. You can specify as many *ram-entries* as needed in your RAM model. *Ram-entries* are separated by commas. For example, consider the following specification:

```
ram  var= x1-x2 y3,
     _A_ 3 1  1.,
     _A_ 3 2;
```

You specify three variables in the VAR= option of the RAM statement. In the first *ram-entry*, variable y3 has a path from variable x1 with a fixed path coefficient 1. In the second *ram-entry*, variable y3 has a path from variable x2. Because the *parameter-spec* is blank, the corresponding path coefficient (or the effect from x2 on y3) is a free parameter by default.

Specifying the matrix-name or matrix-number

The three model matrices in the RAM model are: _A_, _P_, and _W_. See the section "The RAM Model" on page 1229 for the mathematical formulation of the RAM model. The *matrix-name* or *matrix-number* specifications in the *ram-entries* refer to these model matrices. You can use the following keywords for *matrix-name* or *matrix-number*:

_A_, _RAMA_, or 1   for the elements in the **A** matrix, which is for path coefficients or effects

_P_, _RAMP_, or 2   for the elements in the **P** matrix, which is for variances and covariances

_W_, _RAMW_, or 3   for the elements in the **W** vector, which is for intercepts and means

Specifying the row-number and column-number

After you specify the *matrix-name* or *matrix-number* in a *ram-entry*, you need to specify the *row-number* and *column-number* that correspond to the intended element of the matrix being specified.

Specifying the parameter-spec

You can specify three types of parameters in *parameter-spec*:

- A free parameter without an initial estimate: blank or *parameter-name*

  You can specify a free parameter for the matrix element in a *ram-entry* by either omitting the *parameter-spec* (that is, leaving it blank) or specifying a *parameter-name*. For example, both of the following *ram-entries* specify that **_A_[3,1]** is a free parameter in the RAM model:

  ```
  _A_   3   1
  ```

  and

  ```
  _A_   3   1   beta
  ```

  The difference is that in the latter you name the effect (path coefficient) for the **_A_[3,1]** element as beta, while in the former PROC CALIS generates a free parameter name (prefixed with _Parm and followed by a unique parameter number) for the specified element. Leaving the *parameter-spec* blank is handy if you do not need to refer to this parameter in your code. But when you need to specify parameter constraints by referring to parameter names, the *parameter-name* syntax becomes necessary. For example, the following specification constrains the **_A_[3,1]** and **_A_[3,2]** paths to have equal effects (path coefficients) because they have the same *parameter-name* beta:

  ```
  ram  var= x1-x2 y3,
       _A_ 3 1  beta,
       _A_ 3 2  beta;
  ```

- A free parameter with an initial estimate: *(number)* or *parameter-name (number)*

  You can specify a free parameter with an initial estimate in a *ram-entry* by either specifying the initial estimate within parentheses or specifying a *parameter-name* followed by the parenthesized initial estimate. For example, both of the following *ram-entries* specify that **_A_[3,1]** is a free parameter with an initial estimate of 0.3 in the RAM model:

```
_A_  3  1  (0.3)
```

and

```
_A_  3  1  beta  (0.3)
```

In the latter you name the effect (path coefficient) for the `_A_[3,1]` element as beta, while in the former PROC CALIS generates a free parameter name (prefixed with _Parm and followed by a unique parameter number). The latter syntax is necessary when you need to specify parameter constraints by referring to the parameter name beta. The former syntax is more convenient when you do not need to refer to this parameter in other specifications.

For the latter syntax with a *parameter-name* specified, you can omit the pair of parentheses or exchange the position of *parameter-name* and *number* (or both) without changing the nature of the parameter. That is, you can use the following equivalent specifications for a named free parameter with initial values:

```
_A_  3  1  beta  0.3
```

and

```
_A_  3  1  .3  beta
```

- A fixed parameter value: *number*

You can specify a fixed value by simply providing it as the *parameter-spec* in a *ram-entry*. For example, in the following syntax you specify that `_A_[3,1]` is a fixed value of 0.3:

```
_A_  3  1  0.3
```

The fixed value for _A_[3,1] does not change during the estimation. To distinguish this syntax from the initial value specification, notice that you do not put 0.3 inside parentheses, nor do you put a *parameter-name* before or after the provided value.

Notes and Cautions about Specifying *ram-entries*

- Older versions of PROC CALIS treat a blank *parameter-spec* in the *ram-entry* as a fixed constant 1. This is no longer the case in this version of PROC CALIS. Fixed values such as 1.0 must be specified explicitly.
- The *row-number* and *column-number* in the *ram-entries* refer to the VAR= variable list of the RAM statement. An exception is for the _W_ vector, of which the *column-number* should always be 1 and does not refer to any particular variable.
- When a *row-number* or *column-number* in a *ram-entry* (except for the *column-number* of _W_) does not have any reference in the VAR= variable list (or is greater than the number of default observed variables when the VAR= option is not specified), PROC CALIS treats the corresponding row or column variable as a latent variable and generates variable names for it.

- The largest row or column number used in any *ram-entry* should not exceed the sum of observed and latent variables intended in the RAM model. Otherwise, some extraneous latent variables might be created.

Default Parameters

It is important to understand the default parameters in the RAM model. First, if you know which parameters are default free parameters, you can make your specification more efficient by omitting the specifications of those parameters that can be set by default. For example, because all exogenous variances and error variances in the RAM model are free parameters by default, you do not need to specify the diagonal elements of the _P_ matrix if they are not constrained in the model. Second, if you know which parameters are default free parameters, you can specify your model accurately. For example, because all the error covariances in the RAM model are fixed zeros by default, you must specify the corresponding off-diagonal elements of the _P_ matrix in the *ram-entries*. See the section "Default Parameters in the RAM Model" on page 1237 for details about the default parameters of the RAM model.

Modifying a RAM Model from a Reference Model

This section assumes that you use a REFMODEL statement within the scope of a MODEL statement and that the reference model (or base model) is also a RAM model. The reference model is called the old model, and the model that refers to this old model is called the new model. If the new model is not intended to be an exact copy of the old model, you can use the following extended RAM modeling language to make modifications on the model specification. The syntax for modifications is very much the same as the ordinary RAM modeling language (see the section "RAM Statement" on page 1151), except that you cannot specify the VAR= option in the RAM statement. The reason is that the VAR= variable list in the new RAM model should be exactly the same as the old model; otherwise, the *row-number* and *column-number* in the *ram-entries* would not have the same references and thus would make model referencing meaningless. Hence, the syntax for respecifying (modifying) the RAM model contains only the *ram-entries*:

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

The syntax of the *ram-entry* is the same as that of the original RAM statement, with an addition of the missing value specification for the *parameter-spec*, which denotes the deletion of a parameter location.

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

Duplication: If you do not specify in the new model a parameter location (matrix element) that exists in the old model, the old parameter specification is duplicated in the new model.

Addition: If you specify in the new model a parameter location (matrix element) that does not exist in the old model, the new parameter specification is added to the new model.

Deletion: If you specify in the new model a parameter location (matrix element) that also exists in the old model and the new *parameter-spec* is denoted by the missing value '.', the old parameter specification is not copied into the new model.

Replacement: If you specify in the new model a parameter location (matrix element) that also exists in the old model and the new parameter is not denoted by the missing value '.', the new parameter specification replaces the old one in the new model.

For example, consider the following two-group analysis:

```
proc calis;
   group 1 / data=d1;
   group 2 / data=d2;
   model 1 / group=1;
      ram
         var = [V1-V6 = 1 to 6, F1 = 7],
         _A_ 1 7  1.,
         _A_ 2 7  load1,
         _A_ 3 7  load2,
         _A_ 7 4  ,
         _A_ 7 5  ,
         _A_ 7 6  ,
         _P_ 1 1  ,
         _P_ 2 2  ,
         _P_ 3 3  ,
         _P_ 7 7  ,
         _P_ 4 4  ,
         _P_ 5 5  ,
         _P_ 6 6  ,
         _P_ 1 2  cve12;
   model 2 / group=2;
      refmodel 1;
      ram
         _A_ 3 7  load1,
         _P_ 1 2      .,
         _P_ 2 3  cve23;
   run;
```

In this example, you specify Model 2 by referring to Model 1 in the REFMODEL statement. Model 2 is the new model which refers to the old model, Model 1. This example illustrates the four types of model integration process by PROC CALIS:

- Duplication: All parameter specifications, except for **_A_[3,7]** and **_P_[1,2]**, in the old model are duplicated in the new model.
- Addition: The new parameter cve23 is added for the matrix element **_P_[2,3]** in the new model.
- Deletion: The parameter location **_P_[1,2]** and associated parameter cve12 are not copied into the new model, as indicated by the missing value '.' in the new model specification.
- Replacement: The **_A_[3,7]** path in the new model replaces the same path in the old model with another parameter for the path coefficient. As a results, in the new model paths specified by **_A_[3,7]** and **_A_[2,7]** are constrained to have the same path coefficient parameter load1.

PROC CALIS might have generated some default parameters (named with the '_Add' prefix) for the old (reference) model. These default parameters in the old (reference) model do *not* transfer to the new model. Only after the new model is resolved from the reference model, the REFMODEL statement options, the RENAMEPARM statement, and the model respecification are the default parameters of the new RAM model generated. In this way, the generated parameters in the new model are not constrained to be the same as the corresponding parameters in the old (reference) model. If you want any of these default parameters to be constrained across the models, you must specify them explicitly in the *ram-entries* of the RAM statement of the reference model so that these specifications are duplicated to the new model via the REFMODEL statement.

REFMODEL Statement

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

The REFMODEL statement is not a modeling language itself. It is a tool for referencing and modifying models. It is classified into one of the modeling languages because its role is similar to other modeling languages.

REFMODEL *model_number* < / *options* > **;**
RENAMEPARM *parameter renaming* **;**
main model specification statement **;**
subsidiary model specification statements **;**

In the REFMODEL statement, you specify the *model_number* (between 1 and 9,999, inclusive) of the model you are making reference to. The reference model must be well-defined in the same PROC CALIS run. In the *options*, you can rename all the parameters in the reference model by adding a prefix or suffix so that the current model has a new set of parameters. The RENAMEPARM statement renames individual parameters in the reference model to new names. In the main model specification statement and the subsidiary model specification statements, you can respecify or modify the specific parts of the reference model. The specification of these statements must be compatible with the model type of the reference model.

NOTE: The REFMODEL statement does *not* simply copy model specifications from a reference model. If you do not change any of the parameter names of the reference model by any of the REFMODEL statement options, the REFMODEL statement copies only the *explicit* specifications from the reference model to the new model. However, the REFMODEL statement does not copy the default parameters from the reference model to the new model. For example, consider the following statements:

```
proc calis;
   group 1 / data=a1;
   group 2 / data=a2;
   model 1 / group=1;
      path x1 ---> x2;
   model 2 / group=2;
      refmodel 1;
run;
```

In this example, Model 2 makes reference to Model 1. This means that the path relationship between x1 and x2 as specified in Model 1 is exactly the same path relationship you want Model 2 to have. The path coefficients in these two models are constrained to be the same. However, the variance parameter of x1 and the error variance parameter for x2 are not constrained in these models. Rather, these two parameters are set by default in these models separately. If you intend to constrain all parameters in the two models, you can specify all the parameters in Model 1 explicitly and use the REFMODEL statement for Model 2, as shown in the following statements:

```
proc calis;
   group 1 / data=a1;
   group 2 / data=a2;
   model 1 / group=1;
      path x1 ---> x2;
      pvar x1 x2;
   model 2 / group=2;
      refmodel 1;
run;
```

This way Model 2 makes reference to all the explicitly specified parameters in Model 1. Hence the two models are completely constrained. However, a simpler way to fit exactly the same model to two groups is to use a single model definition, as shown in the following statements:

```
proc calis;
   group 1 / data=a1;
   group 2 / data=a2;
   model 1 / group=1,2;
      path x1 ---> x2;
run;
```

This specification has the same estimation results as those for the preceding specification.

When you also use one of the REFMODEL statement options, the REFMODEL statement is no longer a simple copy of explicit parameter specifications from the reference model. All parameters are renamed in the new model in the model referencing process. The following 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 VARIANCE statement—except that in the RENAMEPARM statement, you put parameter names on the left-hand side of equal signs, whereas you put variable names on the left-hand side in the VARIANCE statement. You can use no more than one RENAMEPARM statement within the scope of each REFMODEL statement.

In the REFMODEL statement, you transfer all the model specification information from a base model to the new model being specified. The RENAMEPARM statement enables you to modify the parameter names or types in the base model before transferring them to the new model. For example, in the following example, you define Model 2, which is a new model, by referring it to Model 1, the base model, in the REFMODEL statement.

```
model 1;
   lineqs
      V1 =    F1 + E1,
      V2 = b2 F1 + E2,
      V3 = b3 F1 + E3,
      V4 = b4 F1 + E4;
   variance F1 = vF1,
      E1-E4 = ve1-ve4;
model 2;
   refmodel 1;
   renameparm ve1-ve4=new1-new4, b2=newb2(.2), b4=.3;
```

Basically, the LINEQS model specification in Model 1 is transferred to Model 2. In addition, you redefine some parameters in the base model by using the RENAMEPARM statement. This example illustrates two kinds of modifications that the RENAMEPARM statement can do:

- creating new parameters in the new model

  The error variances for E1–E4 in Model 2 are different from those defined in Model 1 because new parameters new1–new4 are now used. Parameter b2 is renamed as newb2 with a starting value at 0.2 in Model 2. So the two models have distinct path coefficients for the F1-to-V2 path.

- changing free parameters into fixed constants

  By using the specification b4=.3 in the RENAMEPARM statement, b4 is no longer a free parameter in Model 2. The path coefficient for the F1-to-V4 path in Model 2 is now fixed at 0.3.

The RENAMEPARM statement is handy when you have just few parameters to change in the reference model defined by the REFMODEL statement. However, when there are a lot of parameters to modify, the RENAMEPARM statement might not be very efficient. For example, to make all parameters unique to the current model, you might consider using the ALLNEWPARMS, PARM_PREFIX=, or PARM_SUFFIX= option in the REFMODEL statement.

SAS Programming Statements

You can use SAS programming statements to define dependent parameters, parametric functions, and equality constraints among parameters.

Several statistical procedures support the use of SAS programming statements. The syntax of SAS programming statements are common to all these procedures and can be found in the section "Programming Statements" on page 519 in Chapter 19, "Shared Concepts and Topics."

SIMTESTS Statement

SIMTESTS | SIMTEST *sim_test* < *sim_test* . . . > **;**

where *sim_test* represents one of the following:

- *test_name* = [*functions*]
- *test_name* = { *functions* }

and *functions* are either parameters in the model or parametric functions computed in the SAS programming statements.

When the estimates in a model are asymptotically multivariate-normal, continuous and differentiable functions of the estimates are also multivariate-normally distributed. In the SIMTESTS statement, you can test these parametric functions simultaneously. The null hypothesis for the simultaneous tests is assumed to have the following form:

$H_0 : h_1(\theta) = 0, h_2(\theta) = 0, \ldots$

where 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 SIMTESTS statement, you first assign a name for the simultaneously test in *test_name*. Then you put the parametric functions for the simultaneous test inside a pair of parentheses: either the '{' and '}' pair, or the '[' and ']' pair. For example, if 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:

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

In the SIMTESTS statement, you test two functions t1_t2_diff and t3_t4_diff simultaneously in the test named Equality_test. The two parametric functions t1_t2_diff and t3_t4_diff are computed in the SAS programming statements as differences of some parameters in the model.

See also the TESTFUNC statement on page 1163 for testing parametric functions individually.

STD Statement

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

where *assignment* represents:

var_list = *parameter-spec*

The STD statement functions exactly the same as the VARIANCE statement. The STD statement is obsolete and might not be supported in future versions of PROC CALIS. Use the VARIANCE statement instead.

STRUCTEQ Statement

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 SIMTESTS statement if you want to test several null hypotheses simultaneously.

VAR Statement

VAR *variables* **;**

The VAR statement defines and limits the set of observed variables that are available for the corresponding model analysis. 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 CALIS statement. The set of variables in the VAR statement must be present in the data set specified in the associated GROUP or the PROC CALIS 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 *var_list* in the optimization context. You have to make sure that all variables specified in the VAR statement refer 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.

In some modeling languages of PROC CALIS, you can also specify the observed variables either directly (for example, through the VAR= or similar option in some main model specification statements) or indirectly (for example, through the specification of functional relationships between observed variables). How does the VAR statement specifications interplay with the observed variables specified in the model? This depends on the types of models specified. Four different cases are considered in the following.

Case 1. Exploratory Factor Models With No VAR= option in the FACTOR statement. For exploratory factor models specified using the FACTOR statement, it is important for you to use the VAR statement to select and limit the set of the observed variables for analysis. The reason is simply that there is no other options in the FACTOR statement that will serve the same purpose. For example, you analyze only v1–v3 in the following exploratory factor model even though there might be more observed variables available in the data set:

```
proc calis;
   var v1-v3;
   factor  n=1;
```

If you do not specify the VAR statement, PROC CALIS simply selects all numerical variables for analysis. However, to avoid confusions it is a good practice to specify the observed variables explicitly in the VAR statement.

Case 2. Models With a VAR= or Similar Option for Defining the Set of Observed Variables for Analysis. The classes of models considered here are: COSAN, LISMOD, MSTRUCT, and RAM. Except for the LISMOD models, in all other three classes of models you can specify the observed variables in the model by using the a VAR= option in the respective main model specification statement. For the LISMOD models, you can specify all observed variables that should be included in the model in the XVAR= and YVAR= options of the LISMOD statement. Therefore, the use of the VAR statement for these models might become unnecessary. For example, the following MSTRUCT statement specifies the observed variables v1–v6 in the VAR= option:

```
proc calis;
   mstruct  var=v1-v6;
```

It would have been redundant to use a VAR statement to specify v1–v6 additionally. The same conclusion applies to the COSAN and the RAM models.

Another example is when you specify a LISMOD model. In the following LISMOD specification, variables v1–v8 would be the set of observed variables for analysis:

```
proc calis;
   var v1-v8;
   lismod  xvar = v1-v4,
           yvar = v5-v8,
           eta  = factor1,
           xi   = factor2;
```

Again, there is no need to add a VAR statement merely repeating the specification of variables v1–v8.

If you do specify the VAR statement in addition to the specification of variable lists in these models, PROC CALIS will check the consistency between the lists. Conflicts arise if the two lists do not match.

For example, the following statements will generate an error in model specification because v6 specified in the MSTRUCT model is not defined as an observed variable available for analysis in the VAR statement (even if v6 might indeed be present in the data set):

```
proc calis;
   var v1-v5;
   mstruct  var=v1-v6;
```

So it is an error when you specify fewer observed variables in the VAR statement than in the VAR= option in the model. How about if you specify more variables in the VAR statement? PROC CALIS will also general an error because the extra variables in VAR statement will not be well-undefined in the model. For example, v7–v10 specified in the VAR statement are supposed to be included into the model, but they not listed on either the XVAR= or YVAR= list in the following LISMOD statement:

```
proc calis;
   var v1-v10;
   lismod  xvar = v1-v3,
           yvar = v4-v6,
           eta  = factor1,
           xi   = factor2;
```

Therefore, if you must specify the VAR statement for these models, the specifications of the observed variables must be consistent in the VAR statement and in the relevant model options. However, to avoid potential conflicts in these situations, you are recommended to specify the observed variables in the VAR=, XVAR=, or YVAR= lists only.

When the VAR= option is not specified in the COSAN, MSTRUCT, or RAM statement, the VAR statement specification will be used as the list of observed variables in the model. If both of the VAR= option and VAR statement specification are lacking, then all numerical variables in the associated data set will be used in the model. However, to avoid confusions the preferred method is to specify the list of observed variables explicitly on the VAR=, XVAR=, or YVAR= option of the main model specification statements.

Case 3. Models With Certain Indirect Ways to Include the Set of Observed Variables for Analysis. Two types of models are considered here: LINEQS and PATH. For these models, the main use of the VAR statement is to include those observed variables that are not mentioned in model specifications.

For example, in the following statements for a LINEQS model variable v3 is not mentioned in the LINEQS statement:

```
proc calis;
   var v1-v3;
   lineqs   v1 = a1 * v2 + e1;
```

With the specification in the VAR statement, however, variable v3 is included into the model as an exogenous manifest variable. Similarly, the same applies to the following PATH model specification:

```
proc calis;
   var v1-v3;
   path   v1 <- v2;
```

Again, variable v3 is included into the PATH model because it is specified in the VAR statement.

The two preceding examples also suggest that you do not need to use the VAR statement when your already mentions all observed variables in the model specification. For example, if your target set of observed variable are v1–v3, the use of the VAR statement in the following specification is *unnecessary*:

```
proc calis;
   var v1-v3;
   path   v1 <- v2;
      pvar v3;
```

For the two types of models considered here, you can also use the VAR statement to define and limit the set of observed variables for analysis. For example, you might have v1, v2, v3 in your data set as observed variables for analysis; but somehow in your model v2 should be treated as a latent variable. You might use the following code to exclude v2 as an observed variable in the model:

```
proc calis;
   var v1  v3;
   path   v1 <- v2;
      pvar v3;
```

The role of the VAR statement here is to define and limit the set of observed variables available for the model. Hence, only variables v1 and v3 are supposed to be observed variables in the model while variable v2 in the PATH model is treated as latent.

In sum, in the current situation the use of the VAR statement should depend on whether a variable should or should not be included as an observed variable in your theoretical model.

Case 4. Confirmatory Factor Model With the FACTOR statement. In this case, the VAR statement still limits the set of observed variables being analyzed in the confirmatory factor model. However, because all observed variables in a confirmatory factor analysis must be loaded on (or related to) some factors through the specification of *factor-variable-relations* in the FACTOR statement, all observed variables in the model should have been specified (or mentioned) in the FACTOR statement already, making it redundant to use the VAR statement for the same purpose.

VARIANCE Statement

VARIANCE *assignment* < , *assignment . . .* > **;**

where *assignment* represents:

var_list < **=***parameter-spec*>

The VARIANCE statement specifies the variance parameters in connection with the LINEQS model. Notice that the VARIANCE statement is different from the VAR statement, which specifies variables for analysis. In previous versions of PROC CALIS, the STD statement name was used instead of the VARIANCE statement name. Although these two names result in the same functionalities, the VARIANCE statement name reflects the intended usages better.

In the LINEQS model, variance parameters are defined only for *exogenous* manifest and latent variables (including error and disturbance variables) in the model. Therefore, you cannot list any *endogenous* variables in the *var_list* list of the VARIANCE statement. You can specify no more than one VARIANCE statement for each LINEQS model.

In each *assignment* of the VARIANCE statement, you list the *var_list* whose variances you want to specify. Optionally, you can provide a list of parameter specifications (*parameter-spec*) after an equal sign for each *var_list* list.

You can specify the following five types of the parameters for the variances of the exogenous variables in the VARIANCE statement:

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

Consider a LINEQS model with exogenous variables V1, V2, F1, D2, and E3. The following VARIANCE statement illustrates the five types of parameter specifications in five *assignments*:

```
variance
   V1 ,
   V2 = (.5),
   F1 = 1.0,
   D2 = dvar,
   E3 = evar(0.7);
```

In this statement, the variance of V1 is specified as an unnamed free parameter. For this variance, PROC CALIS generates a parameter name with the _Parm prefix and appended with a unique integer (for example, _Parm1). The variance of V2 is an unnamed free parameter with an initial value of 0.5. PROC CALIS also generates a parameter name for this variance. The variance of F1 is a fixed value of 1.0. This value stays the same during the estimation. The variance of D2 is a free parameter named dvar. The variance of E3 is a free parameter named evar with an initial value of 0.7.

When you need to specify a long parameter name list, you can consider using the prefix-name specification for the parameter list. For example, the following statement specifies 100 unique parameter names for the variances of E1–E100:

```
variance
   E1-E100 = 100 * evar__; /* evar with two trailing underscores */
```

In the VARIANCE statement, evar_ _ is a prefix-name with the root evar. The notation **100*** means this prefix-name is applied 100 times, resulting in a generation of the 100 unique parameter names evar001, evar002, . . . , evar100.

The root of the prefix-name should have few characters so that the generated parameter name is not longer than 32 characters. To avoid unintentional equality constraints, the prefix-names should not coincide with other parameter names.

Mixed Parameter Lists

You can specify different types of parameters for the list of variances. For example, the following statement uses a list of parameters with mixed types:

```
variance
   E1-E6 = vp1 vp2(2.0) vp3  4. (.3) vp6(.4);
```

This is equivalent to the following specification:

```
variance
   E1 = vp1
   E2 = vp2(2.0),
   E3 = vp3,
   E4 = 4. ,
   E5 = (.3),
   E6 = vp6(.4);
```

Notice that an initial value followed after a parameter name is associated with the free parameter. For example, in the original mixed list specification, the specification **(2.0)** after vp2 is interpreted as the initial value for the parameter vp2, but not as the initial estimate for the variance of E3.

However, if you indeed want to specify that vp2 is a free parameter *without* an initial value and 2.0 is an initial estimate for the variance of E3 (while keeping all other things the same), you can use a null initial value specification for the parameter vp2, as shown in the following statement:

```
variance
   E1-E6 = vp1 vp2() (2.0)  4. (.3) vp6(.4);
```

This way 2.0 becomes the initial estimate for the variance of E3. Because a parameter list with mixed types might be confusing, you can break down the specifications into separate *assignments* to remove ambiguities. For example, you can use the following equivalent specification:

```
variance
   E1 = vp1
   E2 = vp2,
   E3 = (2.),
   E4 = 4. ,
   E5 = (.3),
   E6 = vp6(.4);
```

Shorter and Longer Parameter Lists

If you provide fewer parameters than the number of variances in the *var_list* list, all the remaining parameters are treated as unnamed free parameters. For example, the following specification assigns a fixed value of 1.0 to the variance of F1 while treating the other three variances as unnamed free parameters:

```
variance
   F1-F4 = 1.0;
```

This specification is equivalent to the following specification:

```
variance
   F1 = 1.0, F2-F4;
```

If you intend to fill up all values with the last parameter specification in the list, you can use the continuation syntax `[...]`, `[..]`, or `[.]`, as shown in the following example:

```
variance
   E1-E100 = 1.0 psi [...];
```

This means that the variance of E1 is fixed at 1.0, while the variances of E1–E100 are all free parameter named psi. All variances except that for E1 are thus constrained to be equal by using the same parameter name.

However, you must be careful not to provide too many parameters. For example, the following specification results in an error:

```
variance
   E1-E6 = 1.0 psi2-psi6 extra;
```

The parameters after psi6 are excessive.

Default Variance Parameters

In the LINEQS model, by default all variances of exogenous manifest and latent variables (including error and disturbance variables) are free parameters. For these default free parameters, PROC CALIS generates the parameter names with the _Add prefix and appended with unique integer suffixes. You can also use the VARIANCE statement specification to override these default variance parameters in situations where you want to specify parameter constraints, provide initial or fixed values, or make parameter references.

Because only exogenous variables can have variance parameters in the LINEQS model and all these exogenous variances are free parameters by default, there are no default fixed zeros for any variances in the LINEQS model.

Modifying a Variance Parameter Specification from a Reference Model

If you define a new LINEQS model by using a reference (old) model in the REFMODEL statement, you might want to modify some parameter specifications from the VARIANCE statement of the reference model before transferring the specifications to the new model. To change a particular variance specification from the reference model, you can simply respecify the same variance with the desired parameter specification in the VARIANCE statement of the new model. To delete a particular variance parameter from the reference model, you can specify the desired variance with a missing value specification in the VARIANCE statement of the new model.

For example, suppose that the variance of V1 is specified in the reference model but you do not want this variance specification to be transferred to the new model, you can use the following VARIANCE statement specification in the new model:

```
variance V1 = .;
```

Note that the missing value syntax is valid only when you use the REFMODEL statement. See the section "Modifying a LINEQS Model from a Reference Model" on page 1094 for a more detailed example of the LINEQS model respecification.

As discussed in a preceding section, PROC CALIS generates default free variance parameters for the LINEQS model if you do not specify them explicitly in the VARIANCE statement. When you use the REFMODEL statement for defining a reference model, these default free variance parameters in the old (reference) model are not transferred to the new model. Instead, the new model generates its own set of default free variance parameters *after* the new model is resolved from the reference model, the REFMODEL statement options, the RENAMEPARM statement, and the VARIANCE statement specifications in the new model. This also implies that if you want any of the variance parameters to be constrained across the models by means of the REFMODEL specification, you must specify them explicitly in the VARIANCE statement of the reference model so that the same variance specification is transferred to the new model.

VARNAMES Statement

VARNAMES *name_assignment* < , *name_assignment* . . . > **;**

VARNAME *name_assignment* < , *name_assignment* . . . > **;**

VNAMES *name_assignment* < , *name_assignment* . . . > **;**

where *name_assignment* represents one of the following forms:

matrix_name variable_names
matrix_name = [*variable_names*]
matrix_name = *matrix_name*

You can use the VARNAMES statement in connection with the COSAN modeling language to assign variable names for matrices. The *matrix_name* refers to any matrix you define in the COSAN statement. The *variable_names* that follow the *matrix_name* are assigned to the column variables of the matrix of interest. This applies to the first two types of VARNAMES specifications. For example,

```
varnames  F   f1-f3;
```

is exactly the same as

```
varnames  F = [ f1-f3 ];
```

Both of these assign f1, f2, and f3 as the names for the first three column variables of matrix F.

You can also use another kind of *name_assignment* in connection with a COSAN statement. Two matrix names equated by an equal sign assign the column names of the matrix on the right-hand side to the column names of the matrix on the left-hand side. This assignment assumes that the column names of at least one of the two matrices are already defined. For example, assuming that **J** and **A** are model matrices defined in a COSAN statement, the following VARNAMES statement specification specifies that both **J** and **A** have the same set of column variable names V1–V6 and F1–F3:

```
varnames  J = [ V1-V6 F1-F3 ] ,
          A = J ;
```

This is the same as the following specification:

```
varnames  J = [ V1-V6 F1-F3 ] ,
          A = [ V1-V6 F1-F3 ] ;
```

The VARNAMES statement appears to enable you to specify only the column variable names for matrices. However, PROC CALIS also uses these column variable names to assign row variable names of the related matrices in the covariance and mean structure formulas for the COSAN model. PROC CALIS uses the following rules to determine the row variable names of a matrix in the model:

- If a matrix is the first matrix of any term in the covariance or mean structure formula, the row variable names are the names of the manifest variables.
- If a matrix is the central covariance matrix of any term in the covariance structure formula, the row variable names are the same as the column variable names.
- For any other matrices, the row variable names are the same as the column variable names of the preceding matrix in the multiplicative formula for the covariance or mean structures.

WEIGHT Statement

WEIGHT *variable* **;**

The WEIGHT statement specifies the weight variable for the observations. It is one of the subsidiary group specification statements. You can use the WEIGHT statement no more than once within the scope of each GROUP statement or the PROC CALIS statement.

Weighting is often done when the error variance associated with each observation is different and the values of the weight variable are proportional to the reciprocals of the variances. The WEIGHT and FREQ statements have a similar effect, except the WEIGHT statement does not alter the number of observations unless VARDEF=WGT or VARDEF=WDF. An observation is used in the analysis only if the WEIGHT variable is greater than 0 and is not missing.

Details: CALIS Procedure

Input Data Sets

You can use four different kinds of input data sets in the CALIS procedure, and you can use them simultaneously. The DATA= data set contains the data to be analyzed, and it can be an ordinary SAS data set containing raw data or a special TYPE=COV, TYPE=UCOV, TYPE=CORR, TYPE=UCORR, TYPE=SSCP, or TYPE=FACTOR data set containing previously computed statistics. The INEST= data set specifies an input data set that contains initial estimates for the parameters used in the optimization process, and it can also contain boundary and general linear constraints on the parameters. If the model does not change too much, you can use an OUTEST= data set from a previous PROC CALIS analysis; the initial estimates are taken from the values of the _TYPE_=PARMS observation. The INMODEL= or INRAM= data set contains information of the analysis models (except for user-written programming statements). Often the INMODEL= data set is created as the OUTMODEL= data set from a previous PROC CALIS analysis. See the section "OUTMODEL= SAS-data-set" on page 1180 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 CALIS several times, you can save computer time by first creating a TYPE=COV, TYPE=UCOV, TYPE=CORR, or TYPE=UCORR data set and using it as input to PROC CALIS.

For example, assuming that PROC CALIS is first run with an OUTMODEL=MODEL option, you can run the following statements in subsequent analyses with the same model in the first run:

```
/* create TYPE=COV data set */
proc corr cov nocorr data=raw outp=cov(type=cov);
run;
/* analysis using correlations */
proc calis corr data=cov inmodel=model;
run;
/* analysis using covariances */
proc calis data=cov inmodel=model;
run;
```

Most procedures automatically set the TYPE= option of an output data set appropriately. However, the CORR procedure sets TYPE=CORR unless an explicit TYPE= option is used. Thus, (TYPE=COV) is needed in the preceding PROC CORR request, since the output data set is a covariance matrix. If you use a DATA step with a SET statement to modify this data set, you must declare the TYPE=COV, TYPE=UCOV, TYPE=CORR, or TYPE=UCORR attribute in the new data set.

You can use a VAR statement with PROC CALIS when reading a TYPE=COV, TYPE=UCOV, TYPE=CORR, TYPE=UCORR, or TYPE=SSCP data set to select a subset of the variables or change the order of the variables.

CAUTION: Problems can arise from using the CORR procedure when there are missing data. By default, PROC CORR computes each covariance or correlation from all observations that have values present for the pair of variables involved ("pairwise deletion"). The resulting covariance or correlation matrix can have negative eigenvalues. A correlation or covariance matrix with negative eigenvalues is recognized as a singular matrix in PROC CALIS, and you cannot compute (default) generalized least squares or maximum likelihood estimates. You can specify the RIDGE option to ridge the diagonal of such a matrix to obtain a positive definite data matrix. If the NOMISS option is used with the CORR procedure, observations with any missing values are completely omitted from the calculations ("listwise deletion"), and there is no possibility of negative eigenvalues (but there is still a chance for a singular matrix).

PROC CALIS can also create a TYPE=COV, TYPE=UCOV, TYPE=CORR, or TYPE=UCORR data set that includes all the information needed for repeated analyses.

If the data set DATA=RAW does not contain missing values, the following statements should give the same PROC CALIS results as the previous example:

```
/* using correlations */
proc calis corr data=raw outstat=cov inmodel=model;
run;
/* using covariances */
proc calis data=cov inmodel=model;
run;
```

You can create a TYPE=COV, TYPE=UCOV, TYPE=CORR, TYPE=UCORR, or TYPE=SSCP data set in a DATA step. Be sure to specify the TYPE= option in parentheses after the data set name in the DATA statement and include the _TYPE_ and _NAME_ variables. If you want to analyze the covariance matrix but your DATA= data set is a TYPE=CORR or TYPE=UCORR data set, you should include an observation with _TYPE_=STD giving the standard deviation of each variable. By default, PROC CALIS analyzes the recomputed covariance matrix even when a TYPE=CORR data set is provided, as shown in the following statements:

```
data correl(type=corr);
   input _type_ $ _name_ $ X1-X3;
   datalines;
std   .    4.   2.   8.
corr  X1   1.0  .    .
corr  X2    .7 1.0   .
corr  X3    .5  .4 1.0
;
proc calis inmodel=model;
run;
```

INEST= 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 CALIS model
- the BY variables that are used in a DATA= input data set
- a numeric variable _RHS_ (right-hand side); needed only if linear constraints are used
- additional variables with names corresponding to constants used in the programming statements

The content of the _TYPE_ variable defines the meaning of the observation of the INEST= data set. PROC CALIS recognizes observations with the following _TYPE_ specifications.

PARMS
: specifies initial values for parameters that are defined in the model statements of PROC CALIS. The _RHS_ variable is not used. Additional variables can contain the values of constants that are referred to in programming statements. At the beginning of each run of PROC CALIS, the values of the constants are read from the PARMS observation for initializing the constants in the SAS programming statements.

UPPERBD | UB
: specifies upper bounds with nonmissing values. The use of a missing value indicates that no upper bound is specified for the parameter. The _RHS_ variable is not used.

LOWERBD | LB
: specifies lower bounds with nonmissing values. The use of a missing value indicates that no lower bound is specified for the parameter. The _RHS_ variable is not used.

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 CALIS run as an INEST= data set in a new run. However, be aware that the OUTEST= data set also contains the boundary and general linear constraints specified in the previous run of PROC CALIS. When you are using this OUTEST= data set without changes as an INEST= data set, PROC CALIS adds the constraints from the data set to the constraints specified by a

BOUNDS and LINCON statement. Although PROC CALIS automatically eliminates multiple identical constraints, you should avoid specifying the same constraint a second time.

INMODEL= SAS-data-set

This data set is usually created in a previous run of PROC CALIS. It is useful if you want to reanalyze a problem in a different way such as using a different estimation method. You can alter an existing OUTMODEL= data set in the DATA step to create the INMODEL= data set that describes a modified model. See the section "OUTMODEL= SAS-data-set" on page 1180 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 $\mathbf{W}$. This property enables you to store more than one weight matrix in the INWGT= data set. You can then run PROC CALIS with each of the weight matrices by changing only the _TYPE_ observation in the INWGT= data set with an intermediate DATA step.

See the section "OUTWGT= SAS-data-set" on page 1190 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 CALIS for another analysis with either PROC CALIS or another SAS procedure. Saving results to an OUTEST= data set is advised for expensive applications that cannot be repeated without considerable effort.

The OUTEST= data set contains the BY variables, two character variables _TYPE_ and _NAME_, t numeric variables corresponding to the parameters used in the model, a numeric variable _RHS_ (right-hand side) that is used for the right-hand-side value b_i of a linear constraint or for the value $f = f(x)$ of the objective function at the final point x^* of the parameter space, and a numeric variable _ITER_ that is set to zero for initial values, set to the iteration number for the OUTITER output, and set to missing for the result output.

The _TYPE_ observations in Table 26.1 are available in the OUTEST= data set, depending on the request.

Table 26.1 _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
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).

Table 26.1 *continued*

TYPE	Description
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.
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

Table 26.1 *continued*

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.
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).

Table 26.1 *continued*

TYPE	Description
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).
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, the computed parameter estimates, and the standard error estimates. This data set is intended to be reused as an INMODEL= data set to specify good initial values in a subsequent analysis by PROC CALIS.

The OUTMODEL= data set contains the following variables:

- the BY variables, if any
- an _MDLNUM_ variable for model numbers, if used
- a character variable _TYPE_, which takes various values that indicate the type of model specification
- a character variable _NAME_, which indicates the model type, parameter name, or variable name
- a character variable _MATNR_, which indicates the matrix number (COSAN models only)
- a character variable _VAR1_, which is the name or number of the first variable in the specification

- a character variable _VAR2_, which is the name or number of the second variable in the specification
- a numerical variable _ESTIM_ for the final estimate of the parameter location
- a numerical variable _STDERR_ for the standard error estimate of the parameter location

Each observation (record) of the OUTMODEL= data set contains a piece of information regarding the model specification. Depending on the type of the specification indicated by the value of the _TYPE_ variable, the meanings of _NAME_, _VAR1_, and _VAR2_ differ. The following tables summarize the meanings of the _NAME_, _MATNR_ (COSAN models only), _VAR1_, and _VAR2_ variables for each value of the _TYPE_ variable, given the type of the model.

COSAN Models

TYPE=	Description	_NAME_	_MATNR_	_VAR1_	_VAR2_
MDLTYPE	Model type	COSAN			
VAR	Variable	Variable name	Matrix number	Column location	
MATRIX	Matrix	Matrix name	Matrix number	Number of rows	Number of columns
MODEL	Model formula	COV or MEAN	Matrix number	Term number	Location in term
ESTIM	Parameters	Parameter name	Matrix number	Row number	Column number

The value of the _NAME_ variable is COSAN for the _TYPE_=MDLTYPE observation.

The _TYPE_=VAR observations store the information about the column variables in matrices. The _NAME_ variable stores the variable names. The value of _VAR1_ indicates the column location of the variable in the matrix with the matrix number stored in _MATNR_.

The _TYPE_=MATRIX observations store the information about the model matrices. The _NAME_ variable stores the matrix names. The value of _MATNR_ indicates the corresponding matrix number. The values of_VAR1_ and _VAR2_ indicates the numbers of rows and columns, respectively, of the matrix.

The _TYPE_=MODEL observations store the covariance and mean structure formulas. The _NAME_ variable indicates whether the mean (MEAN) or covariance (COV) structure information is stored. The value of _MATNR_ indicates the matrix number in the mean or covariance structure formula. The _VAR1_ variable indicates the term number, and the _VAR2_ variable indicates the location of the matrix in the term.

The _TYPE_=ESTIM observations store the information about the parameters and their estimates. The _NAME_ variable stores the parameter names. The value of _MATNR_ indicates the matrix number. The values of _VAR1_ and _VAR2_ indicate the associated row and column numbers, respectively, of the parameter.

FACTOR Models

TYPE=	Description	_NAME_	_VAR1_	_VAR2_
MDLTYPE	Model type	Model type		
FACTVAR	Variable	Variable name	Variable number	Variable type
LOADING	Factor loading	Parameter name	Manifest variable	Factor variable
COV	Covariance	Parameter name	First variable	Second variable
PVAR	(Partial) variance	Parameter name	Variable	
MEAN	Mean or intercept	Parameter name	Variable	
ADDCOV	Added covariance	Parameter name	First variable	Second variable
ADDPVAR	Added (partial) variance	Parameter name	Variable	
ADDMEAN	Added mean or intercept	Parameter name	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.

The _TYPE_=FACTVAR observations store the information about the variables in the model. The _NAME_ variable stores the variable names. The value of _VAR1_ indicates the variable number. The value of _VAR2_ indicates the type of the variable: either DEPV for dependent observed variables or INDF for latent factors.

Other observations specify the parameters and their estimates in the model. The _NAME_ values for these observations are the parameter names. Observation with _TYPE_=LOADING, _TYPE_=COV, or _TYPE_=ADDCOV are for parameters that are associated with two variables. The _VAR1_ and _VAR2_ values of these two types of observations indicate the variables involved.

Observations with _TYPE_=PVAR, _TYPE_=MEAN, _TYPE_=ADDPVAR, or _TYPE_=ADDMEAN are for parameters that are associated with a single variable. The value of _VAR1_ indicates the variable involved.

LINEQS Models

TYPE=	Description	_NAME_	_VAR1_	_VAR2_
MDLTYPE	Model type	LINEQS		
EQSVAR	Variable	Variable name	Variable number	Variable type
EQUATION	Path coefficient	Parameter	Outcome variable	Predictor variable
COV	Covariance	Parameter	First variable	Second variable
VARIANCE	Variance	Parameter	Variable	
MEAN	Mean	Parameter	Variable	
ADDCOV	Added covariance	Parameter	First variable	Second variable
ADDVARIA	Added variance	Parameter	Variable	
ADDINTE	Added intercept	Parameter	Variable	
ADDMEAN	Added mean	Parameter	Variable	

The value of the _NAME_ variable is LINEQS for the _TYPE_=MDLTYPE observation.

The _TYPE_=EQSVAR observations store the information about the variables in the model. The _NAME_ variable stores the variable names. The value of _VAR1_ indicates the variable number. The value of _VAR2_ indicates the type of the variable. There are six types of variables in the LINEQS model:

- DEPV for dependent observed variables
- INDV for independent observed variables
- DEPF for dependent latent factors
- INDF for independent latent factors
- INDD for independent error terms
- INDE for independent disturbance terms

Other observations specify the parameters and their estimates in the model. The _NAME_ values for these observations are the parameter names. Observation with _TYPE_=EQUATION, _TYPE_=COV, or _TYPE_=ADDCOV are for parameters that are associated with two variables. The _VAR1_ and _VAR2_ values of these two types of observations indicate the variables involved.

Observations with _TYPE_=VARIANCE, _TYPE_=MEAN, _TYPE_=ADDVARIA, _TYPE_=ADDINTE, or _TYPE_=ADDMEAN are for parameters associated with a single variable. The value of _VAR1_ indicates the variable involved.

LISMOD Models

TYPE=	Description	_NAME_	_VAR1_	_VAR2_
MDLTYPE	model type	LISMOD		
XVAR	**x**-variable	Variable	Variable number	
YVAR	**y**-variable	Variable	Variable number	
ETAVAR	η-variable	Variable	Variable number	
XIVAR	ξ-variable	Variable	Variable 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
ADDALPHA	Added _ALPHA_ entry	Parameter	Row number	
ADDKAPPA	Added _KAPPA_ entry	Parameter	Row number	
ADDNUX	Added _NUX_ entry	Parameter	Row number	
ADDNUY	Added _NUY_ entry	Parameter	Row number	
ADDPHI	Added _PHI_ entry	Parameter	Row number	Column number
ADDPSI	Added _PSI_ entry	Parameter	Row number	Column number
ADTHETAX	Added _THETAX_ entry	Parameter	Row number	Column number
ADTHETAY	Added _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 in the respective lists in the model. The _NAME_ variable of these observations stores the names of the variables, and the _VAR1_ variable stores the variable numbers in the respective list. The variable numbers in this data set are not arbitrary—that is, they define the variable orders in the rows and columns of the LISMOD model matrices. The _VAR2_ variable of these observations is not used.

All other observations in this data set specify the parameters in the model. The _NAME_ values of these observations are the parameter names. The corresponding _VAR1_ and _VAR2_ values of these observations indicate the row and column locations of the parameters in the LISMOD model matrices that are specified in the _TYPE_ variable. For example, when the value of _TYPE_ is ADDPHI or PHI, the parameter specified is located in the _PHI_ matrix, with its row and column numbers indicated by the _VAR1_ and _VAR2_ values, respectively. Some observations for specifying parameters do not have values in the _VAR2_ variable. This means that the associated LISMOD matrices are vectors so that the column numbers are always 1 for these observations.

MSTRUCT Models

TYPE=	Description	_NAME_	_VAR1_	_VAR2_
MDLTYPE	Model type	MSTRUCT		
VAR	Variable	Variable	Variable number	
COVMAT	Covariance	Parameter	Row number	Column number
MEANVEC	Mean	Parameter	Row number	
ADCOVMAT	Added covariance	Parameter	Row number	Column number
AMEANVEC	Added 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, and the _VAR1_ variable stores the variable numbers in the variable list. The variable numbers in this data set are not arbitrary—that is, they define the variable orders in the rows and columns of the mean and covariance matrices. The _VAR2_ variable of these observations is not used.

All other observations in this data set specify the parameters in the model. The _NAME_ values of these observations are the parameter names. The corresponding _VAR1_ and _VAR2_ values of these observations indicate the row and column locations of the parameters in the mean or covariance matrix, as specified in the _TYPE_ model. For example, when _TYPE_=COVMAT, the parameter specified is located in the covariance matrix, with its row and column numbers indicated by the _VAR1_ and _VAR2_ values, respectively. For observations with _TYPE_=MEANVEC, the _VAR2_ variable is not used because the column numbers are always 1 for parameters in the mean vector.

PATH Models

TYPE=	Description	_NAME_	_VAR1_	_VAR2_
MDLTYPE	Model type	PATH		
PATHVAR	Variable	Variable name	Variable number	Variable type
LEFT	Path coefficient	Parameter	Outcome variable	Predictor variable
RIGHT	Path coefficient	Parameter	Predictor variable	Outcome variable
PCOV	(Partial) covariance	Parameter	First variable	Second variable
PCOVPATH	(Partial) covariance path	Parameter	First variable	Second variable
PVAR	(Partial) variance	Parameter	Variable	
PVARPATH	(Partial) variance path	Parameter	Variable	Variable
MEAN	Mean or intercept	Parameter	Variable	
ONEPATH	Mean or intercept path	Parameter	_ONE_	Variable
ADDPCOV	Added (partial) covariance	Parameter	First variable	Second variable
ADDPVAR	Added (partial) variance	Parameter	Variable	
ADDMEAN	Added mean	Parameter	Variable	

The value of the _NAME_ variable is PATH for the _TYPE_=MDLTYPE observation.

The _TYPE_=PATHVAR observations store the information about the variables in the model. The _NAME_ variable stores the variable names. The value of _VAR1_ indicates the variable number. The value of _VAR2_ indicates the type of the variable. There are four types of variables in the PATH model:

- DEPV for dependent observed variables
- INDV for independent observed variables
- DEPF for dependent latent factors
- INDF for independent latent factors

Other observations specify the parameters in the model. The _NAME_ values for these observations are the parameter names. Observation with _TYPE_=LEFT, _TYPE_=RIGHT, _TYPE_=PCOV, or _TYPE_=ADDPCOV are for parameters that are associated with two variables. The _VAR1_ and _VAR2_ values of these two types of observations indicate the variables involved.

Observations with _TYPE_=PVAR, _TYPE_=MEAN, _TYPE_=ADDPVAR, or _TYPE_=ADDMEAN are for parameters that are associated with a single variable. The value of _VAR1_ indicates the variable involved.

RAM Models

TYPE=	Description	_NAME_	_VAR1_	_VAR2_
MDLTYPE	Model type	RAM		
RAMVAR	Variable name	Variable	Variable number	Variable type
_A_	_A_ entry	Parameter	Row number	Column number
_P_	_P_ entry	Parameter	Row number	Column number
_W_	_W_ entry	Parameter	Row number	Column number
ADD_P_	Added _P_ entry	Parameter	Row number	Column number
ADD_W_	Added _W_ entry	Parameter	Row number	Column number

The value of the _NAME_ variable is RAM for the _TYPE_=MDLTYPE observation.

For the _TYPE_=RAMVAR observations, the _NAME_ variable stores the variable names, the _VAR1_ variable stores the variable number, and the _VAR2_ variable stores the variable type. There are four types of variables in the PATH model:

- DEPV for dependent observed variables
- INDV for independent observed variables
- DEPF for dependent latent factors
- INDF for independent latent factors

Other observations specify the parameters in the model. The _NAME_ variable stores the parameter name. The _TYPE_ variable indicates the associated matrix with the row number indicated in _VAR1_ and column number indicated in _VAR2_.

Reading an OUTMODEL= Data Set As an INMODEL= Data Set in Subsequent Analyses

When the OUTMODEL= data set is treated as an INMODEL= data set in subsequent analyses, you need to pay attention to observations with _TYPE_ values prefixed by “ADD”, “AD”, or “A” (for example, ADDCOV, ADTHETAY, or AMEANVEC). These observations represent default parameter locations that are generated by PROC CALIS in a previous run. Because the context of the new analyses might be different, these observations for added parameter locations might no longer be suitable in the new runs. Hence, these observations are *not* read as input model information. Fortunately, after reading the INMODEL= specification in the new analyses, CALIS analyzes the new model specification again. It then adds an appropriate set of parameters in the new context when necessary. If you are certain that the added parameter locations in the INMODEL= data set are applicable, you can force the input of these observations by using the READADDPARM option in the PROC CALIS statement. However, you must be very careful about using the READADDPARM option. The added parameters from the INMODEL= data set might have the same parameter names as those for the generated parameters in the new run. This might lead to unnecessary constraints in the model.

OUTSTAT= 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 1047)

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=DWLS	
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 or FIML	
MAXASRES	ML asymptotically standardized residuals
MAXNRES	ML normalized residuals
MAXPRED	ML predicted moments
MAXRES	ML raw residuals
MAXSRES	ML variance standardized residuals
METHOD=ULS	
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
FCOV	Factor correlations or covariances
LOADINGS	Unrotated factor loadings
RFCOV	Rotated factor correlations or covariances
RLOADING	Rotated factor loadings
ROTMAT	Rotation matrix
STDFCOV	Standardized factor correlations
STDLOAD	Standardized factor loadings
STDRFCOV	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 $\mathbf{W}$ 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 $\mathbf{W}$ 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 that contains the values of the fit indices of your analysis. If you use two estimation methods such as LSML or LSWLS, the fit indices are for the second analysis. An OUTFIT=data set contains the following variables:

- a character variable _TYPE_ for the types of fit indices
- a 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: stand-alone fit indices

Parsimony: fit indices that take model parsimony into account

Incremental: fit indices that are based on comparison with a baseline model

Possible Values of _INDEX_ When _TYPE_=ModelInfo

Value of _INDEX_	Description
N Observations	Number of observations used in the analysis
N Complete Observations	Number of complete observations (METHOD=FIML)
N Incomplete Observations	Number of incomplete observations (METHOD=FIML)
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
Saturated Model Estimation	Estimation status of the saturated model (METHOD=FIML)
Saturated Model Function Value	Saturated model function value (METHOD=FIML)
Saturated Model -2 Log-Likelihood	Saturated model -2 log-likelihood function value (METHOD=FIML)
Baseline Model Estimation	Estimation status of the baseline model (METHOD=FIML)
Baseline Model Function Value	Baseline model function value
Baseline Model -2 Log-Likelihood	Baseline model -2 log-likelihood function value (METHOD=FIML)
Baseline Model Chi-Square	Baseline model chi-square value
Baseline Model Chi-Square DF	Baseline model chi-square degrees of freedom
Baseline Model DF	Baseline model degrees of freedom (METHOD=ULS or METHOD=DWLS)
Pr > Baseline Model Chi-Square	p value of the baseline model chi-square

Possible Values of _INDEX_ When _TYPE_=Absolute

Value of _INDEX_	Description
Fit Function	Fit function value
-2 Log-Likelihood	−2 log-likelihood function value for the model (METHOD=FIML)
Chi-Square	Model chi-square value
Chi-Square DF	Degrees of freedom for the model chi-square test
Model DF	Degrees of freedom for model (METHOD=ULS or METHOD=DWLS)
Pr > Chi-Square	Probability of obtaining a larger chi-square than the observed value
Percent Contribution to Chi-Square	Percentage contribution to the chi-square value
Percent Contribution to Likelihood	Percentage contribution to the −2 log-likelihood function value (METHOD=FIML)
Elliptic Corrected Chi-Square	Elliptic-corrected chi-square value
Pr > Elliptic Corr. Chi-Square	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	N value that makes 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. (1989) modification of the GFI
RMSEA Estimate	Steiger and Lind (1980) root mean square error approximation
RMSEA Lower **r**% Confidence Limit	Lower **r**%[1] confidence limit for RMSEA
RMSEA Upper **r**% Confidence Limit	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	Lower **r**%[2] confidence limit for ECVI
ECVI Upper **r**% Confidence Limit	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 Marsh (1988) 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 (1985) comparative fit index
Bentler-Bonett NFI	Bentler and Bonett (1980) normed fit index
Bentler-Bonett Non-normed Index	Bentler and Bonett (1980) nonnormed fit index
Bollen Normed Index Rho1	Bollen normed ρ_1
Bollen Non-normed Index Delta2	Bollen nonnormed δ_2
James et al. Parsimonious NFI	James, Mulaik, and Brett (1982) parsimonious normed fit index

The COSAN Model

The original COSAN (covariance structure analysis) model is proposed by McDonald (1978, 1980) for analyzing general covariance structure models. PROC CALIS enables you to analyze a generalized form of the original COSAN model. The generalized COSAN model extends the original COSAN model with the inclusion of addition terms in the covariance structure formula and the associated mean structure formula.

The covariance structure formula of the generalized COSAN model is

$$\mathbf{\Sigma} = \mathbf{F}_1\mathbf{P}_1\mathbf{F}_1' + \cdots + \mathbf{F}_m\mathbf{P}_m\mathbf{F}_m'$$

and the corresponding mean structure formula of the generalized COSAN model is

$$\boldsymbol{\mu} = \mathbf{F}_1\mathbf{v}_1 + \cdots + \mathbf{F}_m\mathbf{v}_m$$

where $\mathbf{\Sigma}$ is a symmetric correlation or covariance matrix for the observed variables, $\boldsymbol{\mu}$ is a vector for the observed variable means, each $\mathbf{P}_k$ is a symmetric matrix, each $\mathbf{v}_k$ is a mean vector, and each $\mathbf{F}_k$ ($k = 1, \ldots, m$,) is the product of $n(k)$ matrices $\mathbf{F}_{k_1}, \ldots, \mathbf{F}_{k_{n(k)}}$; that is,

$$\mathbf{F}_k = \mathbf{F}_{k_1} \cdots \mathbf{F}_{k_{n(k)}}, \quad k = 1, \ldots, m$$

The matrices $\mathbf{F}_{k_j}$ and $\mathbf{P}_k$ in the model can be one of the forms

$$\mathbf{F}_{k_j} = \begin{cases} \mathbf{G}_{k_j} \\ \mathbf{G}_{k_j}^{-1} \\ (\mathbf{I} - \mathbf{G}_{k_j})^{-1} \end{cases} \quad j = 1, \ldots, n(k) \qquad \text{and} \qquad \mathbf{P}_k = \begin{cases} \mathbf{Q}_k \\ \mathbf{Q}_k^{-1} \end{cases}$$

where $\mathbf{G}_{k_j}$ and $\mathbf{Q}_k$ are basic model matrices that are not expressed as functions of other matrices.

The COSAN model matrices and vectors are $\mathbf{G}_{k_j}$, $\mathbf{Q}_k$, and $\mathbf{v}_k$ (when the mean structures are analyzed). The elements of these model matrices and vectors are either parameters (free or constrained) or fixed values. Matrix $\mathbf{P}_k$ is referred to as the central covariance matrix for the kth term in the covariance structure formula.

Essentially, the COSAN modeling language enables you to define the covariance and mean structure formulas of the generalized COSAN model, the basic COSAN model matrices $\mathbf{G}_{k_j}$, $\mathbf{Q}_k$, and $\mathbf{v}_k$, and the parameters and fixed values in the model matrices.

You can also specify a generalized COSAN model without using an explicit central covariance matrix in any term. For example, you can define the kth term in the covariance structure formula as

$$\mathbf{F}_k\mathbf{F}_k' = \mathbf{F}_{k_1} \ldots \mathbf{F}_{k_{n-1}}\mathbf{F}_{k_n}\mathbf{F}_{k_n}'\mathbf{F}_{k_{n-1}}' \ldots \mathbf{F}_{k_1}'$$

The corresponding term for the mean structure becomes

$$\mathbf{F}_{k_1} \ldots \mathbf{F}_{k_{n-1}}\mathbf{v}_m$$

In the covariance structure formula, $\mathbf{F}_{k_n}\mathbf{F}_{k_n}'$ serves as an implicit central covariance matrix in this term of the covariance structure formula. Because of this, $\mathbf{F}_{k_n}$ does not appear in the corresponding mean structure formula.

To take advantage of the modeling flexibility of the COSAN model specifications, you are required to provide the correct covariance and mean structure formulas for the analysis problem. If you are not familiar with the mathematical formulations of structural equation models, you can consider using simpler modeling languages such as PATH or LINEQS.

An Example: Specifying a Second-Order Factor Model

This example illustrates how to specify the covariance structures in the COSAN statement. Consider a second-order factor analysis model with the following formula for the covariance structures of observed variables v1-v9

$$\mathbf{\Sigma} = \mathbf{F}_1(\mathbf{F}_2\mathbf{P}_2\mathbf{F}_2' + \mathbf{U}_2)\mathbf{F}_1' + \mathbf{U}_1$$

where $\mathbf{F}_1$ is a 9×3 first-order factor matrix, $\mathbf{F}_2$ is a 3×2 second-order factor matrix, $\mathbf{P}_2$ is a 2×2 covariance matrix for the second-order factors, $\mathbf{U}_2$ is a 3×3 diagonal matrix for the unique variances of the first-order factors, and $\mathbf{U}_1$ is a 9×9 diagonal matrix for the unique variances of the observed variables.

To fit this covariance structure model, you first rewrite the covariance structure formula in the form of the generalized COSAN model as

$$\mathbf{\Sigma} = \mathbf{F}_1\mathbf{F}_2\mathbf{P}_2\mathbf{F}_2'\mathbf{F}_1' + \mathbf{F}_1\mathbf{U}_2\mathbf{F}_1' + \mathbf{U}_1$$

You can specify the list of observed variables and the three terms for the covariance structure formula in the following COSAN statement:

```
cosan var= v1-v9,
      F1(3) * F2(2) * P2(2,SYM) + F1(3) * U2(3,DIA) + U1(9,DIA);
```

The VAR= option specifies the nine observed variables in the model. Next, the three terms of the covariance structure formula are specified. Because each term in the covariance structure formula is a symmetric product, you only need to specify each term up to the central covariance matrix. For example, although the first term in the covariance structure formula is $\mathbf{F}_1\mathbf{F}_2\mathbf{P}_2\mathbf{F}_2'\mathbf{F}_1'$, you only need to specify `F1(3) * F2(2) * P2(2,SYM)`. PROC CALIS generates the redundant information for the term. Similarly, you specify the other two terms of the covariance structure formula.

In each matrix specification of the COSAN statement, you can specify the following three matrix properties as the arguments in the trailing parentheses: the number of columns, the matrix type, and the transformation of the matrix. For example, `F1(3)` means that the number of columns of `F1` is 3 (while the number of rows is 9 because this number has to match the number of observed variables specified in the VAR= option), **F2(2)** means that the number of columns of **F2** is 2 (while the number of rows is 3 because the number has to match the number of columns of the preceding matrix, **F1**). You can specify the type of the matrix in the second argument. For example, **P2(2,SYM)** means that **P2** is a symmetric (SYM) matrix and **U2(2,DIA)** means that **U2** is a diagonal (DIA) matrix. You can also specify the transformation of the matrix in the third argument. Because there is no transformation needed in the current second-order factor model, this argument is omitted in the specification. See the COSAN statement for details about the matrix types and transformation that are supported by the COSAN modeling language.

Suppose now you also want to analyze the mean structures of the second-order factor model. The corresponding mean structure formula is

$$\boldsymbol{\mu} = \mathbf{F}_1\mathbf{F}_2\mathbf{v} + \mathbf{u}$$

where $\mathbf{v}$ is a 2×1 mean vector for the second-order factors and $\mathbf{u}$ is a 6×1 vector for the intercepts of the observed variables. To analyze the mean and covariance structures simultaneously, you can use the following COSAN statement:

```
cosan var= v1-v9,
      F1(3) * F2(2) * P2(2,SYM) [mean = v] + F1(3) * U2(3,DIA)
      + U1(9,DIA) [mean = u];
```

In addition to the covariance structure specified, you now add the trailing MEAN= options in the first and the third terms. PROC CALIS then generates the mean structure formula by the following steps:

- Remove the last matrix (that is, the central covariance matrix) in each term of the covariance structure formula.
- Append to each term the vector that is specified in the MEAN= option of the term, or if no MEAN= option is specified in a term, that term becomes a zero vector in the mean structure formula.

Following these steps, the mean structure formula generated for the second-order factor model is

$$\boldsymbol{\mu} = \mathbf{F}_1\mathbf{F}_2\mathbf{v} + 0 + \mathbf{u}$$

which is what you expect for the mean structures of the second-order factor model. To complete the COSAN model specification, you can use MATRIX statements to specify the parameters and fixed values in the COSAN model matrices. See Example 26.28 for a complete example.

Special Cases of the Generalized COSAN Model

It is illustrative to see how you can view different types of models as a special case of the generalized COSAN model. This section describes two such special cases.

The Original COSAN Model

The original COSAN (covariance structure analysis) model (McDonald 1978, 1980) specifies the following covariance structures:

$$\boldsymbol{\Sigma} = \mathbf{F}_1 \cdots \mathbf{F}_n\mathbf{P}\mathbf{F}_n' \cdots \mathbf{F}_1'$$

This is the generalized COSAN with only one term for the covariance structure model formula. Hence, using the COSAN statement to specify the original COSAN model is straightforward.

Reticular Action Model

The RAM (McArdle 1980; McArdle and McDonald 1984) model fits the covariance structures

$$\boldsymbol{\Sigma}_a = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{P}(\mathbf{I} - \mathbf{A})^{-1\prime}$$

where $\boldsymbol{\Sigma}_a$ is the symmetric covariance for all latent and observed variables in the RAM model, **A** is a square matrix for path coefficients, **I** is an identity matrix with the same dimensions as **A**, and **P** is a symmetric covariance matrix. For details about the RAM model, see the section "The RAM Model" on page 1229.

Correspondingly, the RAM model fits the mean structure formula

$$\boldsymbol{\mu}_a = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{w}$$

where $\boldsymbol{\mu}_a$ is the mean vector for all latent and observed variables in the RAM model and **w** is a vector for mean or intercepts of the variables.

To extract the covariance and mean structures for the observed variables, a selection matrix **G** is used. The selection matrix **G** contains zeros and ones as its elements. Each row of **G** has exactly one nonzero element at the position that corresponds to the location of a manifest row variable in $\boldsymbol{\Sigma}_a$ or $\boldsymbol{\mu}_a$. The covariance structure formula for the observed variables in the RAM model becomes

$$\boldsymbol{\Sigma} = \mathbf{G}(\mathbf{I} - \mathbf{A})^{-1}\mathbf{P}(\mathbf{I} - \mathbf{A})^{-1\prime}\mathbf{G}^{\prime}$$

The mean structure formula for the observed variables in the RAM model becomes

$$\boldsymbol{\mu} = \mathbf{G}(\mathbf{I} - \mathbf{A})^{-1}\mathbf{w}$$

These formulas suggest that the RAM model is special case of the generalized COSAN model with one term. For example, suppose that there are 10 observed variables (var1–var10) and 3 latent variables in a RAM model. The following COSAN statement represents the RAM model:

```
cosan var= v1-v10,
      G(13,GEN) * A(13,GEN,IMI) * P(13,SYM) [Mean = w];
```

In the COSAN statement, you define the 10 variables in the VAR= option. Next, you provide the formulas for the mean and covariance structures. **G** is 10 × 13 general matrix (GEN), **A** is a 13 × 13 general matrix with the IMI transformation (that is, $(\mathbf{I} - \mathbf{A})^{-1}$), **P** is a 13 × 13 symmetric matrix (SYM), and **w** is a 13 × 1 vector. With these COSAN statement specifications, your mean and covariance structure formulas represent exactly those of the RAM model. To complete the entire model specification, your next step is to use the MATRIX statements to specify the parameters and fixed values in the model matrices **G**, **A**, **P**, and **w**.

Similarly, it is possible to use the COSAN modeling language to represent any other model types such as models defined by the FACTOR, LINEQS, LISMOD, MSTRUCT, PATH, and RAM statements. But this is not an automatic recommendation of using the COSAN modeling languages in all situations. When an analysis can be specified by either the COSAN or a more specific modeling language (for example, PATH), you should consider using the specific modeling language because the specific modeling language can exploit specific model features so that it does the following:

- enables more supplemental analysis (effect analysis, standardized solutions, and so on), which COSAN has no general way to display
- supports better initial estimation methods (the COSAN model can only set initial estimates to certain default or random values)
- leads to more efficient computations due to the availability of more specific formulas and algorithms

Certainly, the COSAN modeling language is still very useful when you fit some nonstandard model structures that cannot be handled otherwise by the more specific modeling languages.

Naming Variables in the COSAN Model

Although you can define the list of observed (manifest) variables in the VAR= option of the COSAN statement, the COSAN modeling language does not support a direct specification of the latent or error variables in the model. In the COSAN statement, you can define the model matrices and how they multiply together to form the covariance and mean structures. However, except for the row variables of the first matrix in each term, you do not need to *identify* the row and column variables in all other matrices. However, you can use the VARNAMES statement to *label* the column variables of the matrices. The names in the VARNAMES statement follow the general naming rules required by the general SAS system. They should not contain special characters and cannot be longer than 32 characters. Also, they do not need to use certain prefixes like what the LINEQS modeling language requires. It is important to realize that the VARNAME statement only *labels*, but does not *identify*, the column variables (and the row variables, by propagation). This means that while keeping all other things equal, changing the names in the VARNAMES statements does not change the mathematical model or the estimation of the model. For example, you can label all columns of a COSAN matrix with the same name but it does not mean that these columns refer to the same variable in the model. See the section "Naming Variables and Parameters" on page 1238 for the general rules about naming variables and parameters.

Default Parameters in the COSAN Model

The default parameters of the COSAN model matrices depend on the types of the matrices. Each element of the IDE or ZID matrix (identity matrix with or without an additional zero matrix) is either a fixed one or a fixed zero. You cannot override the default parameter values of these fixed matrices. For COSAN model matrices with types other than IDE or ZID, all elements are fixed zeros by default. You can override these default zeros by specifying them explicitly in the MATRIX statements.

The FACTOR Model

The FACTOR modeling language is used for specifying exploratory and confirmatory factor analysis models. You can use other general modeling languages such as LINEQS, LISMOD, PATH, and RAM to specify a factor model. But the FACTOR modeling language is more convenient for specifying factor models and is more specialized in displaying factor-analytic results. For convenience, models specified by the FACTOR modeling language are called FACTOR models.

Types of Variables in the FACTOR Model

Each variable in the FACTOR model is either manifest or latent. Manifest variables are those variables that are measured in the research. They must be present in the input data set. Latent variables are not directly observed. Each latent variable in the FACTOR model can be either a factor or an error term.

Factors are unmeasured hypothetical constructs for explaining the covariances among manifest variables, while errors are the unique parts of the manifest variables that are not explained by the (common) factors.

In the FACTOR model, all manifest variables are endogenous, which means that they are predicted from the latent variables. In contrast, all latent variables in the FACTOR model are exogenous, which means that they serve as predictors only.

Naming Variables in the FACTOR Model

Manifest variables in the FACTOR model are referenced in the input data set. In the FACTOR model specification, you use their names as they appear in the input data set. Manifest variable names must not be longer than 32 characters. There are no further restrictions on these names beyond those required by the SAS System.

Error variables in the FACTOR model are not named explicitly, although they are assumed in the model. You can name latent factors only in confirmatory FACTOR models. Factor names must not be longer than 32 characters and must be distinguishable from the manifest variable names in the same analysis. You do not need to name factors in exploratory FACTOR models, however. Latent factors named Factor1, Factor2, and so on are generated automatically in exploratory FACTOR models.

Model Matrices in the FACTOR Model

Suppose in the FACTOR model that there are p manifest variables and n factors. The FACTOR model matrices are described in the following subsections.

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

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

Matrix $\mathbf{P}$ $(n \times n)$: Factor Covariance Matrix

The $\mathbf{P}$ matrix is a symmetric matrix for the variances of and covariances among the n factors.

Matrix $\mathbf{U}$ $(p \times p)$: Error Covariance Matrix

The $\mathbf{U}$ matrix represents a $p \times p$ diagonal matrix for the error variances for the manifest variables. Elements in this matrix are the parts of variances of the manifest variables that are not explained by the common factors. Note that all off-diagonal elements of $\mathbf{U}$ are fixed zeros in the FACTOR model.

Vector $\mathbf{a}$ $(p \times 1)$ *: Intercepts*

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

Vector $\mathbf{v}$ $(n \times 1)$ *: Factor Means*

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

Matrix Representation of the FACTOR Model

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

$$\mathbf{y} = \mathbf{a} + \mathbf{F}\boldsymbol{\xi} + \mathbf{e}$$

With the model matrix definitions in the previous section, the covariance matrix $\boldsymbol{\Sigma}$ $(p \times p)$ of manifest variables is structured as

$$\boldsymbol{\Sigma} = \mathbf{F}\mathbf{P}\mathbf{F}' + \mathbf{U}$$

The mean vector $\boldsymbol{\mu}$ $(p \times p)$ of manifest variables is structured as

$$\boldsymbol{\mu} = \mathbf{a} + \mathbf{F}\mathbf{v}$$

Exploratory Factor Analysis Models

Traditionally, exploratory factor analysis is applied when the relationships of manifest variables with factors have not been well-established in research. All manifest variables are allowed to have nonzero loadings on the factors in the model. First, factors are extracted and an initial solution is obtained. Then, for ease of interpretation a final factor solution is usually derived by rotating the factor space. Factor-variable relationships are determined by interpreting the final factor solution. This is different from the confirmatory factor analysis in which the factor-variable relationships are prescribed and to be confirmed.

So far, confirmatory and exploratory models are not distinguished in deriving the covariance and mean structures. These two types of models are now distinguished in terms of the required structures or restrictions in model matrices.

In PROC CALIS, the initial exploratory factor solution is obtained from a specific confirmatory factor model with restricted model matrices, which are described as follows:

- The factor loading matrix **F** has $p \times (n-1)/2$ 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.
- Except for METHOD=FIML or METHOD=LSFIML, the mean structures are not modeled. That is, the intercept vector **a** or the factor mean vector **v** are not parameterized in the model.

- With METHOD=FIML or METHOD=LSFIML, the mean structures are modeled. The intercept vector **a** contains p free parameters, and the factor mean vector **v** is a zero vector.

The intercept vector **a** is parameterized only in the FIML method because the first-order moments (that is, the variable means) of the data have to be analyzed with the FIML treatment of the incomplete observations. Other estimation methods would simply omit the incomplete observations, and hence the first-order moments are not analyzed.

With the exploratory factor specification, you do not need to specify the patterns of the model matrices. PROC CALIS automatically sets up the correct patterns for the model matrices. For example, for an analysis with nine variables and three factors, the relevant model matrices of an exploratory FACTOR model have the following patterns, where * denotes free parameters in the model matrices:

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

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

and

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

If METHOD=FIML or METHOD=LSFIML, the elements of the intercept vector **a** are all free parameters, as shown in the following:

$$\mathbf{a} = \begin{pmatrix} * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \end{pmatrix}$$

The factor mean vector **v** is a fixed zero vector.

If an initial factor solution is rotated afterward, some of these matrix patterns are changed. In general, rotating a factor solution eliminates the fixed zero pattern in the upper triangle of the factor loading matrix **F**. If you apply an orthogonal rotation, the factor covariance matrix **P** does not change. It is an identity matrix before and after rotation. However, if you apply an oblique rotation, in general the rotated factor covariance matrix **P** is not an identity matrix and the off-diagonal elements are not zeros.

The error covariance matrix **U** remains unchanged after rotation. That is, it would still be a diagonal matrix. For the FIML estimation, the rotation does not affect the estimation of the intercept vector **a** and the fixed factor mean vector **v**.

Confirmatory Factor Analysis Models

In confirmatory FACTOR models, there are no imposed patterns on the **F**, **P**, **a**, and **v** model matrices. All elements in these model matrices can be specified. However, for model identification, you might need to specify some factor loadings or factor variances as constants.

The only model restriction in confirmatory FACTOR models is placed on **U**, which must be a diagonal matrix, as in exploratory FACTOR models too.

For example, for a confirmatory factor analysis with nine variables and three factors, you might specify the following patterns for the model matrices, where * denotes free parameters in the model matrices:

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

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

and

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

In this confirmatory factor model, mean structures are not modeled. In addition, there are some distinctive features that underscore the differences between confirmatory and exploratory models:

- Factor loading matrix **F** contains mostly zero elements and few nonzero free parameters, a pattern which is seen in most confirmatory factor models. In contrast, in exploratory factor models most elements in the **F** matrix are nonzero parameters.
- Factor loading matrix **F** contains fixed values of ones. These fixed values are used for model identification purposes (that is, identifying the scales of the latent variables). In general, you always have to make sure that your confirmatory factor models are identified by putting fixed values in appropriate parameter locations in the model matrices. However, this is not a concern in exploratory FACTOR models because identification has been ensured by imposing certain patterns on the model matrices.
- The nonzero off-diagonal parameters in the factor covariance matrix **P** indicate that correlated factors are hypothesized in the confirmatory factor model. This cannot be the case with the initial model of exploratory FACTOR models, where the **P** matrix must be an identity matrix before rotation.

Summary of Matrices in the FACTOR Model

Let p be the number of manifest variables and n be the number of factors in the FACTOR model. The names, roles, and dimensions of the FACTOR model matrices are shown in the following table.

Matrix	Name	Description	Dimensions
F	_FACTLOAD_	Factor loading matrix	$p \times n$
P	_FACTFCOV_	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 CALIS, you do not need to specify any other parameters in the model matrices. To obtain desired factor solutions, you can use various options for exploratory factor analysis in the FACTOR statement. These options are the *EFA_options* in the FACTOR statement. Two main types of *EFA_options* are shown as follows:

- options for factor extraction: COMPONENT, HEYWOOD, and N=.
- options for factor rotation: GAMMA=, NORM=, RCONVERGE=, RITER=, ROTATE=, and TAU=.

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

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

See the FACTOR statement on page 1072 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 CALIS can determine appropriate initial estimates for these parameters. See the descriptions of *CFA_spec* in the FACTOR statement on page 1072 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 1149 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 1065 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 1125 for more details about the syntax.

Naming the Factors

For the exploratory FACTOR model, PROC CALIS generates the names for the factors automatically. For the confirmatory FACTOR model, you can specify the names for the factors. Unlike the LINEQS model, in the confirmatory FACTOR model you do not need to use the ‘F’ or ‘f’ prefix to denote factors in the model. You can use any valid SAS variable names for the factors, especially those names that reflect the nature of the factors. To avoid confusions with other names in the model, some general rules are recommended. See the section “Naming Variables and Parameters” on page 1238 for these general rules about naming variables and parameters.

Default Parameters in the FACTOR Model

Default parameters in the FACTOR model are different for exploratory and confirmatory factor models.

For the exploratory FACTOR model, all fixed and free parameters of the model are prescribed. These prescribed parameters include a fixed pattern for the factor loading matrix **F**, a diagonal pattern for the error variance matrix **U**, and an identity matrix for factor covariance matrix **P**. This means that factors are uncorrelated in the estimation. However, if you specify an oblique rotation after the estimation of the factor solution, the factors could become correlated. See the section “Exploratory Factor Analysis Models” on page 1199 for more details about the patterns of the exploratory FACTOR model. Because all these patterns are prescribed, you cannot override any of these parameters for the exploratory FACTOR model.

For the confirmatory FACTOR model, the set of default free parameters of the confirmatory FACTOR model includes the following:

- the error variances of the observed variables; these correspond to the diagonal elements of the uniqueness matrix **U**
- the variances and covariances among the factors; these correspond to all elements of the factor covariance matrix **P**

- the intercepts of the observed variables if the mean structures are modeled; these correspond to all elements of the intercept vector **a**

PROC CALIS names the default free parameters with the _Add prefix, followed by a unique integers for each parameter. You can override the default free parameters by explicitly specifying them as free, constrained, or fixed parameters in the COV, MEAN, or PVAR statement.

In addition to default free parameters, another type of default parameter is the fixed zeros applied to the unspecified parameters in the loading matrix **F** and the factor means in the $\boldsymbol{\nu}$ vector. Certainly, you use the FACTOR and MEAN specifications to override those default zero loadings or factor means and set them to free, constrained, or fixed parameters. Notice that the uniqueness matrix **U** in the confirmatory factor model is a diagonal element. You cannot specify any of its off-diagonal elements—they are always fixed zeros by the model restriction.

The LINEQS Model

The LINEQS modeling language is adapted from the EQS (equations) program by Bentler (1995). The statistical models that LINEQS or EQS analyzes are essentially the same as other general modeling languages such as LISMOD, RAM, and PATH. However, the terminology and approach of the LINEQS or EQS modeling language are different from other languages. They are based on the theoretical model developed by Bentler and Weeks (1980). For convenience, models that are analyzed using the LINEQS modeling language are called LINEQS models. Note that these so-called LINEQS models can also be analyzed by other general modeling languages in PROC CALIS.

In the LINEQS (or the original EQS) model, relationships among variables are represented by a system of equations. For example:

$$Y_1 = a_0 + a_1 X_1 + a_2 X_2 + E_1$$

$$Y_2 = b_0 + b_1 X_1 + b_2 Y_1 + E_2$$

On the left-hand side of each equation, an outcome variable is hypothesized to be a linear function of one or more predictor variables and an error, which are all specified on the right-hand side of the equation. The parameters specified in an equation are the effects (or regression coefficients) of the predictor variables. For example, in the preceding equations, Y_1 and Y_2 are outcome variables; E_1 and E_2 are error variables; a_1, a_2, b_1, and b_2 are effect parameters (or regression coefficients); and a_0 and b_0 are intercept parameters. Variables X_1 and X_2 serve as predictors in the first equation, while variables X_1 and Y_1 serve as predictors in the second equation.

This is almost the same representation as in multiple regression models. However, the LINEQS model entails more. It supports a system of equations that can also include latent variables, measurement errors, and correlated errors.

Types of Variables in the LINEQS Model

The distinction between dependent and independent variables is important in the LINEQS model.

A variable is dependent if it appears on the left-hand side of an equation in the model. A dependent variable might be observed (manifest) or latent. It might or might not appear on the right-hand side of other equations, but it cannot appear on the left-hand sides of two or more equations. Error variables cannot be dependent in the LINEQS model.

A variable in the LINEQS model is independent if it is not dependent. Independent variables can be observed (manifest) or latent. All error variables must be independent in the LINEQS model.

Dependent variables are also referred to as endogenous variables; these names are interchangeable. Similarly, independent variables are interchangeable with exogenous variables.

Whereas an outcome variable in any equation must be a dependent variable, a predictor variable in an equation is not necessarily an independent variable in the entire LINEQS model. For example, Y_1 is a predictor variable in the second equation of the preceding example, but it is a dependent variable in the LINEQS model. In summary, the predictor-outcome nature of a variable is determined within a single equation, while the exogenous-endogenous (independent-dependent) nature of variable is determined within the entire system of equations.

In addition to the dependent-independent variable distinction, variables in the LINEQS model are distinguished according to whether they are observed in the data. Variables that are observed in research are called observed or manifest variables. Hypothetical variables that are not observed in the LINEQS model are latent variables.

Two types of latent variables should be distinguished: one is error variables; the other is non-error variables. An error variable is unique to an equation. It serves as the unsystematic source of effect for the outcome variable in an equation. If the outcome variable in the equation is latent, the corresponding error variable is also called disturbance. In contrast, non-error or systematic latent variables are called factors. Factors are unmeasured hypothetical constructs in your model. They are systematic sources that explain or describe functional relationships in your model.

Both manifest variables and latent factors can be dependent or independent. However, error or disturbance terms must be independent (or exogenous) variables in your model.

Naming Variables in the LINEQS Model

Whether a variable in each equation is an outcome or a predictor variable is prescribed by the modeler. Whether a variable is independent or dependent can be determined by analyzing the entire system of equations in the model. Whether a variable is observed or latent can be determined if it is referenced in your data set. However, whether a latent variable serves as a factor or an error can be determined only if you provide the specific information.

To distinguish latent factors from errors and both from manifest variables, the following rules for naming variables in the LINEQS model are followed:

- Manifest variables are referenced in the input data set. You use their names in the LINEQS model specification directly. There is no additional naming rule for the manifest variables in the LINEQS model beyond those required by the SAS System.
- Latent factor variables must start with letter F or f (for factor).
- Error variables must start with letter E or e (for error), or D or d (for disturbance). Although you might enforce the use of D- (or d-) variables for disturbances, it is not required. For flexibility, disturbance variables can also start with letter E or e in the LINEQS model.
- The names of latent variables, errors, and disturbances (F-, E-, and D-variables) should not coincide with the names of manifest variables.
- You should not use Intercept as a name for any variable. This name is reserved for the intercept specification in LINEQS model equations.

See the section "Naming Variables and Parameters" on page 1238 for the general rules about naming variables and parameters.

Matrix Representation of the LINEQS Model

As a programming language, the LINEQS model uses equations to describes relationships among variables. But as a mathematical model, the LINEQS model is more conveniently described by matrix terms. In this section, the LINEQS matrix model is described.

Suppose in a LINEQS model that there are n_i independent variables and n_d dependent variables. The vector of the independent variables is denoted by $\boldsymbol{\xi}$, in the order of manifest variables, latent factors, and error variables. The vector of dependent variables is denoted by $\boldsymbol{\eta}$, in the order of manifest variables and latent factors. The LINEQS model matrices are defined as follows:

$\boldsymbol{\alpha}\ (n_d \times 1)$:	intercepts of dependent variables
$\boldsymbol{\beta}\ (n_d \times n_d)$:	effects of dependent variables (in columns) on dependent variables (in rows)
$\boldsymbol{\gamma}\ (n_d \times n_i)$:	effects of independent variables (in columns) on dependent variables (in rows)
$\boldsymbol{\Phi}\ (n_i \times n_i)$:	covariance matrix of independent variables
$\boldsymbol{\nu}\ (n_i \times 1)$:	means of independent variables

The model equation of the LINEQS model is

$$\boldsymbol{\eta} = \boldsymbol{\alpha} + \boldsymbol{\beta}\boldsymbol{\eta} + \boldsymbol{\gamma}\boldsymbol{\xi}$$

Assuming that $(\mathbf{I} - \boldsymbol{\beta})$ is invertible, under the model the covariance matrix of all variables $(\boldsymbol{\eta}', \boldsymbol{\xi}')'$ is structured as

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

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

$$\boldsymbol{\mu}_a = \begin{pmatrix} (\mathbf{I}-\boldsymbol{\beta})^{-1}(\boldsymbol{\alpha} + \boldsymbol{\gamma}\boldsymbol{\nu}) \\ \boldsymbol{\nu} \end{pmatrix}$$

As is shown in the structured covariance and mean matrices, the means $\mathbf{G}$ and covariances of independent variables are direct model parameters in $\boldsymbol{\nu}$ and $\boldsymbol{\Phi}$; whereas the means and covariances of dependent variables are functions of various model matrices and hence functions of model parameters.

The covariance and mean structures of all observed variables are obtained by selecting the elements in $\boldsymbol{\Sigma}_a$ and $\boldsymbol{\mu}_a$. Mathematically, define a selection matrix $\mathbf{G}$ of dimensions $n \times (n_d + n_i)$, where n is the number of observed variables in the model. The selection matrix $\mathbf{G}$ contains zeros and ones as its elements. Each row of $\mathbf{G}$ has exactly one nonzero element at the position that corresponds to the location of an observed row variable in $\boldsymbol{\Sigma}_a$ or $\boldsymbol{\mu}_a$. With each row of $\mathbf{G}$ selecting a distinct observed variable, the structured covariance matrix of all observed variables is represented by

$$\boldsymbol{\Sigma} = \mathbf{G}\boldsymbol{\Sigma}_a\mathbf{G}'$$

The structured mean vector of all observed variables is represented by

$$\boldsymbol{\mu} = \mathbf{G}\boldsymbol{\mu}_a$$

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 $\boldsymbol{\beta}$ matrix must be zeros. This prevents the direct regression of dependent variables on themselves. Hence, in the LINEQS statement you cannot specify the same variable on both the left-hand and the right-hand sides of the same equation.

Partitions of the γ Matrix and the Associated Model Restrictions

The columns of the $\boldsymbol{\gamma}$ matrix refer to the variables in $\boldsymbol{\xi}$, in the order of manifest variables, latent factors, and error variables. In the LINEQS model, the following partition of the $\boldsymbol{\gamma}$ matrix is assumed:

$$\boldsymbol{\gamma} = (\boldsymbol{\gamma}_0 \quad \mathbf{E})$$

where $\boldsymbol{\gamma}_0$ is an $n_d \times (n_i - n_d)$ matrix for the effects of independent manifest variables and latent factors on the dependent variables and $\mathbf{E}$ is an $n_d \times n_d$ permutation matrix for the effects of errors on the dependent variables.

The dimension of submatrix $\mathbf{E}$ is $n_d \times n_d$ because in the LINEQS model each dependent variable signifies an equation with an error term. In addition, because $\mathbf{E}$ is a permutation matrix (which is formed by exchanging rows of an identity matrix of the same order), the partition of the $\boldsymbol{\gamma}$ matrix ensures that each dependent variable is associated with a *unique* error term and that the effect of each error term on its associated dependent variable is 1.

As a result of the error term restriction, in the LINEQS statement you must specify a unique error term in each equation. The coefficient associated with the error term can only be a fixed value at one, either explicitly (with 1.0 inserted immediately before the error term) or implicitly (with no coefficient specified).

Partitions of the ν Vector and the Associated Model Restrictions

The $\boldsymbol{\nu}$ vector contains the means of independent variables, in the order of the manifest, latent factor, and error variables. In the LINEQS model, the following partition of the $\boldsymbol{\nu}$ vector is assumed:

$$\boldsymbol{\nu} = \begin{pmatrix} \boldsymbol{\nu}_0 \\ 0 \end{pmatrix}$$

where $\boldsymbol{\nu}_0$ is an $(n_i - n_d) \times 1$ vector for the means of independent manifest variables and latent factors and 0 is a null vector of dimension n_d for the means of errors or disturbances. Again, the dimension of the null vector is n_d because each dependent variable is associated uniquely with an error term. This partition restricts the means of errors or disturbances to zeros.

Hence, when specifying a LINEQS model, you cannot specify the means of errors (or disturbances) as free parameter or fixed values other than zero in the MEAN statement.

Partitions of the Φ matrix

The $\boldsymbol{\Phi}$ matrix is for the covariances of the independent variables, in the order of the manifest, latent factor, and error variables. The following partition of the $\boldsymbol{\Phi}$ matrix is assumed:

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

where $\boldsymbol{\Phi}_{11}$ is an $(n_i - n_d) \times (n_i - n_d)$ covariance matrix for the independent manifest variables and latent factors, $\boldsymbol{\Phi}_{22}$ is an $n_d \times n_d$ covariance matrix for the errors, and $\boldsymbol{\Phi}_{21}$ is an $n_d \times (n_i - n_d)$ covariance matrix for the errors with other independent variables in the LINEQS model. Because $\boldsymbol{\Phi}$ is symmetric, $\boldsymbol{\Phi}_{11}$ and $\boldsymbol{\Phi}_{22}$ are also symmetric.

There are actually no model restrictions placed on the submatrices of the partition. However, in most statistical applications, errors represent unsystematic sources of effects and therefore they are not to be correlated with other systematic sources. This implies that submatrix $\boldsymbol{\Phi}_{21}$ is a null matrix. However, $\boldsymbol{\Phi}_{21}$ being null is not enforced in the LINEQS model specification. If you ever specify a covariance between an error variable and a non-error independent variable in the COV statement, as a workaround trick or otherwise, you should provide your own theoretical justifications.

Summary of Matrices and Submatrices in the LINEQS Model

Let n_d be the number of dependent variables and n_i be the number of independent variables. The names, roles, and dimensions of the LINEQS model matrices and submatrices are summarized in the following table.

Matrix	Name	Description	Dimensions
Model Matrices			
$\boldsymbol{\alpha}$	_EQSALPHA_	Intercepts of dependent variables	$n_d \times 1$
$\boldsymbol{\beta}$	_EQSBETA_	Effects of dependent (column) variables on dependent (row) variables	$n_d \times n_d$
$\boldsymbol{\gamma}$	_EQSGAMMA_	Effects of independent (column) variables on dependent (row) variables	$n_d \times n_i$
$\boldsymbol{\nu}$	_EQSNU_	Means of independent variables	$n_i \times 1$
$\boldsymbol{\Phi}$	_EQSPHI_	Covariance matrix of independent variables	$n_i \times n_i$
Submatrices			
$\boldsymbol{\gamma}_0$	_EQSGAMMA_SUB_	Effects of independent variables, excluding errors, on dependent variables	$n_d \times (n_i - n_d)$
$\boldsymbol{\nu}_0$	_EQSNU_SUB_	Means of independent variables, excluding errors	$(n_i - n_d) \times 1$
$\boldsymbol{\Phi}_{11}$	_EQSPHI11_	Covariance matrix of independent variables, excluding errors	$(n_i - n_d) \times (n_i - n_d)$
$\boldsymbol{\Phi}_{21}$	_EQSPHI21_	Covariances of errors with other independent variables	$n_d \times (n_i - n_d)$
$\boldsymbol{\Phi}_{22}$	_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 $\boldsymbol{\alpha}$ vector and the $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ 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 1090 for the details about the syntax.

Because of the LINEQS model restrictions (see the section "Partitions of Some LINEQS Model Matrices and Their Restrictions" on page 1208), you must also follow these rules when specifying LINEQS model equations:

- A dependent variable can appear only on the left-hand side of an equation once. In other words, you must put all predictor variables for a dependent variable in one equation. This is different from some econometric models where a dependent variable can appear on the left-hand sides of two equations to represent an equilibrium point. However, this limitation can be resolved by reparameterization in some cases. See Example 26.17.
- A dependent variable that appears on the left-hand side of an equation cannot appear on the right-hand side of the same equation. If you measure the same characteristic at different time points and the previous measurement serves as a predictor of the next measurement, you should use different variable names for the measurements so as to comply with this rule.
- An error term must be specified in each equation and must be unique. The same error name cannot appear in two or more equations. When an equation is truly intended to have no error term, it should be represented equivalently in the LINEQS equation by introducing an error term with zero variance (specified in the VARIANCE statement).
- The regression coefficient (effect) that is associated with an error term must be fixed at one (1.0). This is done automatically by omitting any fixed constants or parameters that are associated with the error terms. Inserting a parameter or a fixed value other than 1 immediately before an error term is not allowed.

Mean, Variance, and Covariance Parameter Specification

In addition to the intercept and effect parameters that are specified in equations, the means, variances, and covariances among all independent variables are parameters in the LINEQS model. An exception is that the means of all error variables are restricted to fixed zeros in the LINEQS model. To specify the mean, variance, and covariance parameters, you use the MEAN, VARIANCE, and the COV statements, respectively.

The means, variances, and covariances among dependent variables are not parameters themselves in the model. Rather, they are complex functions of the model parameters. See the section "Matrix Representation of the LINEQS Model" on page 1207 for mathematical details.

Default Parameters in the LINEQS Model

There are two types of default parameters of the LINEQS model, as implemented in PROC CALIS. One is the free parameters; the other is the fixed constants.

The following sets of parameters are free parameters by default:

- the variances of all exogenous (independent) observed or latent variables (*including* error and disturbance variables)

- the covariances among all exogenous (independent) manifest or latent variables (*excluding* error and disturbance variances)
- the means of all exogenous (independent) observed variables if the mean structures are modeled
- the intercepts of all endogenous (dependent) manifest variables if the mean structures are modeled

PROC CALIS names the default free parameters with the _Add prefix and a unique integer suffix. You can override the default free parameters by explicitly specifying them as free, constrained, or fixed parameters in the COV, LINEQS, MEAN, or VARIANCE statement.

Parameters that are not default free parameters in the LINEQS model are fixed constants by default. You can override almost all of the default fixed constants of the LINEQS model by using the COV, LINEQS, MEAN, or VARIANCE statement. You cannot override the following two sets of fixed constants:

- fixed zero parameters for the direct effects (path coefficients) of variables on their own. You cannot have an equation in the LINEQS statement that has the same variable specified on the left-hand and the right-hand sides.
- fixed one effects from the error or disturbance variables. You cannot set the path coefficient (effect) of the error or disturbance term to any value other than 1 in the LINEQS statement.

These two sets of fixed parameters reflect the LINEQS model restrictions so that they cannot be modified. Other than these two sets of default fixed parameters, all other default fixed parameters are zeros. You can override these default zeros by explicitly specifying them as free, constrained, or fixed parameters in the COV, LINEQS, MEAN, or VARIANCE statement.

The LISMOD Model and Submodels

As a statistical model, the LISMOD modeling language is derived from the LISREL model proposed by Jöreskog and others (see Keesling 1972; Wiley 1973; Jöreskog 1973). But as a computer language, the LISMOD modeling language is quite different from the LISREL program. To maintain the consistence of specification syntax within the CALIS procedure, the LISMOD modeling language departs from the original LISREL programming language. In addition, to make the programming a little easier, some terminological changes from LISREL are made in LISMOD.

For brevity, models specified by the LISMOD modeling language are called LISMOD models, although you can also specify these LISMOD models by other general modeling languages that are supported in PROC CALIS.

The following descriptions of LISMOD models are basically the same as those of the original LISREL models. The main modifications are the names for the model matrices.

The LISMOD model is described by three component models. The first one is the structural equation model that describes the relationships among latent constructs or factors. The other two are measurement models that relate latent factors to manifest variables.

Structural Equation Model

The structural equation model for latent factors is

$$\boldsymbol{\eta} = \boldsymbol{\alpha} + \boldsymbol{\beta}\boldsymbol{\eta} + \boldsymbol{\Gamma}\boldsymbol{\xi} + \boldsymbol{\zeta}$$

where:
$\boldsymbol{\eta}$ is a random vector of n_η endogenous latent factors
$\boldsymbol{\xi}$ is a random vector of n_ξ exogenous latent factors
$\boldsymbol{\zeta}$ is a random vector of errors
$\boldsymbol{\alpha}$ is a vector of intercepts
$\boldsymbol{\beta}$ is a matrix of regression coefficients of $\boldsymbol{\eta}$ variables on other $\boldsymbol{\eta}$ variables
$\boldsymbol{\Gamma}$ is a matrix of regression coefficients of $\boldsymbol{\eta}$ on $\boldsymbol{\xi}$

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

The covariance matrix of $\boldsymbol{\zeta}$ is denoted by $\boldsymbol{\Psi}$ and its expected value is a null vector. The covariance matrix of $\boldsymbol{\xi}$ is denoted by $\boldsymbol{\Phi}$ and its expected value is denoted by $\boldsymbol{\kappa}$.

Because variables in the structural equation model are not observed, to analyze the model these latent variables must somehow relate to the manifest variables. The measurement models, which are discussed in the subsequent sections, provide such relations.

Measurement Model for y

$$\mathbf{y} = \boldsymbol{\nu}_y + \boldsymbol{\Lambda}_y\boldsymbol{\eta} + \boldsymbol{\epsilon}$$

where:
$\mathbf{y}$ is a random vector of n_y manifest variables
$\boldsymbol{\epsilon}$ is a random vector of errors for $\mathbf{y}$
$\boldsymbol{\nu}_y$ is a vector of intercepts for $\mathbf{y}$
$\boldsymbol{\Lambda}_y$ is a matrix of regression coefficients of $\mathbf{y}$ on $\boldsymbol{\eta}$

It is assumed that $\boldsymbol{\epsilon}$ is uncorrelated with either $\boldsymbol{\eta}$ or $\boldsymbol{\zeta}$. The covariance matrix of $\boldsymbol{\epsilon}$ is denoted by $\boldsymbol{\Theta}_y$ and its expected value is the null vector.

Measurement Model for x

$$\mathbf{x} = \boldsymbol{\nu}_x + \boldsymbol{\Lambda}_x\boldsymbol{\xi} + \boldsymbol{\delta}$$

where:
$\mathbf{x}$ is a random vector of n_x manifest variables
$\boldsymbol{\delta}$ is a random vector of errors for $\mathbf{x}$
$\boldsymbol{\nu}_x$ is a vector of intercepts for $\mathbf{x}$
$\boldsymbol{\Lambda}_x$ is a matrix of regression coefficients of $\mathbf{x}$ on $\boldsymbol{\xi}$

It is assumed that $\boldsymbol{\delta}$ is uncorrelated with $\boldsymbol{\xi}$, $\boldsymbol{\epsilon}$, or $\boldsymbol{\zeta}$. The covariance matrix of $\boldsymbol{\delta}$ is denoted by $\boldsymbol{\Theta}_x$ and its expected value is a null vector.

Covariance and Mean Structures

Under the structural and measurement equations and the model assumptions, the covariance structures of the manifest variables $(y', x')'$ are expressed as

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Lambda}_y (I - \boldsymbol{\beta})^{-1} (\boldsymbol{\Gamma}\boldsymbol{\Phi}\boldsymbol{\Gamma}' + \boldsymbol{\Psi})(I - \boldsymbol{\beta}')^{-1}\boldsymbol{\Lambda}_y' + \boldsymbol{\Theta}_y & \boldsymbol{\Lambda}_y (I - \boldsymbol{\beta})^{-1}\boldsymbol{\Gamma}\boldsymbol{\Phi}\boldsymbol{\Lambda}_x' \\ \boldsymbol{\Lambda}_x \boldsymbol{\Phi}\boldsymbol{\Gamma}'(I - \boldsymbol{\beta}')^{-1}\boldsymbol{\Lambda}_y' & \boldsymbol{\Lambda}_x \boldsymbol{\Phi}\boldsymbol{\Lambda}_x' + \boldsymbol{\Theta}_x \end{pmatrix}$$

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

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\nu}_y + \boldsymbol{\Lambda}_y (I - \boldsymbol{\beta})^{-1} (\boldsymbol{\alpha} + \boldsymbol{\Gamma}\boldsymbol{\kappa}) \\ \boldsymbol{\nu}_x + \boldsymbol{\Lambda}_x \boldsymbol{\kappa} \end{pmatrix}$$

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
$\boldsymbol{\alpha}$	_ALPHA_	Intercepts for $\boldsymbol{\eta}$	$n_\eta \times 1$	$\boldsymbol{\eta}$ (ETAVAR=)	N/A
$\boldsymbol{\beta}$	_BETA_	Effects of $\boldsymbol{\eta}$ on $\boldsymbol{\eta}$	$n_\eta \times n_\eta$	$\boldsymbol{\eta}$ (ETAVAR=)	$\boldsymbol{\eta}$ (ETAVAR=)
$\boldsymbol{\Gamma}$	_GAMMA_	Effects of $\boldsymbol{\xi}$ on $\boldsymbol{\eta}$	$n_\eta \times n_\xi$	$\boldsymbol{\eta}$ (ETAVAR=)	$\boldsymbol{\xi}$ (XIVAR=)
$\boldsymbol{\Psi}$	_PSI_	Error covariance matrix for $\boldsymbol{\eta}$	$n_\eta \times n_\eta$	$\boldsymbol{\eta}$ (ETAVAR=)	$\boldsymbol{\eta}$ (ETAVAR=)
$\boldsymbol{\Phi}$	_PHI_	Covariance matrix for $\boldsymbol{\xi}$	$n_\xi \times n_\xi$	$\boldsymbol{\xi}$ (XIVAR=)	$\boldsymbol{\xi}$ (XIVAR=)
$\boldsymbol{\kappa}$	_KAPPA_	Mean vector for $\boldsymbol{\xi}$	$n_\xi \times 1$	$\boldsymbol{\xi}$ (XIVAR=)	N/A
$\boldsymbol{\nu}_y$	_NUY_	Intercepts for **y**	$n_y \times 1$	**y** (YVAR=)	N/A
$\boldsymbol{\Lambda}_y$	_LAMBDAY_	Effects of $\boldsymbol{\eta}$ on **y**	$n_y \times n_\eta$	**y** (YVAR=)	$\boldsymbol{\eta}$ (ETAVAR=)
$\boldsymbol{\Theta}_y$	_THETAY_	Error covariance matrix for **y**	$n_y \times n_y$	**y** (YVAR=)	**y** (YVAR=)
$\boldsymbol{\nu}_x$	_NUX_	Intercepts for **x**	$n_x \times 1$	**x** (XVAR=)	N/A
$\boldsymbol{\Lambda}_x$	_LAMBDAX_	Effects of $\boldsymbol{\xi}$ on **x**	$n_x \times n_\xi$	**x** (XVAR=)	$\boldsymbol{\xi}$ (XIVAR=)
$\boldsymbol{\Theta}_x$	_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 1216 for details. In the preceding table, each model matrix is given a name in the column Name, followed by a brief description of the parameters in the matrix, the dimensions, and the row and column variables being referred to. In the second column of the table, the LISMOD matrix names are used in the MATRIX statements when specifying the LISMOD model. In the last two columns of the table, following the row or column variables is the variable list (for example, ETAVAR=, YVAR=, and so on) in parentheses. These lists are used in the LISMOD statement for defining variables.

Specification of the LISMOD Model

The LISMOD specification consists of two tasks. The first task is to define the variables in the model. The second task is to specify the parameters in the LISMOD model matrices.

Specifying Variables

The first task is accomplished in the LISMOD statement. In the LISMOD statement, you define the lists of variables of interest: YVAR=, XVAR=, ETAVAR=, and XIVAR= lists, respectively for the $\mathbf{y}$-variables, $\mathbf{x}$-variables, $\boldsymbol{\eta}$-variables, and the $\boldsymbol{\xi}$-variables. While you provide the names of variables in these lists, you also define implicitly the numbers of four types of variables: n_y, n_x, n_η, and n_ξ. The variables in the YVAR= and XVAR= lists are manifest variables and therefore must be present in the analyzed data set. The variables in the ETAVAR= and XIVAR= lists are latent factors, the names of which are assigned by the researcher to represent their roles in the substantive theory. After these lists are defined, the dimensions of the model matrices are also defined by the number of variables on various lists. In addition, the variable orders in the lists are referred to by the row and column variables of the model matrices.

Unlike the LINEQS model, in the LISMOD model you do not need to use the ‘F’ or ‘f’ prefix to denote factors in the ETAVAR= or XIVAR= list. You can use any valid SAS variable names for the factors, especially those names that reflect the nature of the factors. To avoid confusion with other names in the model, some general rules are recommended. See the section “Naming Variables and Parameters” on page 1238 for these general rules about naming variables and parameters.

Specifying Parameters in Model Matrices

The second task is accomplished by the MATRIX statements. In each MATRIX statement, you specify the model matrix by using the matrix names described in the previous table. Then you specify the parameters (free or fixed) in the locations of the model matrix. You can use as many MATRIX statements as needed for defining your model. But each model matrix can be specified only in one MATRIX statement, and each MATRIX statement is used for specifying only one model matrix.

An Example

In the section “LISMOD Model” on page 1008, the LISMOD modeling language is used to specify the model described in the section “A Structural Equation Example” on page 1002. 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 at 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_ ($\mathbf{\Lambda}_x$) 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 an 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 $\mathbf{\Theta}_y$ matrix) for the **y**-variables are specified. The diagonal elements of the _THETAY_ matrix are specified by parameters theta1, theta2, theta1, and theta2, respectively, for the four **y**-variables Anomie67, Powerless67, Anomie71, and Powerless71. By using the same parameter name theta1, the error variances for Anomie67 and Anomie71 are implicitly constrained. Similarly, the error variances for Powerless67 and Powerless71 are also implicitly constrained. Two more parameter locations are specified. The error covariance between Anomie67 and Anomie71 and the error covariance between Powerless67 and Powerless71 are both represented by the parameter theta5. Again, this is an implicit constraint on the covariances. All other unspecified elements in the _THETAY_ matrix are treated as fixed zeros.

In this example, no parameters are specified for matrices _ALPHA_, _KAPPA_, _NUY_, 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 CALIS analyzes submodels derived logically from the specified lists of variables. For example, if only **y**- and **x**-variable lists are specified, the submodel being analyzed would be a multivariate regression model with manifest variables only. Not all combinations of lists lead to meaningful submodels, however. To determine whether and how a submodel (which is formed by a certain combination of variable lists) can be analyzed, the following three principles in the LISMOD modeling language are applied:

- Submodels with at least one of the YVAR= and XVAR= lists are required.
- Submodels that have an ETAVAR= list but no YVAR= list cannot be analyzed.
- When a submodel has a YVAR= (or an XVAR=) list but without an ETAVAR= (or a XIVAR=) list, it is assumed that the set of **y**-variables (**x**-variables) is equivalent to the $\boldsymbol{\eta}$-variables ($\boldsymbol{\xi}$-variables). Hereafter, this principle is referred to as an equivalence interpretation.

Apparently, the third principle is the same as the situation where the latent factors $\boldsymbol{\eta}$ (or $\boldsymbol{\xi}$) are perfectly measured by the manifest variables **y** (or **x**). That is, in such a perfect measurement model, $\boldsymbol{\Lambda}_y$ ($\boldsymbol{\Lambda}_x$) is an identity matrix and $\boldsymbol{\Theta}_y$ ($\boldsymbol{\Theta}_x$) and $\boldsymbol{\nu}_y$ ($\boldsymbol{\nu}_x$) are both null. This can be referred to as a perfect measurement interpretation. However, the equivalence interpretation stated in the last principle presumes that there are actually no measurement equations at all. This is important because under the equivalence interpretation, matrices $\boldsymbol{\Lambda}_y$ ($\boldsymbol{\Lambda}_x$), $\boldsymbol{\Theta}_y$ ($\boldsymbol{\Theta}_x$) and $\boldsymbol{\nu}_y$ ($\boldsymbol{\nu}_x$) are nonexistent rather than fixed quantities, which is assumed under the perfect measurement interpretation. Hence, the **x**-variables are treated as *exogenous* variables with the equivalence interpretation, but they are still treated as *endogenous* with the perfect measurement interpretation. Ultimately, whether **x**-variables are treated as exogenous or endogenous affects the default or automatic parameterization. See the section "Default Parameters" on page 1099 for more details.

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

Presence of Lists		Description	Model Equations	Nonfixed Model Matrices
Presence of Both x- and y-variables				
1	YVAR=, ETAVAR=, XVAR=, XIVAR=	Full model	$\mathbf{y} = \boldsymbol{\nu}_y + \boldsymbol{\Lambda}_y \boldsymbol{\eta} + \boldsymbol{\epsilon}$ $\mathbf{x} = \boldsymbol{\nu}_x + \boldsymbol{\Lambda}_x \boldsymbol{\xi} + \boldsymbol{\delta}$ $\boldsymbol{\eta} = \boldsymbol{\alpha} + \boldsymbol{\beta}\boldsymbol{\eta} + \boldsymbol{\Gamma}\boldsymbol{\xi} + \boldsymbol{\zeta}$	$\boldsymbol{\nu}_y, \boldsymbol{\Lambda}_y, \boldsymbol{\Theta}_y$ $\boldsymbol{\nu}_x, \boldsymbol{\Lambda}_x, \boldsymbol{\Theta}_x, \boldsymbol{\kappa}, \boldsymbol{\Phi}$ $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Psi}$
2	YVAR=, XVAR=, XIVAR=	Full model with $\mathbf{y} \equiv \boldsymbol{\eta}$	$\mathbf{x} = \boldsymbol{\nu}_x + \boldsymbol{\Lambda}_x \boldsymbol{\xi} + \boldsymbol{\delta}$ $\mathbf{y} = \boldsymbol{\alpha} + \boldsymbol{\beta} y + \boldsymbol{\Gamma}\boldsymbol{\xi} + \boldsymbol{\zeta}$	$\boldsymbol{\nu}_x, \boldsymbol{\Lambda}_x, \boldsymbol{\Theta}_x, \boldsymbol{\kappa}, \boldsymbol{\Phi}$ $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Psi}$
3	YVAR=, ETAVAR=, XVAR=	Full model with $\mathbf{x} \equiv \boldsymbol{\xi}$	$\mathbf{y} = \boldsymbol{\nu}_y + \boldsymbol{\Lambda}_y \boldsymbol{\eta} + \boldsymbol{\epsilon}$ $\boldsymbol{\eta} = \boldsymbol{\alpha} + \boldsymbol{\beta}\boldsymbol{\eta} + \boldsymbol{\Gamma} x + \boldsymbol{\zeta}$	$\boldsymbol{\nu}_y, \boldsymbol{\Lambda}_y, \boldsymbol{\Theta}_y$ $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Psi}, \boldsymbol{\kappa}, \boldsymbol{\Phi}$
4	YVAR=, XVAR=	Regression ($\mathbf{y} \equiv \boldsymbol{\eta}$) ($\mathbf{x} \equiv \boldsymbol{\xi}$)	$\mathbf{y} = \boldsymbol{\alpha} + \boldsymbol{\beta} y + \boldsymbol{\Gamma} x + \boldsymbol{\zeta}$, or $(\mathbf{I} - \boldsymbol{\beta})^{-1} y = \boldsymbol{\alpha} + \boldsymbol{\Gamma} x + \boldsymbol{\zeta}$	$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Psi}, \boldsymbol{\kappa}, \boldsymbol{\Phi}$
Presence of x-variables and Absence of y-variables				
5	XVAR=, XIVAR=	Factor model for **x**	$\mathbf{x} = \boldsymbol{\nu}_x + \boldsymbol{\Lambda}_x \boldsymbol{\xi} + \boldsymbol{\delta}$	$\boldsymbol{\nu}_x, \boldsymbol{\Lambda}_x, \boldsymbol{\Theta}_x, \boldsymbol{\kappa}, \boldsymbol{\Phi}$
6	XVAR=	**x**-structures ($\mathbf{x} \equiv \boldsymbol{\xi}$)		$\boldsymbol{\kappa}, \boldsymbol{\Phi}$
Presence of y-variables and Absence of x-variables				
7	YVAR=, ETAVAR=	Factor model for **y**	$\mathbf{y} = \boldsymbol{\nu}_y + \boldsymbol{\Lambda}_y \boldsymbol{\eta} + \boldsymbol{\epsilon}$ $\boldsymbol{\eta} = \boldsymbol{\alpha} + \boldsymbol{\beta}\boldsymbol{\eta} + \boldsymbol{\zeta}$	$\boldsymbol{\nu}_y, \boldsymbol{\Lambda}_y, \boldsymbol{\Theta}_y$ $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Psi}$
8	YVAR=	**y**-structures ($\mathbf{y} \equiv \boldsymbol{\eta}$)	$\mathbf{y} = \boldsymbol{\alpha} + \boldsymbol{\beta} y + \boldsymbol{\zeta}$, or $(\mathbf{I} - \boldsymbol{\beta})^{-1} y = \boldsymbol{\alpha} + \boldsymbol{\zeta}$	$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Psi}$
9	YVAR=, ETAVAR=, XIVAR=	Second-order factor model	$\mathbf{y} = \boldsymbol{\nu}_y + \boldsymbol{\Lambda}_y \boldsymbol{\eta} + \boldsymbol{\epsilon}$ $\boldsymbol{\eta} = \boldsymbol{\alpha} + \boldsymbol{\beta}\boldsymbol{\eta} + \boldsymbol{\Gamma}\boldsymbol{\xi} + \boldsymbol{\zeta}$	$\boldsymbol{\nu}_y, \boldsymbol{\Lambda}_y, \boldsymbol{\Theta}_y$ $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Psi}, \boldsymbol{\kappa}, \boldsymbol{\Phi}$
10	YVAR=, XIVAR=	Factor model ($\mathbf{y} \equiv \boldsymbol{\eta}$)	$\mathbf{y} = \boldsymbol{\alpha} + \boldsymbol{\beta} y + \boldsymbol{\Gamma}\boldsymbol{\xi} + \boldsymbol{\zeta}$, or $(\mathbf{I} - \boldsymbol{\beta})^{-1} y = \boldsymbol{\alpha} + \boldsymbol{\Gamma}\boldsymbol{\xi} + \boldsymbol{\zeta}$	$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Psi}, \boldsymbol{\kappa}, \boldsymbol{\Phi}$

Models 1, 2, 3, and 4—Presence of Both $\mathbf{x}$*- and* $\mathbf{y}$*-Variables*

Submodels 1, 2, 3, and 4 are characterized by the presence of both **x**- and **y**-variables in the model. In fact, Model 1 is the full model with the presence of all four types of variables. All twelve model matrices are parameter matrices in this model.

Depending on the absence of the latent factor lists, manifest variables can replace the role of the latent factors in Models 2–4. For example, the absence of the ETAVAR= list in Model 2 means **y** is equivalent to $\boldsymbol{\eta}$ ($\mathbf{y} \equiv \boldsymbol{\eta}$). 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 $\boldsymbol{\xi}$ ($\mathbf{x} \equiv \boldsymbol{\xi}$) 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 $\boldsymbol{\eta}$ ($\mathbf{y} \equiv \boldsymbol{\eta}$) and **x** is equivalent to $\boldsymbol{\xi}$ ($\mathbf{x} \equiv \boldsymbol{\xi}$). 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 $\mathbf{y}$ can regress on itself in the model equation, the regression description is not totally accurate for Model 4. Nonetheless, if $\boldsymbol{\beta}$ is a null matrix, the equation describes a multivariate regression model with outcome variables $\mathbf{y}$ and predictor variables $\mathbf{x}$. 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 $\boldsymbol{\Phi}$ and $\boldsymbol{\kappa}$ are now the covariance matrix and mean vector, respectively, of manifest variables $\mathbf{x}$, while in Model 1 (the complete model) these matrices are of the latent factors $\boldsymbol{\xi}$.

Models 5 and 6 — Presence of x*-Variables and Absence of* y*-Variables*

Models 5 and 6 are characterized by the presence of the $\mathbf{x}$-variables and the absence of $\mathbf{y}$-variables.

Model 5 is simply a factor model for measured variables $\mathbf{x}$, with $\boldsymbol{\Lambda}_x$ representing the factor loading matrix, $\boldsymbol{\Theta}_x$ the error covariance matrix, and $\boldsymbol{\Phi}$ the factor covariance matrix. If mean structures are modeled, $\boldsymbol{\kappa}$ represents the factor means and $\boldsymbol{\nu}_x$ is the intercept vector. This is the TYPE 1 submodel in LISREL VI (Jöreskog and Sörbom 1985).

Model 6 is a special case where there is no model equation. You specify the mean and covariance structures (in $\boldsymbol{\kappa}$ and $\boldsymbol{\Phi}$, respectively) for the manifest variables $\mathbf{x}$ directly. The $\mathbf{x}$-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 1220 for more details.

Note that because $\boldsymbol{\eta}$-variables cannot exist in the absence of $\mathbf{y}$-variables (see one of the three aforementioned principles for deriving submodels), adding the ETAVAR= list alone to these two submodels does not generate new submodels that can be analyzed by PROC CALIS.

Models 7, 8 ,9 and 10—Presence of y*-Variables and Absence of* x*-Variables*

Models 7–10 are characterized by the presence of the $\mathbf{y}$-variables and the absence of $\mathbf{x}$-variables.

Model 7 is a factor model for $\mathbf{y}$-variables (TYPE 3B submodel in LISREL VI). It is similar to Model 5, but with regressions among latent factors allowed. When $\boldsymbol{\beta}$ is null, Model 7 functions the same as Model 5. It becomes a factor model for $\mathbf{y}$-variables, with $\boldsymbol{\Lambda}_y$ representing the factor loading matrix, $\boldsymbol{\Theta}_y$ the error covariance matrix, $\boldsymbol{\Psi}$ the factor covariance matrix, $\boldsymbol{\alpha}$ the factor means, and $\boldsymbol{\nu}_y$ the intercept vector.

Model 8 (TYPE 2B submodel in LISREL VI) is a model for studying the mean and covariance structures of $\mathbf{y}$-variables, with regression among $\mathbf{y}$-variables allowed. When $\boldsymbol{\beta}$ is null, the mean structures of $\mathbf{y}$ are specified in $\boldsymbol{\alpha}$ and the covariance structures are specified in $\boldsymbol{\Psi}$. This is similar to Model 6. However, there is an important distinction. In Model 6, the $\mathbf{x}$-variables are treated as exogenous (no model equation at all). But the $\mathbf{y}$-variables are treated as endogenous in Model 8 (with or without $\boldsymbol{\beta} = 0$). Consequently, the default parameterization would be different for these two submodels. See the section "Default Parameters" on page 1099 for details about the default parameterization.

Model 9 represents a modified version of the second-order factor model for $\mathbf{y}$. It would be a standard second-order factor model when $\boldsymbol{\beta}$ is null. This is the TYPE 3A submodel in LISREL VI. With $\boldsymbol{\beta}$ being null, $\boldsymbol{\eta}$ represents the first-order factors and $\boldsymbol{\xi}$ represents the second-order factors. The first- and second-order factor loading matrices are $\boldsymbol{\Lambda}_y$ and $\boldsymbol{\Gamma}$, respectively.

Model 10 is another form of factor model when $\boldsymbol{\beta}$ is null, with factors represented by $\boldsymbol{\xi}$ and manifest variables represented by $\mathbf{y}$. However, if $\boldsymbol{\beta}$ is indeed a null matrix in applications, you might want to use Model 5, in which the factor model specification is more direct and intuitive.

Default Parameters in the LISMOD Model

When a model matrix is defined in a LISMOD model, you can specify fixed values or free parameters for the elements of the matrix by the MATRIX statement. All other unspecified elements in the matrix are set by default. There are two types of default parameters for the LISMOD model matrices: one is free parameters; the other is fixed zeros.

The following sets of parameters are free parameters by default:

- the diagonal elements of the _THETAX_, _THETAY_, and _PSI_ matrices; these elements represent the error variances in the model
- all elements of the _PHI_ matrix; these elements represent the variances and covariance among exogenous variables in the model
- all elements in the _NUX_ and _NUY_ vectors if the mean structures are modeled; these elements represent the intercepts of the observed variables
- all elements in the _ALPHA_ vector if a YVAR= list is specified but an ETAVAR= list is not specified and the mean structures are modeled; these elements represent the intercepts of the **y**-variables
- all elements in the _KAPPA_ vector if an XVAR= list is specified but an XIVAR= list is not specified and the mean structures are modeled; these elements represent the means of the **x**-variables

PROC CALIS names the default free parameters with the _Add prefix and a unique integer suffix. You can override the default free parameters by explicitly specifying them as free, constrained, or fixed parameter in the MATRIX statements for the matrices.

Parameters that are not default free parameters in the LISMOD model are fixed zeros by default. You can override almost all of these default fixed zeros of the LISMOD model by using the MATRIX statements for the matrices. The only set of default fixed zeros that you cannot override is the set of the diagonal elements of the _BETA_ matrix. These fixed zeros reflect a model restriction that precludes variables from having direct effects on themselves.

The MSTRUCT Model

In contrast to other modeling languages where the mean and covariance structures are implied from the specification of equations, paths, variable-factor relations, mean parameters, variance parameters, or covariance parameters, the MSTRUCT modeling language is supported in PROC CALIS for modeling mean and covariance structures directly.

A simple example for using the MSTRUCT modeling language is the testing of a covariance model with equal variances and covariances. Suppose that a variable was measured five times in an experiment. The covariance matrix of these five measurements is hypothesized to have the structure

$$\mathbf{\Sigma} = \mathbf{\Sigma}(\theta)$$

where

$$\theta = (\phi, \tau)$$

and

$$\mathbf{\Sigma}(\theta) = \begin{pmatrix} \phi & \tau & \tau & \tau & \tau \\ \tau & \phi & \tau & \tau & \tau \\ \tau & \tau & \phi & \tau & \tau \\ \tau & \tau & \tau & \phi & \tau \\ \tau & \tau & \tau & \tau & \phi \end{pmatrix}$$

For model structures that are hypothesized directly in the covariance matrix, the MSTRUCT modeling language is the most convenient to use. You can also use other general modeling languages such as LINEQS, PATH, or RAM to fit the same model structures, but the specification is less straightforward and more error-prone. For convenience, models that are specified using the MSTRUCT modeling language are called MSTRUCT models.

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
$\mathbf{\Sigma}$	_COV_ or _MSTRUCTCOV_	Structured covariance matrix	$p \times p$
$\boldsymbol{\mu}$	_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 to be analyzed in your MSTRUCT model by this statement:

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

See the MSTRUCT statement on page 1130 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 shorthand for specifying phi five times, once for each of the five diagonal elements in the covariance matrix. All other lower triangular elements are fitted by the tau parameter, as shown in the MATRIX statement. For example, with `[3,]` 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 1111 for details about the syntax.

Default Parameters in the MSTRUCT Model

By using the MATRIX statements, you can specify either fixed values or free parameters (with or without initial values) for the elements in the _COV_ and _MEAN_ model matrices. If some or all elements are not specified, default parameters are applied to the MSTRUCT. There are two types of default parameters: one is free parameters; the other is fixed zeros. They are applied in different situations.

If you do not specify *any* elements of the _COV_ matrix with the MATRIX statement, all elements of the _COV_ matrix are free parameters by default. PROC CALIS names the default free parameters with the _Add prefix and a unique integer suffix. However, if you specify *at least one* fixed or free parameter of the _COV_ matrix with the MATRIX statement, then all other unspecified elements of the _COV_ matrix are fixed zeros by default.

If the mean structures are modeled, the same treatment applies to the _MEAN_ vector. That is, if you do not specify any elements of the _MEAN_ vector with the MATRIX statement, all elements of the _MEAN_ vector are free parameters by default. However, if you specify *at least one* fixed or free parameter of the _MEAN_ vector with the MATRIX statement, then all other unspecified elements of the _MEAN_ vector are fixed zeros by default.

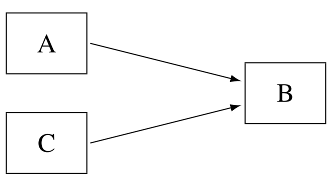
How and Why the Default Parameters Are Treated Differently in the MSTRUCT Model

Notice that the default parameter treatment of the MSTRUCT model is quite different from other types of models such as FACTOR, LINEQS, LISMOD, RAM, or PATH. For these models, unspecified variances and covariances among exogenous variables are all free parameters by default. However, for the MSTRUCT model, either default free parameters or fixed zeros are generated depending on whether at least one element of the covariance matrix is specified. The reason for this different treatment is that you fit the covariance structure directly by using the MSTRUCT modeling language. Hence, in an MSTRUCT model there is no information regarding the functional relationships among the variables that indicates whether the variables are exogenous or endogenous in the model. Hence, PROC CALIS cannot assume default free parameters based on the exogenous or endogenous variable types.

Because of this special default parameter treatment, when fitting an MSTRUCT model you must make sure that each diagonal element in your _COV_ matrix is set as a free, constrained, or fixed parameter, in accordance with your theoretical model. If you specify some elements in the model matrix but omit the specification of other diagonal elements, the default fixed zero variances would lead to a nonpositive definite _COV_ model matrix, making the model fitting problematic.

The PATH Model

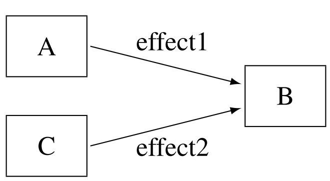
The PATH modeling language is supported in PROC CALIS as a more intuitive modeling tool. It is designed so that specification by using the PATH modeling language translates effortlessly from the path diagram. For example, consider the following simple path diagram:



You can use the following PATH statement to specify the paths easily:

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

There are two path entries in the PATH statement: one is for the path A ---> B, and the other is for the path C ---> B. Sometimes you might want to name the effect parameters in the path diagram, as shown in the following:



You can specify the paths and the parameters together in the following statement:

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

In the first entry of the PATH statement, the path A ---> B is specified together with the path coefficient (effect) effect1. Similarly, in the second entry, the C ---> B path is specified together with the effect parameter effect2. In addition to the path coefficients (effects) in the path diagram, you can also specify other types of parameters by using the PVAR and PCOV statements. See the section “A Structural Equation Example” on page 1002 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 single-headed arrow or it serves as an outcome variable in at least one path. Otherwise, it is exogenous. In the preceding path diagram, for example, variable B is endogenous and both variables A and C are exogenous. Note that although any variable that serves as an outcome variable at least in one path must be endogenous, it does not mean that all endogenous variables must serve only as outcome variables in all paths. An endogenous variable in a model might also serve as a predictor variable in a path. For example, variable B in the following PATH statement is an endogenous variable, and it serves as an outcome variable in the first path but as a predictor variable in the second path.

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

A variable is a manifest or observed variable in the PATH model if it is measured and exists in the input data set. Otherwise, it is a latent variable. Because error variables are not explicitly defined in the PATH modeling language, all latent variables that are named in the PATH model are *factors*, which are considered to be the systematic source of effects in the model. Each manifest variable in the PATH model can be endogenous or exogenous. The same is true for any latent factor in the PATH model.

Because you do not name error variables in the PATH model, you do not need to specify paths from errors to any endogenous variables. Error terms are implicitly assumed for all endogenous variables in the PATH model. Although error variables are not named in the PATH model, the error variances are expressed equivalently as partial variances of the associated endogenous variables. These partial variances are set by default in the PATH modeling language. Therefore, you do not need to specify error variance parameters explicitly unless constraints on these parameters are desirable in the model. You can use the PVAR statement to specify the error variance or partial variance parameters explicitly.

Naming Variables in the PATH Model

Manifest variables in the PATH model are referenced in the input data set. Their names must not be longer than 32 characters. There are no further restrictions beyond those required by the SAS System. You use the names of manifest variables directly in the PATH model specification.

Because you do not name error variables in the PATH model, all latent variables named in the PATH model specification are factors (non-errors). Factor names in the PATH model must not be longer than 32 characters, and they should be different from the manifest variables. Unlike the LINEQS model, you do not need to use 'F' or 'f' prefix to denote latent factors in the PATH model. As a general naming convention, you should not use Intercept as either a manifest or latent variable name. See the section "Naming Variables and Parameters" on page 1238 for these general rules about naming variables and parameters.

Specification of the PATH Model

(1) Specification of Effects or Paths

You specify the "causal" paths or linear functional relationships among variables in the PATH statement. For example, if there is a path from v2 to v1 in your model and the effect parameter is named parm1 with a starting value at 0.5, you can use either of these specifications:

```
path     v1 <---  v2    = parm1(0.5);
path     v2 --->  v1    = parm1(0.5);
```

If you have more than one path in your model, path specifications should be separated by commas, as shown in the following PATH statement:

```
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 1137 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 CALIS), you are specifying the variance parameter of the variable. Otherwise, you are specifying the partial or error variance for an endogenous variable.

You do not need to supply the parameter names for the variances or partial variances if these parameters are not constrained. For example, the following statement specifies the unnamed free parameters for variances or partial variances of v1 and v2:

```
pvar     v1 v2;
```

If you have more than one variance or partial variance parameter to specify in your model, you can put a variable list on the left-hand side of the equal sign, and a parameter list on the right-hand side, as shown in the following PVAR statement specification:

```
pvar
   v1 v2 v3 = parm1(0.5) parm2 parm3;
```

In the specification, variance or partial variance parameters for variables 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 CALIS.

You can also separate the specifications into several entries in the PVAR statement. Entries should be separated by commas. For example, the preceding specification is equivalent to the following specification:

```
pvar
   v1 = parm1 (0.5),
   v2 = parm2,
   v3 = parm3;
```

Because the PVAR statement can be used only once in each model specification, all variance and partial variance parameters in the model must be specified together in a single PVAR statement. See the PVAR statement on page 1149 for more details about the syntax.

(3) Specification of Covariances and Partial Covariances

If you want to specify the (partial) covariance between two variables v3 and v4 as a parameter named parm4 with a starting value at 3, you can use the following PCOV statement specification:

```
pcov  v3  v4 = parm4 (5.);
```

Whether parm4 is a covariance or partial covariance parameter depends on the variable types of v3 and v4. If both v3 and v4 are exogenous variables (manifest or latent), parm4 is a covariance parameter between v3 and v4. If both v3 and v4 are endogenous variables (manifest or latent), parm4 is a parameter for the covariance between the errors for v3 and v4. In other words, it is a partial covariance or error covariance parameter for v3 and v4.

A less common case is when one of the variables is exogenous and the other is endogenous. In this case, parm4 is a parameter for the partial covariance between the endogenous variable and the exogenous variable, or the covariance between the error for the endogenous variable and the exogenous variable. Fortunately, such covariances are relatively uncommon in statistical modeling. Their uses confuse the roles of systematic and unsystematic sources in the model and lead to difficulties in interpretations. Therefore, you should almost always avoid this kind of partial covariance.

Like the syntax of the PVAR statement, you can specify a list of (partial) covariance parameters in the PCOV statement. For example, consider the following statement:

```
pcov
   v1 v2 = parm4,
   v1 v3 = parm5,
   v2 v3 = parm6;
```

In the specification, three (partial) covariance parameters parm4, parm5, and parm6 are specified, respectively, for the variable pairs (v1,v2), (v1,v3), and (v2,v3). Entries for (partial) covariance specification are separated by commas.

Again, if all these covariances are not constrained, you can omit the names for the parameters. For example, the preceding specification can be specified as the following statement when the three covariances are free parameters in the model:

```
pcov
   v1 v2,
   v1 v3,
   v2 v3;
```

Or, you can simply use the following within-list covariance specification:

```
pcov
   v1 v2 v3;
```

Three covariance parameters are generated by this specification.

Because the PCOV statement can be used only once in each model specification, all covariance and partial covariance parameters in the model must be specified together in a single PCOV statement. See the PCOV statement on page 1147 for more details about the syntax.

(4) Specification of Means and Intercepts

Means and intercepts are specified when the mean structures of the model are of interest. You can specify mean and intercept parameters in the MEAN statement. For example, consider the following statement:

```
mean     V5 = parm5(11.);
```

If V5 is an exogenous variable (which is determined by PROC CALIS automatically), you are specifying parm5 as the mean parameter of V5. If V5 is an endogenous variable, you are specifying parm5 as the intercept parameter for V5.

Because each named variable in the PATH model is either exogenous or endogenous (exclusively), each variable in the PATH model would have either a mean or an intercept parameter (but not both) to specify in the MEAN statement. Like the syntax of the PVAR statement, you can specify a list of mean or intercept parameters in the MEAN statement. For example, in the following statement you specify a list of mean or intercept parameters for variables v1-v4:

```
mean
   v1-v4 = parm6-parm9;
```

This specification is equivalent to the following specification with four entries of parameter specifications:

```
mean
   v1 = parm6,
   v2 = parm7,
   v3 = parm8,
   v4 = parm9;
```

Again, entries in the MEAN statement must be separated by commas, as shown in the preceding statement.

Because the MEAN statement can be used only once in each model specification, all mean and intercept parameters in the model must be specified together in a single MEAN statement. See the MEAN statement on page 1125 for more details about the syntax.

Specifying Parameters without Initial Values

If you do not have any knowledge about the initial value for a parameter, you can omit the initial value specification and let PROC CALIS compute it. For example, you can provide just the parameter locations and parameter names as in the following specification:

```
path    v1 <--- v2   = parm1;
   pvar v2 = parm2,
        v1 = parm3;
```

Specifying Fixed Parameter Values

If you want to specify a fixed parameter value, you do not need to provide a parameter name. Instead, you provide the fixed value (without parentheses) in the specification.

For example, in the following statement the path coefficient for the path is fixed at 1.0 and the (partial) variance of F1 is also fixed at 1.0:

```
path    v1 <--- F1  = 1.;
   pvar
        F1 = 1.;
```

A Complete PATH Model Specification Example

The following specification shows a more complete PATH model specification:

```
path     v1 <---  v2 ,
         v1 <---  v3 ;
   pvar  v1,
         v2 = parm3,
         v3 = parm3;
   pcov  v3 v2 =  parm5(5.);
```

The two paths specified in the PATH statement have unnamed free effect parameters. These parameters are named by PROC CALIS with the _Parm prefix and unique integer suffixes. The error variance of v1 is an unnamed parameter, while the variances of v2 and v3 are constrained by using the same parameter parm3. The covariance between v2 and v3 is a free parameter named parm5, with a starting value of 5.0.

Default Parameters in the PATH Model

There are two types of default parameters of the PATH model. One is the free parameters; the other is the fixed constants.

The following sets of parameters are free parameters by default:

- the variances or partial (or error) variances of all variables, manifest or latent
- the covariances among all exogenous (independent) manifest or latent variables
- the means of all exogenous (independent) manifest variables if the mean structures are modeled
- the intercepts of all endogenous (dependent) manifest variables if the mean structures are modeled

For each of the default free parameters, PROC CALIS generates a parameter name with the _Add prefix and a unique integer suffix. Parameters that are not default free parameters in the PATH model are fixed zeros by default. You can override almost all of the default zeros of the PATH model by using the MEAN, PATH, PCOV, and MEAN statements. The only exception is the single-headed path that has the same variable on both sides. That is, the following specification is not accepted by PROC CALIS:

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

This path should always has a zero coefficient, which is treated as a model restriction that prevents a variable from having a direct effect on itself.

Relating the PATH Model to the RAM Model

Mathematically, the PATH model is essentially the RAM model. You can consider the PATH model to share exactly the same set of model matrices as in the RAM model. See the section "Model Matrices in the RAM Model" on page 1230 and the section "Summary of Matrices and Submatrices in the RAM Model" on page 1233 for details about the RAM model matrices. In the RAM model, the **A** matrix contains effects or path coefficients for describing relationships among variables. In the PATH model, you specify these effect or coefficient parameters in the PATH statement. The **P** matrix in the RAM model contains (partial) variance and (partial) covariance parameters. In the PATH model, you use the PVAR and PCOV statements to specify these parameters. The **W** vector in the RAM model contains the mean and intercept parameters, while in the PATH model you use the MEAN statement to specify these parameters. By using these model matrices in the PATH model, the covariance and mean structures are derived in the same way as they are derived in the RAM model. See the section "The RAM Model" on page 1229 for derivations of the model structures.

The RAM Model

The RAM modeling language is adapted from the basic RAM model developed by McArdle (1980). For brevity, models specified by the RAM modeling language are called RAM models. You can also specify these so-called RAM models by other general modeling languages that are supported in PROC CALIS.

Types of Variables in the RAM Model

A variable in the RAM model is manifest if it is observed and is defined in the input data set. A variable in the RAM model is latent if it is not manifest. Because error variables are not explicitly named in the RAM model, all latent variables in the RAM model are considered as factors (non-error-type latent variables).

A variable in the RAM model is endogenous if it ever serves as an outcome variable in the RAM model. That is, an endogenous variable has at least one path (or an effect) from another variable in the model. A variable is exogenous if it is not endogenous. Endogenous variables are also referred to as dependent variables, while exogenous variables are also referred to as independent variables.

In the RAM model, distinctions between exogenous and endogenous and between latent and manifest for variables are not essential to the definitions of model matrices, although they are useful for conceptual understanding when the model matrices are partitioned.

Naming Variables in the RAM Model

Manifest variables in the RAM model are referenced in the input data set. Their names must not be longer than 32 characters. There are no further restrictions beyond those required by the SAS System.

Latent variables in the RAM model are those not being referenced in the input data set. Their names must not be longer than 32 characters. Unlike the LINEQS model, you do not need to use any specific prefix (for example, ‘F’ or ‘f’) for the latent factor names. The reason is that error or disturbance variables in the RAM model are not named explicitly in the RAM model. Thus, any variable names that are not referenced in the input data set are for latent factors.

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

Model Matrices in the RAM Model

In terms of the number of model matrices involved, the RAM model is the simplest among all the general structural equations models that are supported by PROC CALIS. Essentially, there are only three model matrices in the RAM model: one for the interrelationships among variables, one for the variances and covariances, and one for the means and intercepts. These matrices are discussed in the following subsections.

Matrix $\mathbf{A}$ *($n_a \times n_a$) : Effects of Column Variables on Row Variables*

The row and column variables of matrix $\mathbf{A}$ 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 $\mathbf{A}$ 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 $\mathbf{A}$ determines whether a variable is endogenous or exogenous. A variable in the RAM model is endogenous if its associated row in the $\mathbf{A}$ matrix has at least one nonzero entry. Any other variable in the RAM model is exogenous.

Mathematically, you do not need to arrange the set of variables for matrix **A** in a particular order, as long as the order of variables is the same for the rows and the columns. However, arranging the variables according to whether they are endogenous or exogenous is useful for showing the partitions of the model matrices and certain mathematical properties. See the section "Partitions of the RAM Model Matrices and Some Restrictions" on page 1232 for details.

Matrix $\mathbf{P}$ *($n_a \times n_a$): Variances, Covariances, Partial Variances, and Partial Covariances*

The row and column variables of matrix **P** refer to the same set of manifest and latent variables that are defined in the RAM model matrix **A**. The diagonal entries of **P** contain variances or partial variances of variables. If a variable is exogenous, then the corresponding diagonal element in the **P** matrix represents its variance. Otherwise, the corresponding diagonal element in the **P** matrix represents its partial variance. This partial variance is an unsystematic source of variance that is not explained by the interrelationships of variables in the model. In most cases, you can interpret a partial variance as the error variance for an endogenous variable.

The off-diagonal elements of **P** contain covariances or partial covariances among variables. An off-diagonal element in **P** that is associated with exogenous row and column variables represents covariance between the two exogenous variables. An off-diagonal element in **P** that is associated with endogenous row and column variables represents *partial* covariance between the two variables. This partial covariance is unsystematic, in the sense that it is not explained by the interrelationships of variables in the model. In most cases, you can interpret a partial covariance as the error covariance between the two endogenous variables involved. An off-diagonal element in **P** that is associated with one exogenous variable and one endogenous variable in the row and column represents the covariance between the exogenous variable and the error of the endogenous variable. While this interpretation sounds a little awkward and inelegant, this kind of covariance, fortunately, is rare in most applications.

Vector $\mathbf{W}$ *($n_a \times 1$): Intercepts and Means*

The row variables of vector **W** refer to the same set of manifest and latent variables that are defined in the RAM model matrix **A**. Elements in **W** represent either intercepts or means. An element in **W** that is associated with an exogenous row variable represents the mean of the variable. An element in **W** that is associated with an endogenous row variable represents the intercept term for the variable.

Covariance and Mean Structures

Assuming that $(\mathbf{I} - \mathbf{A})$ is invertible, where **I** is an identity matrix of the same dimension as **A**, the structured covariance matrix of all variables (including latent variables) in the RAM model is shown as follows:

$$\boldsymbol{\Sigma}_a = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{P}(\mathbf{I} - \mathbf{A})^{-1\prime}$$

The structured mean vector of all variables is shown as follows:

$$\boldsymbol{\mu}_a = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{W}$$

The covariance and mean structures of all manifest variables are obtained by selecting the elements in $\boldsymbol{\Sigma}_a$ and $\boldsymbol{\mu}_a$. This can be achieved by defining a selection matrix **G** of dimensions $n \times n_a$, where n is the number of manifest variables in the model. The selection matrix **G** contains zeros and ones as its elements. Each row of $\mathbf{G}$ has exactly one nonzero element at the position that corresponds to the location of a manifest row variable in $\boldsymbol{\Sigma}_a$ or $\boldsymbol{\mu}_a$. With each row of $\mathbf{G}$ selecting a distinct manifest variable, the structured covariance matrix of all manifest variables is expressed as the following:

$$\boldsymbol{\Sigma} = \mathbf{G}\boldsymbol{\Sigma}_a\mathbf{G}'$$

The structured mean vector of all observed variables is expressed as the following:

$$\boldsymbol{\mu} = \mathbf{G}\boldsymbol{\mu}_a$$

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 $\mathbf{A}$ *Matrix*

The $\mathbf{A}$ matrix is partitioned as

$$\mathbf{A} = \begin{pmatrix} \boldsymbol{\beta} & \boldsymbol{\gamma} \\ 0 & 0 \end{pmatrix}$$

where $\boldsymbol{\beta}$ is an $n_d \times n_d$ matrix for paths or effects from (column) endogenous variables to (row) endogenous variables and $\boldsymbol{\gamma}$ is an $n_d \times n_i$ matrix for paths (effects) from (column) exogenous variables to (row) endogenous variables.

As shown in the matrix partitions, there are four submatrices. The two submatrices at the lower parts are seemingly structured to zeros. However, this should not be interpreted as restrictions imposed by the model. The zero submatrices are artifacts created by the exogenous-endogenous arrangement of the row and column variables. The only restriction on the $\mathbf{A}$ matrix is that the diagonal elements must all be zeros. This implies that the diagonal elements of the submatrix $\boldsymbol{\beta}$ are also zeros. This restriction prevents a direct path from any endogenous variable to itself. There are no restrictions on the pattern of $\boldsymbol{\gamma}$.

It is useful to denote the lower partitions of the $\mathbf{A}$ matrix by $\mathbf{A}_{LL}$ (lower left) and $\mathbf{A}_{LR}$ (lower right) so that

$$\mathbf{A} = \begin{pmatrix} \boldsymbol{\beta} & \boldsymbol{\gamma} \\ \mathbf{A}_{LL} & \mathbf{A}_{LR} \end{pmatrix}$$

Although they are zero matrices in the initial model specification, their entries could become non-zero (paths) in an improved model when you modify your model by using the Lagrange multiplier statistics (see the section "Modification Indices" on page 1277 or the MODIFICATION option). Hence, you might need to reference these two submatrices when you apply the customized LM tests on them during the model modification process (see the LMTESTS statement).

For the purposes of defining specific parameter regions in customized LM tests, you might also partition the $\mathbf{A}$ matrix in other ways. First, you can partition $\mathbf{A}$ into the left and right portions,

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{Left} & \mathbf{A}_{Right} \end{pmatrix}$$

where $\mathbf{A}_{Left}$ is top-down concatenation of the $\boldsymbol{\beta}$ and $\mathbf{A}_{LL}$ matrices and $\mathbf{A}_{Right}$ is the top-down concatenation of the $\boldsymbol{\gamma}$ and $\mathbf{A}_{LR}$ matrices. Second, you can partition $\mathbf{A}$ into the upper and lower portions,

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{Upper} \\ \mathbf{A}_{Lower} \end{pmatrix}$$

where $\mathbf{A}_{Upper}$ is the side-by-side concatenation of the $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ matrices and $\mathbf{A}_{Lower}$ is the side-by-side concatenation of the $\mathbf{A}_{LL}$ and $\mathbf{A}_{LR}$ matrices.

In your initial model, because of the arrangement of the endogenous and exogenous variables $\mathbf{A}_{Lower}$ is a null matrix. But if you improve your model by applying the LM tests on the entries in $\mathbf{A}_{Lower}$, some of these entries might become free paths in your improved model. Hence, some exogenous variables in your initial model now become endogenous variables in your improved model. For this reason, $\mathbf{A}_{Lower}$ is also designated as a parameter region for *new* endogenous variables, which is exactly what the NEWENDO region means in the LMTESTS statement.

Partition of the P *Matrix*

The $\mathbf{P}$ matrix is partitioned as

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}'_{21} \\ \mathbf{P}_{21} & \mathbf{P}_{22} \end{pmatrix}$$

where $\mathbf{P}_{11}$ is an $n_d \times n_d$ partial covariance matrix for the endogenous variables, $\mathbf{P}_{22}$ is an $n_i \times n_i$ covariance matrix for the exogenous variables, and $\mathbf{P}_{21}$ is an $n_i \times n_d$ covariance matrix between the exogenous variables and the error terms for the endogenous variables. Because $\mathbf{P}$ is symmetric, $\mathbf{P}_{11}$ and $\mathbf{P}_{22}$ are also symmetric.

There are virtually no model restrictions placed on these submatrices. However, in most statistical applications, errors for endogenous variables represent unsystematic sources of effects and therefore they are not to be correlated with other systematic sources such as the exogenous variables in the RAM model. This means that in most practical applications $\mathbf{P}_{21}$ would be a null matrix, although this is not enforced in PROC CALIS.

Partition of the W *Vector*

The $\mathbf{W}$ vector is partitioned as

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

where $\boldsymbol{\alpha}$ is an $n_d \times 1$ vector for intercepts of the endogenous variables and $\boldsymbol{\nu}$ is an $n_i \times 1$ vector for the means of the exogenous variables. There is no model restriction on these subvectors.

Summary of Matrices and Submatrices in the RAM Model

Let n_a be the total number of manifest and latent variables in the RAM model. Of these n_a variables, n_d are endogenous and n_i are exogenous. Suppose that the rows and columns of the RAM model matrices $\mathbf{A}$ and $\mathbf{P}$ and the rows of $\mathbf{W}$ 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			
$\mathbf{A}$	_A_ or _RAMA_	Effects of column variables on row variables, or paths from the column variables to the row variables	$n_a \times n_a$
$\mathbf{P}$	_P_ or _RAMP_	(Partial) variances and covariances	$n_a \times n_a$
$\mathbf{W}$	_W_ or _RAMW_	Intercepts and means	$n_a \times 1$
Submatrices			
$\boldsymbol{\beta}$	_RAMBETA_	Effects of endogenous variables on endogenous variables	$n_d \times n_d$
$\boldsymbol{\gamma}$	_RAMGAMMA_	Effects of exogenous variables on endogenous variables	$n_d \times n_i$
$\mathbf{A}_{LL}$	_RAMA_LL_	The null submatrix at the lower left portion of _A_	$n_i \times n_d$
$\mathbf{A}_{LR}$	_RAMA_LR_	The null submatrix at the lower right portion of _A_	$n_i \times n_i$
$\mathbf{A}_{Left}$	_RAMA_LEFT_	The left portion of _A_, including $\boldsymbol{\beta}$ and $\mathbf{A}_{LL}$	$n_a \times n_d$
$\mathbf{A}_{Right}$	_RAMA_RIGHT_	The right portion of _A_, including $\boldsymbol{\gamma}$ and $\mathbf{A}_{LR}$	$n_a \times n_i$
$\mathbf{A}_{Upper}$	_RAMA_UPPER_	The upper portion of _A_, including $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$	$n_d \times n_a$
$\mathbf{A}_{Lower}$	_RAMA_LOWER_	The lower portion of _A_, including $\mathbf{A}_{LL}$ and $\mathbf{A}_{LR}$	$n_i \times n_a$
$\mathbf{P}_{11}$	_RAMP11_	Error variances and covariances for endogenous variables	$n_d \times n_d$
$\mathbf{P}_{21}$	_RAMP21_	Covariances between exogenous variables and error terms for endogenous variables	$n_d \times n_i$
$\mathbf{P}_{22}$	_RAMP22_	Variances and covariances for exogenous variables	$n_i \times n_i$
$\boldsymbol{\alpha}$	_RAMALPHA_	Intercepts for endogenous variables	$n_d \times 1$
$\boldsymbol{\nu}$	_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. The VAR= option specifies the variables (including manifest and latent) in the model. For example, the following statement specifies five variables in the model:

```
RAM
   var= v1 v2 v3;
```

The order of variables in the VAR= option is important. The same order is used for the row and column variables in the model matrices. After you specify the variables in the model, you can specify three types of parameters, which correspond to the elements in the three model matrices. The three types of RAM entries are described in the following.

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

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

```
RAM
   var= v1 v2 v3,
   _A_   1   2  parm1(0.5);
```

The *ram-entry* that starts with _A_ means that an element of the ram matrix **A** is being specified. The row number and the column number of this element are 1 and 2, respectively. With reference to the VAR= list, the row number 1 refers to variable v1, and the column number 2 refers to variable v2. Therefore, the effect of V2 on V1 is a parameter named parm1, with an initial value of 0.5.

You can specify fixed values in the *ram-entries* too. Suppose the effect of v3 on v1 is fixed at 1.0. You can use the following specification:

```
RAM
   var= v1 v2 v3,
   _A_   1   2  parm1(0.5),
   _A_   1   3  1.0;
```

(2) Specification of the Latent Factors in the Model

In the RAM model, you specify the list of variables in VAR= list of the RAM statement. The list of variables can include the latent variables in the model. Because observed variables have references in the input data sets, those variables that do not have references in the data sets are treated as latent factors automatically. Unlike the LINEQS model, you do not need to use ‘F’ or ‘f’ prefix to denote latent factors in the RAM model. It is recommended that you use meaningful names for the latent factors. See the section “Naming Variables and Parameters” on page 1238 for the general rules about naming variables and parameters.

For example, suppose that SES_Factor and Education_Factor are names that are not used as variable names in the input data set. These two names represent two latent factors in the model, as shown in the following specification:

```
RAM
   var= v1 v2 v3 SES_FACTOR Education_Factor,
   _A_   1    4    b1,
   _A_   2    5    b2,
   _A_   3    5    1.0;
```

This specification shows that the effect of SES_Factor on v1 is a free parameter named b1, and the effects of Education_Factor on v2 and v3 are a free parameter named b2 and a fixed value of 1.0, respectively.

However, naming latent factors is not compulsory. The preceding specification is equivalent to the following specification:

```
RAM
   var= v1 v2 v3,
   _A_   1    4    b1,
   _A_   2    5    b2,
   _A_   3    5    1.0;
```

Although you do not name the fourth and the fifth variables in the VAR= list, PROC CALIS generates the names for these two latent variables. In this case, the fourth variable is named _Factor1 and the fifth variable is named _Factor2.

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

Suppose now you want to specify the variance of v2 as a free parameter named parm2. You can add a new *ram-entry* for this variance parameter, as shown in the following statement:

```
RAM
   var= v1 v2 v3,
   _A_   1   2  parm1(0.5),
   _A_   1   3  1.0,
   _P_   2   2  parm2;
```

The *ram-entry* that starts with _P_ means that an element of the RAM matrix **P** is being specified. The (2,2) element of **P**, which is the variance of v2, is a parameter named parm2. You do not specify an initial value for this parameter.

You can also specify the error variance of v1 similarly, as shown in the following statement:

```
RAM
   var= v1 v2 v3,
   _A_   1   2  parm1(0.5),
   _A_   1   3  1.0,
   _P_   2   2  parm2,
   _P_   1   1;
```

In the last *ram-entry*, the (1,1) element of **P**, which is the error variance of v1, is an unnamed free parameter.

Covariance parameters are specified in the same manner. For example, the following specification adds a *ram-entry* for the covariance parameter between v2 and v3:

```
RAM
   var= v1 v2 v3,
   _A_   1   2  parm1(0.5),
   _A_   1   3  1.0,
   _P_   2   2  parm2,
   _P_   1   1,
   _P_   2   3   (.5);
```

The covariance between v2 and v3 is an unnamed parameter with an initial value of 0.5.

*(4) Specification of Means and Intercepts in Model Matrix _W_*

To specifying means or intercepts, you need to start the *ram-entries* with the _W_ keyword. For example, the last two entries of following statement specify the intercept of v1 and the mean of v2, respectively:

```
RAM
   var= v1 v2 v3,
   _A_   1   2  parm1(0.5),
   _A_   1   3  1.0,
   _P_   2   2  parm2,
   _P_   1   1 ,
   _P_   2   3  (.5),
   _W_   1   1  int_v1,
   _W_   2   2  mean_v2;
```

The intercept of v1 is a free parameter named int_v1, and the mean of v2 is a free parameter named mean_v2.

Default Parameters in the RAM Model

There are two types of default of parameters of the RAM model in PROC CALIS. One is the free parameters; the other is the fixed zeros.

By default, certain sets of model matrix elements in the RAM model are free parameters. These parameters are set automatically by PROC CALIS, although you can also specify them explicitly in the *ram-entries*. In general, default free parameters enable you to specify only what are absolutely necessary for defining your model. PROC CALIS automatically sets those commonly assumed free parameters so that you do not need to specify them routinely. The sets of default free parameters of the RAM model are as follows:

- Diagonal elements of the _P_ matrix—this includes the variance of exogenous variables (latent or observed) and error variances of all endogenous variables (latent or observed)
- The off-diagonal elements that pertain to the exogenous variables of the _P_ matrix—this includes all the covariances among exogenous variables, latent or observed
- If the mean structures are modeled, the elements that pertain to the observed variables (but not the latent variables) in the _W_ vector— this includes all the means of exogenous observed variables and the intercepts of all endogenous observed variables

For example, suppose you are fitting a RAM model with three observed variables x1, x2, and y3, you specify a simple multiple-regression model with x1 and x2 predicting y3 by the following statements:

```
proc calis meanstr;
   ram  var= x1-x2 y3,
        _A_ 3 1 ,
        _A_ 3 2 ;
```

In the RAM statement, you specify that path coefficients represented by **_A_[3,1]** and **_A_[3,2]** are free parameters in the model. In addition to these free parameters, PROC CALIS sets several other free parameters by default. **_P_[1,1]**, **_P_[2,2]**, and **_P_[3,3]** are set as free parameters for the variance of x1, the variance of x2, and the error variance of x3, respectively. **_P_[2,1]** (and hence **_P_[1,2]**) is set as a free parameter for the covariance between the exogenous variables x1 and x2. Because the mean structures are also analyzed by the MEANSTR option in the PROC CALIS statement, **_W_[1,1]**, **_W_[2,1]**, and **_W_[3,1]** are also set as free parameters for the mean of x1, the mean of x2, and the intercept of x3, respectively. In the current situation, this default parameterization is consistent with using PROC REG for multiple regression analysis, where you only need to specify the functional relationships among variables.

If a matrix element is not a default free parameter in the RAM model, then it is a fixed zero by default. You can override almost all default fixed zeros in the RAM model matrices by specifying the *ram-entries*. The diagonal elements of the _A_ matrix are exceptions. These elements are always fixed zeros. You cannot set these elements to free parameters or other fixed values—this reflects a model restriction that prevents a variable from having a direct effect on itself.

Naming Variables and Parameters

Follow these rules when you name your variables:

- Use the usual naming conventions of the SAS System.
- Variable names are not more than 32 characters.
- When you create latent variable names, make sure that they are not used in the input data set that is being analyzed.
- For the LINEQS model, error or disturbance variables must start with ‘E’, ‘e’, ‘D’, or ‘d’.
- For the LINEQS model, non-error-type latent variables (that is, factors) must start with ‘F’ or ‘f’.
- For modeling languages other than LINEQS, names for errors or disturbances are not needed. As a result, you do not need to distinguish latent factors from errors or disturbances by using particular prefixes. Variable names that are not referenced in the analyzed data set are supposed to be latent factors.
- You should not use Intercept (case-insensitive) as a variable name in your data set or as a latent variable name in your model.

Follow these rules when you name your parameters:

- Use the usual naming conventions of the SAS System.
- Parameter names are not more than 32 characters.
- Use a prefix-name when you want to generate new parameter names automatically. A prefix-name contains a short string of characters called a “root,” followed by double underscores ‘__’. Each occurrence of such a prefix-name generates a new parameter name by replacing the two trailing underscores with a unique integer. For example, occurrences of Gen__ generate Gen1, Gen2, and so on.
- A special prefix-name is the one without a root—that is, it contains only double underscores ‘__’. Occurrences of ‘__’ generate _Parm1, _Parm2, and so on.

- PROC CALIS generates parameter names for default parameters to safeguard ill-defined models. These generated parameter names start with the _Add prefix and unique integer suffixes. For example, _Add1, _Add2, and so on.

- Avoid using parameter names that start with either _, _Add, or _Parm. These names might get confused with the names generated by PROC CALIS. The confusion might lead to unintended constraints to your model if the parameter names that you use match those generated by PROC CALIS.

- Avoid using parameter names that are roots of prefix-names. For example, you should not use Gen as a parameter name if Gen__ is also used in the same model specification. Although violation of this rule might not distort the model specification, it might cause ambiguities and confusion.

Finally, parameter names and variable names in PROC CALIS are not distinguished by explicit declarations. That is, a valid SAS name can be used as a parameter name or a variable name in any model that is supported by PROC CALIS. Whether a name in a model specification is for a parameter or a variable is determined by the syntactic structure. For example, consider the following path specification:

```
proc calis;
   path
      a ---> b    = c;
   run;
```

PROC CALIS parses the path specification according to the syntactic structure of the PATH statement and determines that a and b are variable names and c is a parameter name. Consider another specification as follows:

```
proc calis;
   path
      a ---> b    = b;
   run;
```

This is a syntactically correct specification. Variables a and b are defined in a path relationship with a path coefficient parameter also named b. While such a name conflict between parameter and variable names would not confuse PROC CALIS in terms of model specification and fitting, it would create unnecessary confusion in programming and result interpretations. Hence, using parameter names that match variable names exactly is a bad practice and should be avoided.

Setting Constraints on Parameters

The CALIS procedure offers a very flexible way to constrain parameters. There are two main methods for specifying constraints. One is explicit specification by using specially designed statements for constraints. The other is implicit specification by using the SAS programming statements.

Explicit Specification of Constraints

Explicit constraints can be set in the following ways:

- specifying boundary constraints on independent parameters in the BOUNDS statement
- specifying general linear equality and inequality constraints on independent parameters in the LINCON statement
- specifying general nonlinear equality and inequality constraints on parametric functions in the NLINCON statement

BOUNDS Statement

You can specify one-sided or two-sided boundaries on independent parameters in the BOUNDS statement. For example, in the following statement you constrain parameter var1 to be nonnegative and parameter effect to be between 0 and 5.

```
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 CALIS will reduce the number of independent parameters by one automatically. Note also that only independent parameters are allowed to be bounded in the BOUNDS statement.

LINCON 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 CALIS 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 CALIS procedure. PROC CALIS is able to compute analytic first- and second-order derivatives that you would have to specify using the COSAN program.

Some traditional ways to enforce parameter constraints by using reparameterization or parameter transformation (McDonald 1980) are considered in the following:

- **one-sided boundary constraints of the form:**

  $$q \geq a \qquad \text{or} \qquad q \leq b$$

  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 CALIS, 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:**

  $$q_1 \leq q_2, \quad q_1 \leq q_3, \quad \ldots, \quad q_1 \leq q_k$$

  where $q_1, \ldots, q_k$ are the parameters of interest. These inequality constraints can be expressed as the following set of equality constraints:

  $$q_1 = x_1, \quad q_2 = x_1 + x_2^2, \quad \ldots, \quad q_k = x_1 + x_k^2$$

  where the new parameters $x_1, \ldots, x_k$ are unconstrained.

For example, to implement $q_1 \leq q_2$, $q_1 \leq q_3$, and $q_1 \leq q_4$ simultaneously, you can use the following statements:

```
parameters x1-x4 (4*0.);
q1 = x1;
q2 = x1 + x2 * x2;
q3 = x1 + x3 * x3;
q4 = x1 + x4 * x4;
```

In this specification, you essentially redefine q_1–q_4 as dependent parameters that are functions of x_1–x_4, which are defined as independent parameters in the PARAMETERS statement with starting values of zeros. No constraints on x_i's are needed. The way that q_i's are computed in the SAS programming statements guarantees the required order constraints on q_i's are satisfied.

- **two-sided order constraints of the form:**

  $$q_1 \leq q_2 \leq q_3 \leq \ldots \leq q_k$$

  These inequality constraints can be expressed as the following set of equality constraints:

  $$q_1 = x_1, \quad q_2 = q_1 + x_2^2, \quad \ldots \quad q_k = q_{k-1} + x_k^2$$

  where the new parameters $x_1, \ldots, x_k$ are unconstrained.

  For example, to implement $q_1 \leq q_2 \leq q_3 \leq q_4$ simultaneously, you can use the following statements:

```
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 CALIS checks the variables mentioned in these statements against the variable list of the input data set. If a variable in your model is also found in your data set, PROC CALIS knows that it is a manifest variable. Otherwise, it is either a latent variable or an invalid variable.

To save computational resources, only manifest variables defined in your model are automatically selected for analysis. For example, even if you have 100 variables in your input data set, only a covariance matrix of 10 manifest variables is computed for the analysis of the model if only 10 variables are selected for analysis.

In some special circumstances, the automatic variable selection performed for the analysis might be a drawback. For example, if you are interested in modification indices connected to some of the variables that are not used in the model, automatic variable selection in the specification stage will exclude those variables from consideration in computing modification indices. Fortunately, a little trick can be done. You can use the VAR statement to include as many exogenous manifest variables as needed. Any variables in the VAR statement that are defined in the input data set but are not used in the main and subsidiary model specification statements are included in the model as exogenous manifest variables.

For example, the first three steps in a stepwise regression analysis of the Werner Blood Chemistry data (Jöreskog and Sörbom 1988, p. 111) can be performed as follows:

```
proc calis data=dixon method=gls nobs=180 print mod;
   var    x1-x7;
   lineqs y = e;
   variance    e = var;
run;
proc calis data=dixon method=gls nobs=180 print mod;
   var    x1-x7;
   lineqs y = g1 x1 + e;
   variance    e = var;
run;
proc calis data=dixon method=gls nobs=180 print mod;
   var    x1-x7;
   lineqs y = g1 x1 + g6 x6 + e;
   variance    e = var;
run;
```

In the first analysis, no independent manifest variables are included in the regression equation for dependent variable y. However, 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 six estimation methods are available in PROC CALIS:

- unweighted least squares (ULS)
- full information maximum likelihood (FIML)
- generalized least squares (GLS)
- normal-theory maximum likelihood (ML)
- weighted least squares (WLS, ADF)
- diagonally weighted least squares (DWLS)

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 CALIS does not implement all estimation methods in the field. As mentioned in the section "Overview: CALIS Procedure" on page 986, partial least squares (PLS) is not implemented. The PLS method is developed under less restrictive statistical assumptions. It circumvents some computational and theoretical problems encountered by the existing estimation methods in PROC CALIS; however, PLS estimates are less efficient in general. When the statistical assumptions of PROC CALIS are tenable (for example, large sample size, correct distributional assumptions, and so on), ML, GLS, or WLS methods yield better estimates than the PLS method. Note that there is a SAS/STAT procedure called PROC PLS that employs the partial least squares technique, but for a different class of models than those of PROC CALIS. For example, in a PROC CALIS model each latent variable is typically associated with only a subset of manifest variables (predictor or outcome variables). However, in PROC PLS latent variables are not prescribed with subsets of manifest variables. Rather, they are extracted from linear combinations of all manifest predictor variables. Therefore, for general path analysis with latent variables you should use PROC CALIS.

ULS, GLS, and ML Discrepancy Functions

In each estimation method, the parameter vector is estimated iteratively by a nonlinear optimization algorithm that minimizes a discrepancy function F, which is also known as the fit function in the literature. With p denoting the number of manifest variables, $\mathbf{S}$ 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 CALIS are straightforward generalizations of those of covariance analysis. In very limited cases these standard errors are good approximations. However, in general they are not even asymptotically correct.
- The model fit chi-square statistic for correlation analysis might not follow the theoretical distribution, thus making model fit testing difficult.

Despite these issues in correlation analysis, if your primary interest is to obtain the estimates in the correlation models, you might still find PROC CALIS results for correlation analysis useful.

The statistical techniques used in PROC CALIS are primarily developed for the analysis of covariance structures, and hence COVARIANCE is the default option. Depending on the nature of your research, you can add the mean structures in the analysis by specifying mean and intercept parameters in your models. However, you cannot analyze mean structures simultaneously with correlation structures (see the CORRELATION option) in PROC CALIS.

FIML Discrepancy Function

The full information maximum likelihood method (FIML) assumes multivariate normality of the data. Suppose that you analyze a model that contains p observed variables. The discrepancy function for FIML is

$$F_{FIML} = \frac{1}{N}\sum_{j=1}^{N}(ln(|\boldsymbol{\Sigma}_j|) + (\boldsymbol{x}_j - \boldsymbol{\mu}_j)'\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_j - \boldsymbol{\mu}_j) + K_j)$$

where $\boldsymbol{x}_j$ is a data vector for observation j, and K_j is a constant term (to be defined explicitly later) independent of the model parameters $\boldsymbol{\Theta}$. In the current formulation, $\boldsymbol{x}_j$'s are not required to have the same dimensions. For example, $\boldsymbol{x}_1$ could be a complete vector with all p variables present while $\boldsymbol{x}_2$ is a $(p-1)\times 1$ vector with one missing value that has been excluded from the original $p \times 1$ data vector. As a consequence, subscript j is also used in $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ to denote the submatrices that are extracted from the entire $p \times 1$ structured mean vector $\boldsymbol{\mu}$ ($\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{\Theta})$) and $p \times p$ covariance matrix $\boldsymbol{\Sigma}$ ($\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(\boldsymbol{\Theta})$). In other words, in the current formulation $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ do not mean that each observation is fitted by distinct mean and covariance structures (although theoretically it is possible to formulate FIML in such a way). The notation simply signifies that the dimensions of $\boldsymbol{x}_j$ and of the associated mean and covariance structures could vary from observation to observation.

Let p_j be the number of variables without missing values for observation j. Then $\boldsymbol{x}_j$ denotes a $p_j \times 1$ data vector, $\boldsymbol{\mu}_j$ denotes a $p_j \times 1$ vector of means (structured with model parameters), $\boldsymbol{\Sigma}_j$ is a $p_j \times p_j$ matrix for variances and covariances (also structured with model parameters), and K_j is defined by the following formula, which is a constant term independent of model parameters:

$$K_j = ln(2\pi) * p_j$$

As a general estimation method, the FIML method is based on the same statistical principle as the ordinary maximum likelihood (ML) method for multivariate normal data—that is, both methods maximize the normal theory likelihood function given the data. In fact, F_{FIML} used in PROC CALIS is related to the log-likelihood function L by the following formula:

$$F_{FIML} = \frac{-2L}{N}$$

Because the FIML method can deal with observations with various levels of information available, it is primarily developed as an estimation method that could deal with data with random missing values. See the section "Relationships among Estimation Criteria" on page 1252 for more details about the relationship between FIML and ML methods.

Whenever you use the FIML method, the mean structures are automatically assumed in the analysis. This is due to fact that there is no closed-form formula to obtain the saturated mean vector in the FIML discrepancy function if missing values are present in the data. You can certainly provide explicit specification of the mean parameters in the model by specifying intercepts in the LINEQS statement or means and intercepts in the MEAN or MATRIX statement. However, usually you do not need to do the explicit specification if all you need to achieve is to saturate the mean structures with p parameters (that is, the same number as the number of observed variables in the model). With METHOD=FIML, PROC CALIS uses certain default parameterizations for the mean structures automatically. For example, all intercepts of endogenous observed variables and all means of exogenous observed variables are default parameters in the model, making the explicit specification of these mean structure parameters unnecessary.

WLS and ADF Discrepancy Functions

Another important discrepancy function to consider is the weighted least squares (WLS) function. Let $u = (s, \bar{\mathbf{x}})$ be a $p(p+3)/2$ vector containing all nonredundant elements in the sample covariance matrix $\mathbf{S}$ and sample mean vector $\bar{\mathbf{x}}$, 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 $\eta = (\sigma, \boldsymbol{\mu})$ be a $p(p+3)/2$ vector containing all nonredundant elements in the fitted covariance matrix $\boldsymbol{\Sigma}$ and the fitted mean vector $\boldsymbol{\mu}$, with $\sigma = vecs(\boldsymbol{\Sigma})$ representing the vector of the $p(p+1)/2$ lower triangle elements of the symmetric matrix $\boldsymbol{\Sigma}$.

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 $\boldsymbol{\Theta}$ under the structural model, you can write the WLS function as:

$$F_{WLS} = (u - \eta(\boldsymbol{\Theta}))' \mathbf{W}^{-1} (u - \eta(\boldsymbol{\Theta}))$$

Suppose that u converges to $\eta_o = (\sigma_o, \boldsymbol{\mu}_o)$ with increasing sample size, where σ_o and $\boldsymbol{\mu}_o$ denote the population covariance matrix and mean vector, respectively. By default, the WLS weight matrix $\mathbf{W}$ in PROC CALIS is computed from the raw data as a consistent estimate of the asymptotic covariance matrix $\boldsymbol{\Gamma}$ of $\sqrt{N}(u - \eta_o)$, with $\boldsymbol{\Gamma}$ partitioned as

$$\Gamma = \begin{pmatrix} \boldsymbol{\Gamma}_{ss} & \boldsymbol{\Gamma}'_{\bar{x}s} \\ \boldsymbol{\Gamma}_{\bar{x}s} & \boldsymbol{\Gamma}_{\bar{x}\bar{x}} \end{pmatrix}$$

where $\boldsymbol{\Gamma}_{ss}$ denotes the $(p(p+1)/2) \times (p(p+1)/2)$ asymptotic covariance matrix for $\sqrt{N}(s - \sigma_o)$, $\boldsymbol{\Gamma}_{\bar{x}\bar{x}}$ denotes the $p \times p$ asymptotic covariance matrix for $\sqrt{N}(\bar{x} - \boldsymbol{\mu}_o)$, and $\boldsymbol{\Gamma}_{\bar{x}s}$ denotes the $p \times (p(p+1)/2)$ asymptotic covariance matrix between $\sqrt{N}(\bar{x} - \boldsymbol{\mu}_o)$ and $\sqrt{N}(s - \sigma_o)$.

To compute the default weight matrix $\mathbf{W}$ as a consistent estimate of $\boldsymbol{\Gamma}$, 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 $\mathbf{W}$ is a consistent but biased estimator of the asymptotic covariance matrix $\mathbf{\Gamma}$.

By using the ASYCOV= option, you can use Browne's unbiased estimator (Browne 1984, formula (3.8)) 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 $\mathbf{W}$ 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 $\mathbf{W}$ computed by PROC CALIS, the WLS estimation is also called as the asymptotically distribution-free (ADF) method. In fact, as options in PROC CALIS, METHOD=WLS and METHOD=ADF are totally equivalent, even though WLS in general might include cases with special weight matrices other than the default weight matrix.

When the mean structures are not modeled, the WLS discrepancy function is still the same quadratic form statistic. However, with only the elements in covariance matrix being modeled, the dimensions of u and 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 $\mathbf{S}$ 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:

$$\begin{aligned} F_{WLS} &= \sum_{i=2}^{n}\sum_{j=1}^{i-1}\sum_{k=2}^{n}\sum_{l=1}^{k-1}[\mathbf{W}_{ss}]^{ij,kl}([\mathbf{S}]_{ij} - [\boldsymbol{\Sigma}]_{ij})([\mathbf{S}]_{kl} - [\boldsymbol{\Sigma}]_{kl}) \\ & \quad + r\sum_{i}^{n}([\mathbf{S}]_{ii} - [\boldsymbol{\Sigma}]_{ii})^2 \end{aligned}$$

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 $\mathbf{S}$. 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 $\mathbf{W}$ 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 $\mathbf{W}$ from the WLS estimation in the following discrepancy function:

$$\begin{aligned} F_{DWLS} &= (u - \eta)'[\mathrm{diag}(\mathbf{W})]^{-1}(u - \eta) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{i}[\mathbf{W}_{ss}]^{-1}_{ij,ij}([\mathbf{S}]_{ij} - [\boldsymbol{\Sigma}]_{ij})^2 + \sum_{i=1}^{n}[\mathbf{W}_{\bar{x}\bar{x}}]^{-1}_{ii}(\bar{x}_i - \boldsymbol{\mu}_i)^2 \end{aligned}$$

When only the covariance structures are modeled, the discrepancy function becomes:

$$F_{DWLS} = \sum_{i=1}^{n}\sum_{j=1}^{i}[\mathbf{W}_{ss}]^{-1}_{ij,ij}([\mathbf{S}]_{ij} - [\boldsymbol{\Sigma}]_{ij})^2$$

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 **W** by using an INWGT= data set. The CALIS procedure requires a positive definite weight matrix that has positive diagonal elements.

Multiple-Group Discrepancy Function

Suppose that there are k independent groups in the analysis and $N_1, N_2, \ldots, N_k$ are the sample sizes for the groups. The overall discrepancy function $F(\mathbf{\Theta})$ 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 CALIS, all discrepancy function F_i's in the overall discrepancy function must belong to the same estimation method. You cannot specify different estimation methods for the groups in a multiple-group analysis. In addition, the same analysis type must be applied to all groups—that is, you can analyze either covariance structures, covariance and mean structures, and correlation structures for all groups.

Relationships among Estimation Criteria

There is always some arbitrariness to classify the estimation methods according to certain mathematical or numerical properties. The discussion in this section is not meant to be a thorough classification of the estimation methods available in PROC CALIS. Rather, classification is done here with the purpose of clarifying the uses of different estimation methods and the theoretical relationships of estimation criteria.

Assumption of Multivariate Normality

GLS, ML, and FIML assume multivariate normality of the data, while ULS, WLS, and DWLS do not. Although the ML method with covariance structure analysis alone can also be based on the Wishart distribution of the sample covariance matrix, for convenience GLS, ML, and FIML are usually classified as normal-theory based methods, while ULS, WLS, and DWLS are usually classified as distribution-free methods.

An intuitive or even naive notion is usually that methods without distributional assumptions such as WLS and DWLS are preferred to normal theory methods such as ML and GLS in practical situations where multivariate normality is doubt. This notion might need some qualifications because there are simply more factors to consider in judging the quality of estimation methods in practice. For example, the WLS method might need a very large sample size to enjoy its purported asymptotic properties, while the ML might be robust against the violation of multi-normality assumption under certain circumstances. No recommendations regarding which estimation criterion should be used are attempted here, but you should make your choice based more than the assumption of multivariate normality.

Contribution of the Off-Diagonal Elements to the Estimation of Covariance or Correlation Structures

If only the covariance or correlation structures are considered, the six estimation functions, F_{ULS}, F_{GLS}, F_{ML}, F_{FIML}, F_{WLS}, and F_{DWLS}, belong to the following two groups:

- The functions F_{ULS}, F_{GLS}, F_{ML}, and F_{FIML} take into account all n^2 elements of the symmetric residual matrix $\mathbf{S} - \mathbf{\Sigma}$. 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 CALIS differs from that used by the LISREL 7 program. Formula (1.25) of the LISREL 7 manual (Jöreskog and Sörbom 1985, p. 23) shows that LISREL groups the F_{DWLS} function in the first group by taking into account all n^2 elements of the symmetric residual matrix $\mathbf{S} - \mathbf{\Sigma}$.

- Relationship between DWLS and WLS:
  PROC CALIS: The F_{DWLS} and F_{WLS} discrepancy functions deliver the same results for the special case that the weight matrix $\mathbf{W} = \mathbf{W}_{ss}$ 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 CALIS: 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.

ML and FIML Methods

Both the ML and FIML methods can be derived from the log-likelihood function for multivariate normal data. The preceding section "Estimation Criteria" on page 1246 mentions that F_{FIML} is essentially the same as $\frac{-2L}{N}$, where L is the log-likelihood function for multivariate normal data. For the ML estimation, you can also consider $\frac{-2L}{N}$ as a part of the F_{ML} discrepancy function that contains the information regarding the model parameters (while the rest the F_{ML} function contains some constant terms given the data). That is, with some algebraic manipulations and assuming that there is no missing value in the analysis (so that all $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ are the same as $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, respectively), it can shown that

$$\begin{aligned} F_{FIML} &= \frac{-2L}{N} \\ &= \frac{1}{N}\sum_{j=1}^{n}(ln(|\boldsymbol{\Sigma}|) + (\boldsymbol{x}_j - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_j - \boldsymbol{\mu}) + K) \\ &= ln(|\boldsymbol{\Sigma}|) + Tr(\mathbf{S_N}\boldsymbol{\Sigma}^{-1}) + (\bar{\boldsymbol{x}} - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\bar{\boldsymbol{x}} - \boldsymbol{\mu}) + K \end{aligned}$$

where $\bar{\boldsymbol{x}}$ is the sample mean and $\mathbf{S_N}$ is the biased sample covariance matrix. Compare this FIML function with the ML function shown in the following expression, which shows that both functions are very similar:

$$F_{ML} = ln(|\boldsymbol{\Sigma}|) + Tr(\mathbf{S}\boldsymbol{\Sigma}^{-1}) + (\bar{\boldsymbol{x}} - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\bar{\boldsymbol{x}} - \boldsymbol{\mu}) - p - ln(|\mathbf{S}|)$$

The two expressions differ only in the constant terms, which are independent of the model parameters, and in the formulas for computing the sample covariance matrix. While the FIML method assumes the biased formula (with N as the divisor, by default) for the sample covariance matrix, the ML method (as implemented in PROC CALIS) uses the unbiased formula (with $N - 1$ as the divisor, by default).

The similarity (or dissimilarity) of the ML and FIML discrepancy functions leads to some useful conclusions here:

- Because the constant terms in the discrepancy functions play no part in parameter estimation (except for shifting the function values), overriding the default ML method with VARDEF=N (that is, using N as the divisor in the covariance matrix formula) leads to the same estimation results as that of the FIML method, given that there are no missing values in the analysis.
- Because the FIML function is evaluated at the level of individual observations, it is much more expensive to compute than the ML function. As compared with ML estimation, FIML estimation takes longer and uses more computing resources. Hence, for data without missing values, the ML method should always be chosen over the FIML method.
- The advantage of the FIML method lies solely in its ability to handle data with random missing values. While the FIML method uses the information maximally from each observation, the ML method (as implemented in PROC CALIS) simply throws away any observations with at least one missing value. If it is important to use the information from observations with random missing values, the FIML method should be given consideration over the ML method.

See Example 26.14 for an application of the FIML method and Example 26.15 for an empirical comparison of the ML and FIML methods.

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 $\mathbf{S}, \bar{\boldsymbol{x}}$.

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 CALIS output.

Taking the inverse of the empirical information matrix with sample size adjustment, PROC CALIS 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 CALIS offers two ways to compute a generalized inverse of the matrix and, therefore, two ways to compute approximate standard errors of implicitly constrained parameter estimates, t values, and modification indices. Depending on the G4= specification, either a Moore-Penrose inverse or a G2 inverse is computed. The expensive Moore-Penrose inverse computes an estimate of the null space by using an eigenvalue decomposition. The cheaper G2 inverse is produced by sweeping the linearly independent rows and columns and zeroing out the dependent ones.

Multiple-Group Extensions

In the section "Multiple-Group Discrepancy Function" on page 1252, 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 CALIS results.

Taking the inverse of the empirical information matrix with sample size adjustment, PROC CALIS 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 CALIS did not set the right subset of eigenvalues to zero, you can specify the COVSING= option to set a larger or smaller subset of eigenvalues to zero in a further run of PROC CALIS.

Counting the Degrees of Freedom

When fitting covariance and mean structure models, the population moments are hypothesized to be functions of model parameters $\mathbf{\Theta}$. The population moments refer to the first-order moments (means) and the second-order central moments (variances of and covariances among the variables). Usually, the number of nonredundant population moments is greater than the number of model parameters for a structural model. The difference between the two is the degrees of freedom (df) of your model.

Formally, define a multiple-group situation where you have k independent groups in your model. The set of variables in each group might be different so that you have $p_1, p_2, \ldots, p_k$ manifest or observed variables for the k groups. It is assumed that the primary interest is to study the covariance structures. The inclusion of mean structures is optional for each of these groups. Define $\delta_1, \delta_2, \cdots, \delta_k$ 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 PARAMETERS statement, or it is generated automatically by PROC CALIS as additional parameters. Quite intuitively, all independent parameter specified in the main or subsidiary model specification statements are independent parameters *in the model*. All automatic parameters added by PROC CALIS are also independent parameters *in the model*.

Intentionally or not, some independent parameters specified in the PARMS statement might not be counted as independent parameters in the model. Independent parameters in the PARMS statement belong in the model only when they are used to define at least one dependent parameter specified in the main or subsidiary model specification statements. This restriction eliminates the counting of superfluous independent parameters which have no bearing of model specification.

Note that when counting the number of independent parameters, you are counting the number of distinct independent parameter names but not the number of distinct parameter locations for independent parameters. For example, consider the following statement for defining the error variances in a LINEQS model:

```
variance    E1-E3 = vare1 vare2 vare3;
```

You define three variance parameter locations with three independent parameters vare1, vare2, and vare3. However, in the following specification:

```
variance    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 CALIS will automatically increase the degrees of freedom by the number of active linear constraints. Adjusting degrees of freedom not only affects the significance of the model fit chi-square statistic, but it also affects the computation of many fit statistics and indices. 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 CALIS statement.

Another case where you need to pay attention to the computation of degrees of freedom is when you fit correlation models. The degrees-of-freedom calculation in PROC CALIS applies mainly to models with covariance structures with or without mean structures. When you model correlation structures, the degrees of freedom calculation in PROC CALIS is a straightforward generalization of the covariance structures. It does not take the fixed ones at the diagonal elements of the sample correlation matrix into account. Some might argue that with correlation structures, the degrees of freedom should be reduced by the total number of diagonal elements in the correlation matrices in the model. While PROC CALIS does not do this automatically, you can use the DFREDUCE=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 CALIS to specify the effective number of observations in the input data set only.

Assessment of Fit

In PROC CALIS, there are three main tools for assessing model fit:

- residuals for the fitted means or covariances
- overall model fit indices
- squared multiple correlations and determination coefficients

This section contains a collection of formulas for these assessment tools. The following notation is used:

- N for the total sample size
- k for the total number of independent groups in analysis
- p for the number of manifest variables
- t for the number of parameters to estimate
- $\mathbf{\Theta}$ 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
- $\bar{\boldsymbol{x}} = (\bar{x}_i)$ for the p-vector of sample means
- $\hat{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}(\hat{\boldsymbol{\Theta}}) = (\hat{\sigma}_{ij})$ 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, $\mathbf{S}_r$, $\bar{\boldsymbol{x}}_r$, $\hat{\boldsymbol{\Sigma}}_r$, $\hat{\boldsymbol{\mu}}_r$, 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 CALIS 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 greater than the asymptotically standardized residuals (which use the diagonal of the form $\mathbf{\Gamma} - \mathbf{J}\hat{\mathrm{Cov}}(\hat{\mathbf{\Theta}})\mathbf{J}'$).

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 CALIS produces a large number of overall model fit indices in the fit summary table. If you prefer to display only a subset of these fit indices, you can use the ONLIST(ONLY)= option of the FITINDEX statement to customize the fit summary table.

Fit indices are classified into three classes in the fit summary table of PROC CALIS:

- absolute or standalone Indices
- parsimony indices
- incremental indices

Absolute or Standalone Indices

These indices are constructed so that they measure model fit without comparing with a baseline model and without taking the model complexity into account. They measure the absolute fit of the model.

- **fit function or discrepancy function**
  The fit function or discrepancy function F is minimized during the optimization. See the section "Estimation Criteria" on page 1246 for definitions of various discrepancy functions available in PROC CALIS. For a multiple-group analysis, the fit function can be written as a weighted average of discrepancy functions for k independent groups as:

  $$F = \sum_{r=1}^{k} a_r F_r$$

  where $a_r = \frac{(N_j - 1)}{(N - k)}$ 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 unstandardized 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 CALIS, 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 (1975, p. 527) for an application of the Z-test.

- **critical N index (Hoelter 1983)**
  The critical N (Hoelter 1983) is defined as:

$$\mathrm{CN} = \mathrm{int}(\frac{\chi^2_{crit}}{f_{min}})$$

  where χ^2_{crit} is the critical chi-square value for the given d degrees of freedom and probability $\alpha = 0.05$, and int() takes the integer part of the expression. Refer to Bollen (1989b, p. 277). Conceptually, the CN value is the largest number of observations that could still make the chi-square model fit statistic insignificant if it were to apply to the actual sample fit function value f_{min}. Hoelter (1983) suggests that CN should be at least 200; however, Bollen (1989b) notes that the CN value might lead to an overly pessimistic assessment of fit for small samples.

  Note that when you have a perfect model fit for your data (that is, $f_{min} = 0$) or a zero degree of freedom for your model (that is, $d = 0$), CN is not computable.

- **root mean square residual (RMR)**
  For a single-group analysis, the RMR is the root of the mean of the squared residuals:

$$\mathrm{RMR} = \sqrt{\frac{2}{p(p+1+2\delta)}[\sum_i^p \sum_j^i (s_{ij} - \hat{\sigma}_{ij})^2 + \delta \sum_i^p (\bar{x}_i - \hat{\mu}_i)^2]}$$

  For multiple-group analysis, PROC CALIS 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,

$$\text{overall RMR} = \sqrt{\sum_{r=1}^{k} a_r \mathrm{RMR}_r^2}$$

- **standardized root mean square residual (SRMR)**
  For a single-group analysis, the SRMR is the root of the mean of the squared standardized residuals:

$$\mathrm{SRMR} = \sqrt{\frac{2}{p(p+1+2\delta)}[\sum_i^p \sum_j^i \frac{(s_{ij} - \hat{\sigma}_{ij})^2}{s_{ii}s_{jj}} + \delta \sum_i^p \frac{(\bar{x}_i - \hat{\mu}_i)^2}{s_{ii}}]}$$

  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 $a_r = \frac{N_r - 1}{N-k}$

$$\text{overall SRMR} = \sqrt{\sum_{r=1}^{k} a_r \mathrm{SRMR}_r^2}$$

- **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,

$$u = (\text{vecs}'(\mathbf{S}), \bar{\mathbf{x}}')', \quad \hat{\eta} = (\text{vecs}'(\hat{\mathbf{\Sigma}}), \hat{\boldsymbol{\mu}}')'$$

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,

$$u = \text{vecs}(\mathbf{S}), \quad \hat{\eta} = \text{vecs}(\hat{\mathbf{\Sigma}})$$

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 greater than 1.

For a multiple-group analysis, individual GFI_r's are computed for groups. The overall measure is a weighted average of individual GFI_r's, using weight $a_r = \frac{N_r - 1}{N - k}$. That is,

$$\text{overall GFI} = \sum_{r=1}^{k} a_r \text{GFI}_r$$

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 greater 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 1269 for the definition of independence model. The PGFI uses the same parsimonious factor as the parsimonious normed Bentler-Bonett index (James, Mulaik, and Brett 1982).

- **RMSEA index (Steiger and Lind 1980; Steiger 1998)**
  The root mean square error approximation (RMSEA) coefficient is:

  $$\epsilon = \sqrt{k}\sqrt{\max\left(\frac{f_{min}}{d_{min}} - \frac{1}{(N-k)}, 0\right)}$$

  The lower and upper limits of the $(1-\alpha)\%$-confidence interval are computed using the cumulative distribution function of the noncentral chi-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 $\Phi(x|\lambda_U, d_{min}) = \frac{\alpha}{2}$:

  $$(\epsilon_{\alpha_L}; \epsilon_{\alpha_U}) = \left(\sqrt{k}\sqrt{\frac{\lambda_L}{(N-k)d_{min}}}; \sqrt{k}\sqrt{\frac{\lambda_U}{(N-k)d_{min}}}\right)$$

  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: $\epsilon = 0$ is replaced by the null hypothesis of close fit H_0: $\epsilon \le 0.05$ 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 CALIS. For GLS and WLS, the estimator c of the ECVI is linearly related to AIC, Akaike's Information Criterion (Akaike 1974, 1987):

$$c = f_{min} + \frac{2t}{N-1}$$

For ML estimation, c_{ML} is used:

$$c_{ML} = f_{min} + \frac{2t}{N-p-2}$$

For GLS and WLS, the confidence interval $(c_L; c_U)$ for ECVI is computed using the cumulative distribution function $\Phi(x|\lambda, d_{min})$ of the noncentral chi-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.

$$\text{AIC} = h + 2t$$

where h is the -2 times the likelihood function value for the FIML method or the χ^2 value for other estimation methods.

- **consistent Akaike's information criterion (CAIC) (Bozdogan 1987)**
This is another criterion, similar to AIC, for selecting the best model among alternatives. The model that yields the smallest value of CAIC is considered the best. CAIC is preferred by some people to AIC or the χ^2 test.

$$\text{CAIC} = h + (ln(N) + 1)t$$

where h is the -2 times the likelihood function value for the FIML method or the χ^2 value for other estimation methods. Notice that N includes the number of incomplete observations for the FIML method while it includes only the complete observations for other estimation methods.

- **Schwarz's Bayesian criterion (SBC) (Schwarz 1978; Sclove 1987)**
This is another criterion, similar to AIC, for selecting the best model. The model that yields the smallest value of SBC is considered the best. SBC is preferred by some people to AIC or the χ^2 test.

$$\text{SBC} = h + ln(N)t$$

where h is the -2 times the likelihood function value for the FIML method or the χ^2 value for other estimation methods. Notice that N includes the number of incomplete observations for the FIML method while it includes only the complete observations for other estimation methods.

- **McDonald's measure of centrality (McDonald and Marsh 1988)**

$$\text{CENT} = \exp(-\frac{(\chi^2 - d_{min})}{2N})$$

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)**

$$\text{CFI} = 1 - \frac{\max((N-k)f_{min} - d_{min}, 0)}{\max((N-k)f_{min} - d_{min}, \max((N-k)f_0 - d_0, 0)}$$

- **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 the IFI2 index of Mulaik et al. (1989).

- **parsimonious normed fit index (James, Mulaik, and Brett 1982)**
  The PNFI is a modification of Bentler-Bonett's normed fit index that takes parsimony of the model into account,

  $$\text{PNFI} = \frac{d_{min}}{d_0} \frac{(f_0 - f_{min})}{f_0}$$

  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 CALIS statement. The availability of fit indices is summarized as follows:

- Adjusted (elliptic) chi-square and its probability are available only for METHOD=ML or GLS and with the presence of raw data input.
- For METHOD=ULS or DWLS, probability of the chi-square value, RMSEA and its confidence intervals, probability of close fit, ECVI and its confidence intervals, critical N index, Z-test, AIC, CAIC, SBC, and measure of centrality are not appropriate and therefore not displayed.

Individual Fit Indices for Multiple Groups

When you compare the fits of individual groups in a multiple-group analysis, you can examine the residuals of the groups to gauge which group is fitted better than the others. While examining residuals is good for knowing specific locations with inadequate fit, summary measures like fit indices for individual groups would be more convenient for overall comparisons among groups.

Although the overall fit function is a weighted sum of individual fit functions for groups, these individual functions are not statistically independent. Therefore, in general you cannot partition the degrees of freedom or 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 CALIS computes only a limited number of descriptive fit indices for individual groups.

- **fit function**
  The overall fit function is:

  $$F = \sum_{r=1}^{k} a_r F_r$$

where $a_r = \frac{(N_j-1)}{(N-k)}$ 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.

- **$\mathbf{R^2}$ 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.

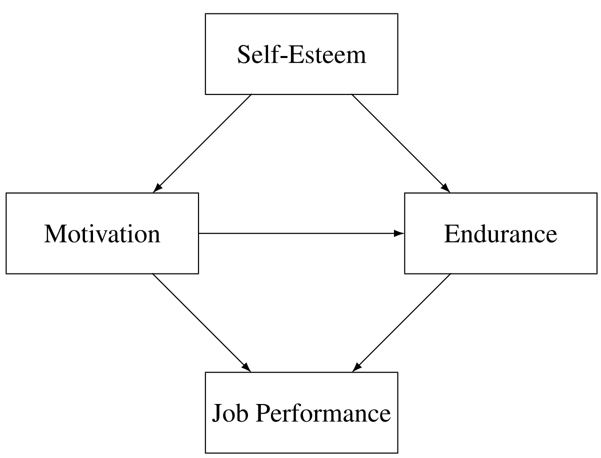
You can also use the DETERM statement to request the computations of determination coefficients for any subsets of dependent variables.

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 26.3 Direct and Indirect Effects of Self-Esteem on Job Performance



Clearly, each path in the diagram represents a direct effect of a predictor variable on an outcome variable. Less apparent are the total and indirect effects implied by the same path diagram. Despite this, interesting theoretical questions regarding the total and indirect effects can be raised in such a model. For example, even though there is no direct effect of Self-Esteem on Job Performance, what is its indirect effect on Job Performance? In addition to its direct effect on Job Performance, Motivation also has an indirect effect on Job Performance via its effect on Endurance. So, what is the total effect of Motivation on Job Performance and what portion of this total effect is indirect? The TOTEFF option of the CALIS statement and the EFFPART statement are designed to address these questions. By using the TOTEFF option or the EFFPART statement, PROC CALIS can compute the total, direct, and indirect effects of any sets of predictor variables on any sets of outcome variables. In this section, formulas for computing these effects are presented.

Formulas for Computing Total, Direct and Indirect Effects

No matter which modeling language is used, variables in a model can be classified into three groups. The first group is the so-called dependent variables, which serve as outcome variables at least once in the model. The other two groups consist of the remaining independent variables, which never serve as outcome variables in the model. The second group consists of independent variables that are unsystematic sources such as error and disturbance variables. The third group consists of independent variables that are systematic sources only.

Any variable, no matter which group it falls into, can have effects on the first group of variables. By definition, however, effects of variables in the first group on the other two groups do not exist. Because the effects of unsystematic sources in the second group are treated as residual effects on the first group of dependent variables, these effects are trivial in the sense that they always serve as direct effects only. That is, the effects from the second group of unsystematic sources partition trivially—total effects are always the same as the direct effects for this group. Therefore, for the purpose of effect analysis or partitioning, only the first group (dependent variables) and the third group (systematic independent variables) are considered.

Define u to be the set of n_u dependent variables in the first group and w to be the set of n_w systematic independent variables in the third group. Variables in both groups can be manifest or latent. All variables in the effect analysis is thus represented by the vector $(u', w')'$.

The $(n_u + n_w) \times (n_u + n_w)$ matrix $\mathbf{D}$ 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 $\mathbf{D}$ 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 $\mathbf{T}$ 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 $\boldsymbol{\mu}$ of *indirect* effects of column variables on the row variables is computed by the difference between $\mathbf{T}$ and $\mathbf{D}$ as:

$$\boldsymbol{\mu} = \begin{pmatrix} (I - \beta)^{-1} - I - \beta & (I - \beta)^{-1}\gamma - \gamma \\ 0 & 0 \end{pmatrix}$$

In PROC CALIS, any subsets of $\mathbf{D}$, $\mathbf{T}$, and $\boldsymbol{\mu}$ can be requested via the specification in the EFFPART statement. All you need to do is to specify the sets of column variables (variables that have effects on others) and row variables (variables that receive the effects, direct or indirect). Specifications of the column and row variables are done conveniently by specifying variable names—no matrix terminology is needed. This feature is very handy if you have some focused subsets of effects that you want to analyze a priori. See the EFFPART statement on page 1071 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 CALIS will check the appropriateness of effect computations by evaluating the stability coefficient of reciprocal causation. If the stability coefficient is greater than one, computations of the total and indirect effects will not be done.

Standardized Solutions

Standardized solutions are useful when you want to compare parameter values that are measured on quite different scales. PROC CALIS provides standardized solutions routinely. In standardizing a solution, parameters are classified into five groups:

- **path coefficients, regression coefficients, or direct effects**
  With each parameter 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 1273, 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 1263 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 26.27 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 1263 for more details). However, for just a little sacrifice in model fit, removing non-significant parameters increases the simplicity and precision of your model, which is the virtue that any modeler should look for.

Whether adding parameters by using the LM test indices or removing unnecessary parameters by the Wald statistics, you should not treat your modified model as if it were your original hypothesized model. That is, you should not publish your modified model as if it were hypothesized a priori. It is perfectly fine to use modification indices to gain additional insights for future research. But if you want to publish your modified model together with your original model, you should report the modification process that leads to your modified model. Theoretical justifications of the modified model should be supplemented if you want to make strong statements to support your modified model. Whenever possible, the best practice is to show reasonable model fit of the modified model with new data.

To modify your model either by LM test indices or Wald statistics, you can use the MODIFICATION or MOD option in the PROC CALIS statement. To customize the LM tests by setting specific regions of parameters, you can use the LMTESTS statements. PROC CALIS computes and displays the following default set of modification indices:

- **univariate Lagrange multiplier (LM) test indices for parameters in the model**
  These are second-order approximations of the decrease in the 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 greatest model improvements (that is, greatest 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 and the Analysis of Missing Patterns

If the DATA= data set contains raw data (rather than a covariance or correlation matrix), in general observations with missing values for any variables in the analysis are omitted from the computations. The only exception is with METHOD=FIML. Incomplete observations with at least one nonmissing variables in the analysis are also used for the estimation.

If a covariance or correlation matrix is read, missing values are allowed as long as every pair of variables has at least one nonmissing value. Unlike the raw data input, METHOD=FIML does not allow missing values in the covariance or correlation matrix.

When you use METHOD=FIML, PROC CALIS provide several analyses on the missing patterns of the raw input data sets. First, PROC CALIS shows the coverage results for the means and covariances. The coverage results refer to the proportions of data present for computing the means and the covariances. Because distinct missing patterns in the data sets are possible, the coverage proportions for the individual means and covariances could vary. Average coverage proportions of the means and covariances give you an overall idea about the missingness (or the lack of). In order to help locate the problematic means and covariances that have the low coverage, PROC CALIS shows the rank orders of the smallest coverages of mean and covariance elements. The number of smallest coverages shown for the means is equal to half of the total number of variables. The number of smallest coverages shown for the covariances is equal to half of the total number of the distinct elements in the lower triangular of the covariance matrix. However, in both cases at most 10 smallest coverages would be shown.

Second, PROC CALIS ranks the most frequent missing patterns in the data set (the nonmissing pattern is excluded in the ranking). Because the number of missing patterns could be quite large, PROC CALIS displays only a limited number of most frequent missing patterns in the output. You can use the MAXMISSPAT= and the TMISSPAT= options to control the number of missing patterns to display. See these options for details.

Third, PROC CALIS shows the means of the most frequent missing patterns, along with the means for the nonmissing pattern for comparison.

See Example 26.14 for an illustration of the use of the full information maximum likelihood method and the analysis of missing patterns.

Measures of Multivariate Kurtosis

In many applications, the manifest variables are not even approximately multivariate normal. If this happens to be the case with your data set, the default generalized least squares and maximum likelihood estimation methods are not appropriate, and you should compute the parameter estimates and their standard errors by an asymptotically distribution-free method, such as the WLS estimation method. If your manifest variables are multivariate normal, then they have a zero relative multivariate kurtosis, and all marginal distributions have zero kurtosis (Browne 1982). If your DATA= data set contains raw data, PROC CALIS computes univariate skewness and kurtosis and a set of multivariate kurtosis values. By default, the values of univariate skewness and kurtosis are corrected for bias (as in PROC UNIVARIATE), but using the BIASKUR option enables you to compute the uncorrected values also. The values are displayed when you specify the PROC CALIS statement option KURTOSIS.

In the following formulas, N denotes the sample size and p denotes the number of variables.

- **corrected variance for variable** z_j

$$\sigma_j^2 = \frac{1}{N-1}\sum_i^N (z_{ij} - \bar{z}_j)^2$$

- **uncorrected univariate skewness for variable** z_j

$$\gamma_{1(j)} = \frac{N \sum_i^N (z_{ij} - \bar{z}_j)^3}{\sqrt{N[\sum_i^N (z_{ij} - \bar{z}_j)^2]^3}}$$

- **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**

$$\gamma_2 = \frac{1}{N}\sum_i^N [(z_i - \bar{z})' \mathbf{S}^{-1} (z_i - \bar{z})]^2 - p(p+2)$$

  where $\mathbf{S}$ is the biased sample covariance matrix with N as the divisor.

- **relative multivariate kurtosis**

$$\eta_2 = \frac{\gamma_2 + p(p+2)}{p(p+2)}$$

- **normalized multivariate kurtosis**

$$\kappa_0 = \frac{\gamma_2}{\sqrt{8p(p+2)/N}}$$

- **Mardia based kappa**

$$\kappa_1 = \frac{\gamma_2}{p(p+2)}$$

- **mean scaled univariate kurtosis**

$$\kappa_2 = \frac{1}{3p} \sum_{j}^{p} \gamma_{2(j)}$$

- **adjusted mean scaled univariate kurtosis**

$$\kappa_3 = \frac{1}{3p} \sum_{j}^{p} \gamma^*_{2(j)}$$

with

$$\gamma^*_{2(j)} = \begin{cases} \gamma_{2(j)} \quad , & \text{if} \quad \gamma_{2(j)} > \frac{-6}{p+2} \\ \frac{-6}{p+2} \quad , & \text{otherwise} \end{cases}$$

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 CALIS 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 CALIS 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 CALIS statement is useful in this regard.

Except for the case of exploratory FACTOR model, you can specify initial estimates manually for all different types of models. If you do not specify some of the initial estimates and the RANDOM= option is not used, PROC CALIS will use a combination of good strategic methods to compute initial estimates for your model.

These initial estimation methods are used in PROC CALIS:

- two-stage least squares estimation
- instrumental variable method (Hägglund 1982; Jennrich 1987)
- approximate factor analysis method
- ordinary least squares estimation
- estimation method of McDonald (McDonald and Hartmann 1992)
- observed moments of manifest exogenous variables

The choice of initial estimation methods is dependent on the data and on the model. In general, it is difficult to tell in advance which initial estimation methods will be used for a given analysis. However, PROC CALIS displays the methods used to obtain initial estimates in the output. Notice that none of these initial estimation methods can be applied to the COSAN model because of the general formulation of the COSAN model. If you do not provide initial parameter estimates for the COSAN model, the default values or random values are used (see the START= and the RANDOM= options).

Poor initial values can cause convergence problems, especially with maximum likelihood estimation. Sufficiently large positive initial values for variance estimates (as compared with the covariance estimates) might help prevent a nonnegative definite initial predicted covariance model matrix from happening. If maximum likelihood estimation fails to converge, it might help to use METHOD=LSML, which uses the final estimates from an unweighted least squares analysis as initial estimates for maximum likelihood. Or you can fit a slightly different but better-behaved model and produce an OUTMODEL= data set, which can then be modified in accordance with the original model and used as an INMODEL= data set to provide initial values for another analysis.

If you are analyzing a covariance or scalar product matrix, be sure to take into account the scales of the variables. The default initial values might be inappropriate when some variables have extremely large or small variances.

Use of Optimization Techniques

No algorithm for optimizing general nonlinear functions exists that can always find the global optimum for a general nonlinear minimization problem in a reasonable amount of time. Since no single optimization technique is invariably superior to others, PROC CALIS provides a variety of optimization techniques that work well in various circumstances. However, you can devise problems for which none of the techniques in PROC CALIS can find the correct solution. All optimization techniques in PROC CALIS use $O(n^2)$ memory except the conjugate gradient methods, which use only $O(n)$ of memory and are designed to optimize problems with many parameters.

The PROC CALIS statement NLOPTIONS can be especially helpful for tuning applications with nonlinear equality and inequality constraints on the parameter estimates. Some of the options available in NLOPTIONS can also be invoked as PROC CALIS options. The NLOPTIONS statement can specify almost the same options as the SAS/OR NLP procedure.

Nonlinear optimization requires the repeated computation of the following:

- the function value (optimization criterion)
- the gradient vector (first-order partial derivatives)
- for some techniques, the (approximate) Hessian matrix (second-order partial derivatives)
- values of linear and nonlinear constraints
- the first-order partial derivatives (Jacobian) of nonlinear constraints

For the criteria used by PROC CALIS, computing the gradient takes more computer time than computing the function value, and computing the Hessian takes *much* more computer time and memory than computing the gradient, especially when there are many parameters to estimate. Unfortunately, optimization techniques that do not use the Hessian usually require many more iterations than techniques that do use the (approximate) Hessian, and so they are often slower. Techniques that do not use the Hessian also tend to be less reliable (for example, they might terminate at local rather than global optima).

The available optimization techniques are displayed in the following table and can be chosen by the OMETHOD=*name* option.

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 200 parameters, depending on the computer used). For large problems, a quasi-Newton technique, especially with the BFGS update, can be far more efficient.

For a poor choice of initial values, the Levenberg-Marquardt method seems to be more reliable.

If memory problems occur, you can use one of the conjugate gradient techniques, but they are generally slower and less reliable than the methods that use second-order information.

There are several options to control the optimization process. You can specify various termination criteria. You can specify the GCONV= option to specify a relative gradient termination criterion. If there are active boundary constraints, only those gradient components that correspond to inactive constraints contribute to the criterion. When you want very precise parameter estimates, the GCONV= option is useful. Other criteria that use relative changes in function values or parameter estimates in consecutive iterations can lead to early termination when active constraints cause small steps to occur. The small default value for the FCONV= option helps prevent early termination. Using the MAXITER= and MAXFUNC= options enables you to specify the maximum number of iterations and function calls in the optimization process. These limits are especially useful in combination with the INMODEL= and OUTMODEL= options; you can run a few iterations at a time, inspect the results, and decide whether to continue iterating.

Nonlinearly Constrained QN Optimization

The algorithm used for nonlinearly constrained quasi-Newton optimization is an efficient modification of Powell's Variable Metric Constrained WatchDog (VMCWD) algorithm (Powell 1978a, b, 1982a, b) A similar but older algorithm (VF02AD) is part of the Harwell library. Both VMCWD and VF02AD use Fletcher's VE02AD algorithm (also part of the Harwell library) for positive definite quadratic programming. The PROC CALIS QUANEW implementation uses a quadratic programming subroutine that updates and downdates the approximation of the Cholesky factor when the active set changes. The nonlinear QUANEW algorithm is not a feasible point algorithm, and the value of the objective function might not necessarily decrease (minimization) or increase (maximization) monotonically. Instead, the algorithm tries to reduce a linear combination of the objective function and constraint violations, called the *merit function*.

The following are similarities and differences between this algorithm and VMCWD:

- A modification of this algorithm can be performed by specifying VERSION=1, which replaces the update of the Lagrange vector μ with the original update of Powell (1978a, b), which is used in VF02AD. This can be helpful for some applications with linearly dependent active constraints.
- If the VERSION= option is not specified or VERSION=2 is specified, the evaluation of the Lagrange vector μ is performed in the same way as Powell (1982a, b) describes.
- Instead of updating an approximate Hessian matrix, this algorithm uses the dual BFGS (or DFP) update that updates the Cholesky factor of an approximate Hessian. If the condition of the updated matrix gets too bad, a restart is done with a positive diagonal matrix. At the end of the first iteration after each restart, the Cholesky factor is scaled.
- The Cholesky factor is loaded into the quadratic programming subroutine, automatically ensuring positive definiteness of the problem. During the quadratic programming step, the Cholesky factor of the projected Hessian matrix $\mathbf{Z}_k'\mathbf{G}\mathbf{Z}_k$ 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, b). However, this algorithm does not call for derivatives during the line search; hence, it generally needs fewer derivative calls than function calls. The VMCWD algorithm always requires the same number of derivative and function calls. It was also found in several applications of VMCWD that Powell's line-search method sometimes uses steps that are too long during the first iterations. In those cases, you can use the INSTEP= option specification to restrict the step length α of the first iterations.
- The watchdog strategy is similar to that of Powell (1982a, b). However, this algorithm does not return automatically after a fixed number of iterations to a former better point. A return here is further delayed if the observed function reduction is close to the expected function reduction of the quadratic model.
- Although Powell's termination criterion still is used (as FCONV2), the QUANEW implementation uses two additional termination criteria (GCONV and ABSGCONV).

This algorithm is automatically invoked when you specify the NLINCON statement. The nonlinear QUANEW algorithm needs the Jacobian matrix of the first-order derivatives (constraints normals) of the constraints:

$$(\nabla c_i) = (\frac{\partial c_i}{\partial x_j}), \quad i = 1, \ldots, nc, j = 1, \ldots, n$$

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 greater than 1 means that the radius can be increased, indicating a good quadratic function approximation.

For the Newton-Raphson technique (NRRIDG), the iteration history also includes the following information:

- the value of the ridge parameter. This value is 0 when a Newton step can be performed, and it is greater than 0 when either the Hessian approximation is singular or a Newton step fails to reduce the optimization criterion. Large values indicate optimization difficulties.
- the value of the ratio 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 1034. 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 CALIS 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 α_- of the previous iteration $k-1$ to compute a first value $\alpha_1^{(0)}$.
   - Using the DAMPSTEP<r> option:

     $$\alpha_1^{(0)} = \min(1, r\alpha_-)$$

     The initial value for the new step length can be no greater than r times the final step length 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 $0 < c \leq 4$ 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 CALIS computes an appropriate set of initial values automatically. However, for some of the model specifications (for example, structural equations with latent variables on the left-hand side and manifest variables on the right-hand side), PROC CALIS might generate very obscure initial values. In these cases, you have to set the initial values yourself.
  - Increase the initial values of the variance parameters by one of the following ways:
    * Set the variance parameter values in the model specification manually.
    * Use the 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 CALIS.

Singular Predicted Covariance Model Matrix

Sometimes you might inadvertently specify models with singular predicted covariance model matrices (for example, by fixing diagonal elements to zero). In such cases, you cannot compute maximum likelihood estimates (the ML function value F is not defined). Since singular predicted covariance model matrices can also occur temporarily in the minimization process, PROC CALIS tries in such cases to change the parameter estimates so that the predicted covariance model matrix becomes positive definite. This process does not always work well, especially if there are fixed instead of free diagonal elements in the predicted covariance model matrices. A famous example where you cannot compute ML estimates is a component analysis with fewer components than given manifest variables. See the section "FACTOR Statement" on page 1072 for more details. If you continue to obtain a singular predicted covariance model matrix after changing initial values and optimization techniques, then your model might be specified so that ML estimates cannot be computed.

Saving Computing Time

For large models, the most computing time is needed to compute the modification indices. If you do not really need the Lagrange multipliers or multiple Wald test indices (the univariate Wald test indices are the same as the t values), using the NOMOD option can save a considerable amount of computing time.

Predicted Covariance Matrices with Negative Eigenvalues

A covariance matrix cannot have negative eigenvalues, since a negative eigenvalue means that some linear combination of the variables has negative variance. PROC CALIS displays a warning if the predicted covariance matrix has negative eigenvalues but does not actually compute the eigenvalues. Sometimes this warning can be triggered by 0 or very small positive eigenvalues that appear negative because of numerical error. If you want to be sure that the predicted covariance matrix you are fitting can be considered to be a variance-covariance matrix, you can use the SAS/IML command *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 CALIS includes the following:

- a list of basic modeling information such as: the data set, the number of records read and used in the raw data set, the number of observations assumed by the statistical analysis, and the model type. When a multiple-group analysis is specified, the groups and their corresponding models are listed. This output assumes at least the PSHORT option.

- a list of all variables in the models. This output is displayed by default or by the PINITIAL option. It will not be displayed when you use the PSHORT or the PSUMMARY option.

  Depending on the modeling language, the variable lists vary, as shown in the following:

  - 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-, $\boldsymbol{\xi}$-, and $\boldsymbol{\eta}$- variables specified in the model
  - MSTRUCT: a list of the manifest variables specified in the model

- initial model specification. This output is displayed by default or by the PINITIAL option. It will not be displayed when you use the PSHORT or the PSUMMARY option.

  Depending on the modeling language, the sets of output vary, as shown in the following:

  - 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 1279 for more information.

- covariance coverage, variable coverage, average coverage of covariances and means, rank orders of the variable (mean) and covariance coverage, most frequent missing patterns in the input data set, and the means of the missing patterns when there are incomplete observations (with some missing values in the analysis variables) in the input raw data set and when you use METHOD=FIML or METHOD=LSFIML for estimation.

- covariance or correlation matrix to be analyzed and the value of its determinant if you specify the output option PCORR or PALL. A zero determinant indicates a singular data matrix. In this case, the generalized least squares estimates with default weight matrix **S** and maximum likelihood estimates cannot be computed.

- 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 1283.

- 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 1263. 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 1263 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 1270 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 1261 for more details.
- rank order of the largest normalized residuals if you specify the RESIDUAL option or at least the PRINT option.
- bar chart of the normalized residuals if you specify the RESIDUAL option or at least the PRINT option.
- plotting of smoothed density functions of residuals if you request ODS Graphics by the 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 1260 for more details. If you specify subsets of variables in the DETERM statements, the corresponding determination coefficients will also be shown. If one of the determinants in the formula for computing the determination coefficient is zero, the corresponding coefficient is displayed as the missing value '.'.

- the matrix of estimated covariances among the latent variables in the model if you specify the PLATCOV option or at least the PRINT option.

- the matrix of estimated covariances between latent and manifest variables in the model if you specify the PLATCOV option or at least the PRINT option.

- the vector of estimated means for the latent and manifest variables in the model if you specify the PLATCOV option or at least the PRINT option.

- the matrix **FSR** of latent variable scores regression coefficients if you specify the PLATCOV option or at least the PRINT option. The **FSR** matrix is a generalization of Lawley and Maxwell (1971, p. 109) factor scores regression matrix,

  $$\mathbf{FSR} = \hat{\mathbf{\Sigma}}_{yx}\hat{\mathbf{\Sigma}}_{xx}^{-1}$$

  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 1277 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 CALIS, 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 1277 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 1277 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 1277 for more detail.

- the stepwise multivariate Wald test for constraining estimated parameters to zero constants if the MODIFICATION (or MOD) or the PALL option is specified and the univariate probability is greater than the value specified in the PMW= option (default PMW=0.05). Note that this output is not applicable to the ULS and the DWLS estimation methods. See the section “Modification Indices” on page 1277 for more details.

ODS Table Names

PROC CALIS assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. For more information about ODS, see Chapter 20, “Using the Output Delivery System.”

There are numerous ODS tables in the CALIS procedure. The conditions for these ODS tables to display vary a lot. For convenience in presentation, the ODS tables for the PROC CALIS procedure are organized in the following categories:

- ODS tables for descriptive statistics, missing patterns, and residual analysis
- ODS tables for model specification and results
- ODS tables for supplementary model analysis
- ODS tables for modification indices
- ODS tables for optimization control and results

Many ODS tables are displayed when you set either a specialized option in a certain statement or a global display option in the PROC CALIS statement. Rather than requesting displays by setting specialized options separately, you can request a group of displays by using a global display option.

There are five global display levels, represented by five options: PALL (highest), PRINT, *default*, PSHORT, and PSUMMARY. The higher the level, the more output requested. The *default* printing level is in effect when you do not specify any other global printing options in the PROC CALIS statement. See the section "Global Display Options" on page 1022 for details.

In the following description of ODS tables whenever applicable, the lowest level of global printing options for an ODS table to print is listed. It is understood that global printing options at higher levels can also be used. For example, if PSHORT is the global display option to print an ODS table, you can also use PALL, PRINT, or *default*.

ODS Tables for Descriptive Statistics, Missing Patterns, and Residual Analysis

These ODS tables are group-oriented, meaning that each group has its own set of tables in the output. To display these tables in your output, you can set a specialized option in either the PROC CALIS or GROUP statement. If the specialized option is set in the PROC CALIS statement, it will apply to all groups. If the option is set in the GROUP statement, it will apply to the associated group only. Alternatively, you can set a global printing option in the PROC CALIS statement to print these tables. Either a specialized or a global printing option is sufficient to print the tables.

Table Names for Descriptive Statistics

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 Missing Pattern Analysis

ODS Table Name	Description	Specialized Option	Global Display Option
AveCoverage	Average proportion coverages of means (variances) and covariances	SIMPLE or PCORR[1]	Default[2]
MeanCovCoverage	Proportions of data present for means (variances) and covariances	SIMPLE or PCORR[1]	Default[2]
MissPatternsMeans	Means of the nonmissing and the most frequent missing patterns	SIMPLE or PCORR[1]	Default[2]
RankCovCoverage	Rank order of the covariance coverages	SIMPLE or PCORR[1]	Default[2]
RankMissPatterns	Rank order of the most frequent missing patterns	SIMPLE or PCORR[1]	Default[2]
RankVariableCoverage	Rank order of the proportion coverages of the variables	SIMPLE or PCORR[1]	Default[2]

1. You can use the NOMISSPAT option in the PROC CALIS statement to suppress the analytic output of the missing patterns. If you use the NOMISSPAT option in the GROUP statements, only the output of the missing pattern analysis for the corresponding groups are suppressed.
2. PROC CALIS outputs these tables by default only when there are incomplete observations in the data sets and you use METHOD=FIML or METHOD=LSFIML for estimation.

Table Names for Residual Displays

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 CALIS 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
Simtests	Simultaneous tests of parametric functions	PSHORT	SIMTESTS 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 CALIS: initial specification, (unstandardized) estimated model results, and standardized model results. To distinguish these three types of ODS tables, different suffixes for the ODS table names are used. An "Init" suffix indicates initial specification, while a "Std" suffix indicates standardized solutions. All other tables are for unstandardized solutions.

These ODS tables require some specialized options to print. If you set the specialized option in the PROC CALIS statement, it applies to all models. If you set the specialized option in the MODEL statement, it applies to the associated model only. Alternatively, to print *all* these ODS tables, you can use the PSHORT or any higher level global printing option in the PROC CALIS statement. Either a specialized or a global printing option is sufficient to print these ODS tables. The following is a summary of the specialized and global printing options for these three types of ODS tables:

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 CALIS or MODEL statement. If you want matrix output for all models, set this option in the PROC CALIS statement. If you want matrix output for a specific model, set this option in the associated MODEL statement only.

ODS Table Name	Description	Additional Option
COSANVariables	Variables in the COSAN model	
COSANModel	Mean and covariance structure formulas	
COSANMatrixSummary	Summary of COSAN model matrices	
COSANMatrix	Estimated model matrix	
COSANMatrixInit	Initial model matrix	
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

ODS Table Name	Description	Additional Option
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
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	

ODS Table Name	Description	Additional Option
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	
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

ODS Table Name	Description	Additional Option
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 CALIS or MODEL statement. If the specialized options are used in the PROC CALIS statement, they apply to all models. If the specialized options are used in the MODEL statement, they apply to the associated model only. For some of these ODS tables, certain specialized *statements* for the model might also enable the printing. Alternatively, you can use the global printing options in the PROC CALIS statement to print these ODS tables. Either a specialized option (or statement) or a global printing option is sufficient to print a particular ODS table.

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 CALIS or MODEL statement.
2. This refers to the EFFPART statement specifications.
3. NOSTAND option must not be specified in the MODEL or PROC CALIS statement.
4. PDETERM is an option specified in the PROC CALIS or MODEL statement, while DETERM is a statement name.

ODS Tables for Model Modification Indices

To print the ODS tables for model modification indices, you can use the MODIFICATION option in either the PROC CALIS or MODEL statement. When this option is set in the PROC CALIS statement, it applies to all models. When this option is set in the MODEL statement, it applies to the associated model only. Alternatively, you can also use the PALL option in the PROC CALIS statement to print these ODS tables.

If the NOMOD option is set in the PROC CALIS statement, these ODS tables are not printed for all models, unless the MODIFICATION is respecified in the individual MODEL statements. If the NOMOD option is set in the MODEL statement, then the ODS tables for modification do not print for the associated model.

For convenience in presentation, three different classes of ODS tables for model modifications are described in the following. First, ODS tables for ranking of LM indices are the default printing when the MODIFICATION option is specified. Second, ODS tables for LM indices in matrix forms require an additional option to print. Last, ODS tables for other modification indices, including the Wald test indices, require specific data-analytic conditions to print. While the first two classes of ODS tables are model-oriented (that is, each model has its own sets of output), the third one is not.

ODS Table Names for Ranking of LM indices

Rankings of the LM statistics in different regions of parameter space are the default printing format when you specify the MODIFICATION option in the PROC CALIS or MODEL statement. You can also turn off these default printing by the NODEFAULT option in the LMTESTS statement for models. If you want to print matrices of LM test statistics rather than the rankings of LM test statistics, you can use the NORANK or MAXRANK=0 option in the LMTESTS statement.

These ODS tables for ranking LM statistics are specific to the types of modeling languages used. This is noted in the last column of the following table.

ODS Table Name	Description	Model
LMRankCosanMatrix	Any COSAN model matrix	COSAN
LMRankCov	Covariances among variables	MSTRUCT
LMRankCovErr	Covariances among errors	LINEQS
LMRankCovErrofVar	Covariances among errors of variables	PATH
LMRankCovExog	Covariances among existing exogenous variables	LINEQS or PATH
LMRankCovFactors	Covariance among factors	FACTOR
LMRankCustomSet	Customized sets of parameters defined in LMTESTS statements	any model
LMRankErrorVar	Error variances	FACTOR
LMRankFactMeans	Factor means	FACTOR
LMRankIntercepts	Intercepts	FACTOR, LINEQS, or PATH
LMRankLisAlpha	LISMOD _ALPHA_	LISMOD
LMRankLisBeta	LISMOD _BETA_	LISMOD
LMRankLisGamma	LISMOD _GAMMA_	LISMOD
LMRankLisKappa	LISMOD _KAPPA_	LISMOD
LMRankLisLambdaX	LISMOD _LAMBDAX_	LISMOD

ODS Table Name	Description	Model
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
LMRankLoadings	Factor loadings	FACTOR
LMRankMeans	Means of existing variables	LINEQS, MSTRUCT, or PATH
LMRankPaths	All possible paths in the model	PATH
LMRankPathsFromEndo	Paths from existing endogenous variables	LINEQS
LMRankPathsFromExog	Paths from existing exogenous variables	LINEQS
LMRankPathsNewEndo	Paths to existing exogenous variables	LINEQS
LMRankRamA	_RAMA_ matrix	RAM
LMRankRamAlpha	_RAMALPHA_ matrix	RAM
LMRankRamNu	_RAMNU_ matrix	RAM
LMRankRamP11	_RAMP11_ matrix	RAM
LMRankRamP22	_RAMP22_ matrix	RAM

ODS Table Names for Lagrange Multiplier Tests in Matrix Form

To print matrices of LM test indices for a model, you must also use the LMMAT option in the LMTESTS statement for the model. Some of these matrices are printed by default, while others are printed only when certain regions of parameter are specified in the LM test sets. In the following tables, the ODS table names for LM test statistics in matrix form are listed for each model type.

The COSAN Model

ODS Table Name	Description	Selected Region in Test Sets
LMCosanMatrix	Any COSAN model matrix	(default)

The FACTOR Model

ODS Table Name	Description	Selected Region in Test Sets
LMFactErrv	Vector of error variances	FACTERRV (default)
LMFactFcov	Factor covariance matrix	FACTFCOV (default)
LMFactInte	Intercept vector	FACTINTE (default)
LMFactLoad	Factor loading matrix	FACTLOAD (default)
LMFactMean	Factor mean vector	FACTMEAN (default)

The LINEQS Model

ODS Table Name	Description	Selected Regions in Test Sets
LMEqsAlpha	_EQSALPHA_ vector	_EQSALPHA_ (default)
LMEqsBeta	_EQSBETA_ matrix	_EQSBETA_ (default)
LMEqsGammaSub	_EQSGAMMA_ matrix, excluding entries with error variables in columns	_EQSGAMMA_ (default)
LMEqsNewDep	New rows for expanding _EQSBETA_ and _EQSGAMMA_ matrices	NEWDEP
LMEqsNuSub	_EQSNU_ vector, excluding fixed zero means for error variables	_EQSNU_ (default)
LMEqsPhi	_EQSPHI_ matrix	_EQSPHI_ alone or _EQSPHI11_, _EQSPHI21_ and _EQSPHI22_ together
LMEqsPhi11	Upper left portion (exogenous variances and covariances) of the _EQSPHI_ matrix	_EQSPHI11_ (default)
LMEqsPhi21	Lower left portion (error variances and covariances) of the _EQSPHI_ matrix	_EQSPHI21_
LMEqsPhi22	Lower right portion (error variances and covariances) of the _EQSPHI_ matrix	_EQSPHI22_ (default)

The LISMOD Model

ODS Table Name	Description	Selected Regions in Test Sets
LMLisAlpha	LISMOD _ALPHA_ vector	_ALPHA_ (default)
LMLisBeta	LISMOD _BETA_ matrix	_BETA_ (default)
LMLisGamma	LISMOD _GAMMA_ matrix	_GAMMA_ (default)
LMLisKappa	LISMOD _KAPPA_ vector	_KAPPA_ (default)
LMLisLambdaX	LISMOD _LAMBDAX_ matrix	_LAMBDAX_ (default) or _LAMBDA_
LMLisLambdaY	LISMOD _LAMBDAY_ matrix	_LAMBDAY_ (default) or _LAMBDA_
LMLisNuX	LISMOD _NUX_ vector	_NUX_ (default) or _NU_
LMLisNuY	LISMOD _NUY_ vector	_NUY_ (default) or _NU_
LMLisPhi	LISMOD _PHI_ matrix	_PHI_ (default)
LMLisPsi	LISMOD _PSI_ matrix	_PSI_ (default)
LMLisThetaX	LISMOD _THETAX_ matrix	_THETAX_ (default) or _THETA_
LMLisThetaY	LISMOD _THETAY_ matrix	_THETAY_ (default) or _THETA_

The MSTRUCT Model

ODS Table Name	Description	Selected Regions in Test Sets
LMMstructCov	Covariance matrix	MSTRUCTCOV (default) or _COV_
LMMstructMean	Mean vector	MSTRUCTMEAN (default) or _MEAN_

The PATH Model

ODS Table Name	Description	Selected Regions in Test Sets
LMRamA	_RAMA_ matrix	ARROWS or _RAMA_ (default)
LMRamALeft	Left portion of the _RAMA_ matrix	_RAMA_LEFT_ alone or _RAMBETA_ and _RAMA_LL_ together
LMRamALL	Lower left portion of the _RAMA_ matrix	_RAMA_LL_
LMRamALower	Lower portion of the _RAMA_ matrix	NEWENDO or _RAMA_LOWER_ or _RAMA_LL_ and _RAMA_LR_ together
LMRamALR	Lower right portion of the _RAMA_ matrix	_RAMA_LR_
LMRamARight	Right portion of the _RAMA_ matrix	_RAMA_RIGHT_
LMRamAUpper	Upper portion of the _RAMA_ matrix	_RAMA_UPPER_ alone or _RAMBETA_ and _RAMGAMMA_ together
LMRamAlpha	_RAMALPHA_ matrix	INTERCEPTS (default)
LMRamBeta	Upper left portion of the _RAMA_ matrix	_RAMBETA_
LMRamGamma	Upper right portion of the _RAMA_ matrix	_RAMGAMMA_
LMRamNu	_RAMNU_ matrix	MEANS (default)
LMRamP	_RAMP_ matrix	_RAMP_ alone or _RAMP11_, _RAMP21_, and _RAMP22_ together
LMRamP11	Upper left portion of the _RAMP_ matrix	COVERR (default)
LMRamP21	Lower left portion of the _RAMP_ matrix	COVEXOGERR
LMRamP22	Lower right portion of the _RAMP_ matrix	COVEXOG (default)
LMRamW	_RAMW_ matrix	FIRSTMOMENTS alone or MEANS and INTERCEPTS together

The RAM Model

ODS Table Name	Description	Selected Regions in Test Sets
LMRamA	_RAMA_ matrix	_RAMA_ (default)
LMRamALeft	Left portion of the _RAMA_ matrix	_RAMA_LEFT_ alone or _RAMBETA_ and _RAMA_LL_ together
LMRamALL	Lower left portion of the _RAMA_ matrix	_RAMA_LL_
LMRamALower	Lower portion of the _RAMA_ matrix	NEWENDO or _RAMA_LOWER_ or _RAMA_LL_ and _RAMA_LR_ together
LMRamALR	Lower right portion of the _RAMA_ matrix	_RAMA_LR_
LMRamARight	Right portion of the _RAMA_ matrix	_RAMA_RIGHT_
LMRamAUpper	Upper portion of the _RAMA_ matrix	_RAMBETA_ and _RAMGAMMA_
LMRamAlpha	_RAMALPHA_ matrix	_RAMALPHA_ (default)
LMRamBeta	Upper left portion of the _RAMA_ matrix	_RAMBETA_
LMRamGamma	Upper right portion of the _RAMA_ matrix	_RAMGAMMA_
LMRamNu	_RAMNU_ matrix	_RAMNU_ (default)
LMRamP	_RAMP_ matrix	_RAMP_ alone or _RAMP11_, _RAMP21_, and _RAMP22_ together
LMRamP11	Upper left portion of the _RAMP_ matrix	_RAMP11_ (default)
LMRamP21	Lower left portion of the _RAMP_ matrix	_RAMP21_
LMRamP22	Lower right portion of the _RAMP_ matrix	_RAMP22_ (default)
LMRamW	_RAMW_ matrix	_RAMW_

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 CALIS statement: PRINT, PALL, or default (that is, NOPRINT is not specified). Also, you must not use the NOPRINT option in the NLOPTIONS statement. For some of these tables, you must also specify additional options, either in the PROC CALIS or the NLOPTIONS statement. Some restrictions might apply. Additional options and restrictions are noted in the last column.

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]
InitEstMethods	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 CALIS statement.
2. PALL option is specified in the NLOPTIONS statement.
3. Estimation method must not be ULS or DWLS.
4. Existence of dependent parameters.
5. Linear equality or boundary constraints are imposed.
6. Existence of parameter dependencies during optimization, but not due to model specification.

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS."

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" on page 612 in Chapter 21, "Statistical Graphics Using ODS."

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" on page 611 in Chapter 21, "Statistical Graphics Using ODS."

In the following table, ODS graph names and the options to display the graphs are listed.

Table 26.11 Graphs Produced by PROC CALIS

ODS Graph Name	Plot Description	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: CALIS Procedure

Example 26.1: Estimating Covariances and Correlations

This example shows how you can use PROC CALIS to estimate the covariances and correlations of the variables in your data set. Estimating the covariances introduces you to the most basic form of covariance structures—a saturated model with all variances and covariances as parameters in the model. To fit such a saturated model when there is no need to specify the functional relationships among the variables, you can use the MSTRUCT modeling language of PROC CALIS.

The following data set contains four variables q1–q4 for the quarterly sales (in millions) of a company. The 14 observations represent 14 retail locations in the country. The input data set is shown in the following DATA step:

```
data sales;
   input q1 q2 q3 q4;
   datalines;
1.03   1.54   1.11   2.22
1.23   1.43   1.65   2.12
3.24   2.21   2.31   5.15
1.23   2.35   2.21   7.17
 .98   2.13   1.76   2.38
1.02   2.05   3.15   4.28
1.54   1.99   1.77   2.00
1.76   1.79   2.28   3.18
1.11   3.41   2.20   3.21
1.32   2.32   4.32   4.78
1.22   1.81   1.51   3.15
1.11   2.15   2.45   6.17
1.01   2.12   1.96   2.08
1.34   1.74   2.16   3.28
;
```

Use the following PROC CALIS specification to estimate a saturated covariance structure model with all variances and covariances as parameters:

```
proc calis data=sales pcorr;
   mstruct var=q1-q4;
   run;
```

In the PROC CALIS statement, specify the data set with the DATA= option. Use the PCORR option to display the observed and predicted covariance matrix. Next, use the MSTRUCT statement to fit a covariance matrix of the variables that are provided in the VAR= option. Without further specifications such as the MATRIX statement, PROC CALIS assumes all elements in the covariance matrix are model parameters. Hence, this is a saturated model.

Output 26.1.1 shows the modeling information. Information about the model is displayed: the name and location of the data set, the number of data records read and used, and the number of observations in the analysis. The number of data records read is the actual number of records (or observations) that PROC CALIS processes from the data set. The number of data records used might or might not be the same as the actual number of records read from the data set. For example, records with missing values are read but not used in the analysis for the default maximum likelihood (ML) method. The number of observations refers to the N used for testing statistical significance and model fit. This number might or might not be the same as the number of records used for at least two reasons. First, if you use a frequency variable in the FREQ statement, the number of observations used is a weighted sum of the number of records, with the frequency variable being the weight. Second, if you use the NOBS= option in the PROC CALIS statement, you can override the number of observations that are used in the analysis. Because the current data set does not have any missing data and there are no frequency variables or an NOBS= option specified, these three numbers are all 14.

The model type is MSTRUCT because you use the MSTRUCT statement to define your model. The analysis type is covariances, which is the default. Output 26.1.1 then shows the four variables in the covariance structure model.

Output 26.1.1 Modeling Information of the Saturated Covariance Structure Model for the Sales Data

```
         Estimating the Covariance Matrix by the MSTRUCT Modeling Language

                              The CALIS Procedure
              Covariance Structure Analysis: Model and Initial Values

                             Modeling Information

                        Data Set              WORK.SALES
                        N Records Read        14
                        N Records Used        14
                        N Obs                 14
                        Model Type            MSTRUCT
                        Analysis              Covariances

                            Variables in the Model

                            q1   q2   q3   q4

                            Number of Variables = 4
```

Output 26.1.2 shows the initial covariance structure model for these four variables. All lower triangular elements (including the diagonal elements) of the covariance matrix are parameters in the model. PROC CALIS generates the names for these parameters: _Add01–_Add10. Because the covariance matrix is symmetric, all upper triangular elements of the matrix are redundant. The initial estimates for covariance are denoted by missing values no initial values were specified.

Output 26.1.2 Initial Saturated Covariance Structure Model for the Sales Data

Initial MSTRUCT _COV_ Matrix

	q1	q2	q3	q4
q1	. [_Add01]	. [_Add02]	. [_Add04]	. [_Add07]
q2	. [_Add02]	. [_Add03]	. [_Add05]	. [_Add08]
q3	. [_Add04]	. [_Add05]	. [_Add06]	. [_Add09]
q4	. [_Add07]	. [_Add08]	. [_Add09]	. [_Add10]

The PCORR option in the PROC CALIS statement displays the sample covariance matrix in Output 26.1.3. By default, PROC CALIS computes the unbiased sample covariance matrix (with variance divisor equal to $N-1$) and uses it for the covariance structure analysis.

Output 26.1.3 Sample Covariance Matrix for the Sales Data

Covariance Matrix (DF = 13)

	q1	q2	q3	q4
q1	0.33830	0.00020	0.03610	0.22137
q2	0.00020	0.22466	0.12653	0.24425
q3	0.03610	0.12653	0.60633	0.63012
q4	0.22137	0.24425	0.63012	2.66552

The fit summary and the fitted covariance matrix are shown in Output 26.1.4 and Output 26.1.5, respectively.

Output 26.1.4 Fit Summary of the Saturated Covariance Structure Model for the Sales Data

Fit Summary

Chi-Square	0.0000
Chi-Square DF	0
Pr > Chi-Square	.

Output 26.1.5 Fitted Covariance Matrix for the Sales Data

MSTRUCT _COV_ Matrix: Estimate/StdErr/t-value

	q1	q2	q3	q4
q1	0.3383	0.000198	0.0361	0.2214
	0.1327	0.0765	0.1260	0.2704
	2.5495	0.002587	0.2865	0.8186
q2	0.000198	0.2247	0.1265	0.2443
	0.0765	0.0881	0.1082	0.2251
	0.002587	2.5495	1.1693	1.0853
q3	0.0361	0.1265	0.6063	0.6301
	0.1260	0.1082	0.2378	0.3935
	0.2865	1.1693	2.5495	1.6012
q4	0.2214	0.2443	0.6301	2.6655
	0.2704	0.2251	0.3935	1.0455
	0.8186	1.0853	1.6012	2.5495

In Output 26.1.4, the model fit chi-square is 0 (df=0). The p-value cannot be computed because the degrees of freedom is zero. This fit is perfect because the model is saturated.

Output 26.1.5 shows the fitted covariance matrix, along with standard error estimates and t values in each cell. The variance and covariance estimates match exactly those of the sample covariance matrix shown in Output 26.1.3.

A common practice for determining statistical significance for estimates in structural equation modeling is to require the absolute value of t to be greater than 1.96, which is the critical value of a standard normal variate at α=0.05. While all diagonal elements in Output 26.1.5 show statistical significance, all off-diagonal elements are not significantly different from zero. The t values for these elements range from 0.002 to 1.601.

Output 26.1.6 shows the standardized estimates of the variance and covariance elements. This is also the correlation matrix under the MSTRUCT model. Standard error estimates and t values are computed with the correlation estimates. Note that because the diagonal element values are fixed at 1, no standard errors or t values are shown.

Output 26.1.6 Standardized Covariance Matrix for the Sales Data

Standardized MSTRUCT _COV_ Matrix: Estimate/StdErr/t-value

	q1	q2	q3	q4
q1	1.0000	0.000717 0.2773 0.002587	0.0797 0.2756 0.2892	0.2331 0.2623 0.8888
q2	0.000717 0.2773 0.002587	1.0000	0.3428 0.2448 1.4008	0.3156 0.2497 1.2640
q3	0.0797 0.2756 0.2892	0.3428 0.2448 1.4008	1.0000	0.4957 0.2092 2.3692
q4	0.2331 0.2623 0.8888	0.3156 0.2497 1.2640	0.4957 0.2092 2.3692	1.0000

Sometimes researchers do not need to estimate the standard errors that are in their models. You can suppress the standard error and t value computations by using the NOSE option in the PROC CALIS statement:

```
proc calis data=sales nose;
   mstruct var=q1-q4;
   run;
```

Output 26.1.7 shows the fitted covariance matrix with the NOSE option. These values are exactly the same as in the sample covariance matrix shown in Output 26.1.3.

Output 26.1.7 Fitted Covariance Matrix without Standard Error Estimates for the Sales Data

MSTRUCT _COV_ Matrix

	q1	q2	q3	q4
q1	0.3383	0.000198	0.0361	0.2214
q2	0.000198	0.2247	0.1265	0.2443
q3	0.0361	0.1265	0.6063	0.6301
q4	0.2214	0.2443	0.6301	2.6655

This example shows a very simple application of PROC CALIS: estimating the covariance matrix with standard error estimates. The covariance structure model is saturated. Several extensions of this very simple model are possible. To estimate the means and covariances simultaneously, see Example 26.2. To fit nonsaturated covariance structure models with certain hypothesized patterns, see Example 26.3 and Example 26.4. To fit structural models with implied covariance structures that are based on specified functional relationships among variables, see Example 26.6.

Example 26.2: Estimating Covariances and Means Simultaneously

This example uses the same data set that is used in Example 26.1 and estimates the means and covariances. Use the MSTRUCT model specification as shown in the following statements:

```
proc calis data=sales meanstr nostand;
   mstruct var=q1-q4;
   run;
```

In the PROC CALIS statement, specify the MEANSTR option to request the mean structure analysis in addition to the default covariance structure analysis. If you are not interested in the standardized solution, specify the NOSTAND option in the PROC CALIS statement to suppress computation of the standardized estimates. Without further model specification (such as the MATRIX statement), PROC CALIS assumes a saturated structural model with all means, variances, and covariances as model parameters.

Output 26.2.1 shows the modeling information. With the MEANSTR option specified in the PROC CALIS statement, the current analysis type is Means and Covariances, instead of the default Covariances in Example 26.1.

Output 26.2.1 Modeling Information of the Saturated Mean and Covariance Structure Model for the Sales Data

```
             Saturated Means and Covariance Structures Using MSTRUCT

                              The CALIS Procedure
             Mean and Covariance Structures: Model and Initial Values

                             Modeling Information

                    Data Set              WORK.SALES
                    N Records Read        14
                    N Records Used        14
                    N Obs                 14
                    Model Type            MSTRUCT
                    Analysis              Means and Covariances

                            Variables in the Model

                            q1   q2   q3   q4

                            Number of Variables = 4
```

Output 26.2.2 shows the fit summary of the current model. Again, this is a perfect model fit with 0 chi-square value and 0 degrees of freedom.

Output 26.2.2 Fit Summary of the Saturated Mean and Covariance Structure Model for the Sales Data

Fit Summary

Chi-Square	0.0000
Chi-Square DF	0
Pr > Chi-Square	.

Output 26.2.3 shows the estimates of the means, together with the standard error estimates and the t values. These estimated means are exactly the same as the sample means, which are not shown here.

Output 26.2.3 Mean Estimates for the Sales Data

MSTRUCT _Mean_ Vector

Variable	Estimate	Standard Error	t Value
q1	1.36714	0.16132	8.47491
q2	2.07429	0.13146	15.77902
q3	2.20286	0.21596	10.20008
q4	3.65500	0.45281	8.07176

Output 26.2.4 shows the variance and covariance estimates. These estimates are exactly the same as the elements in the sample covariance matrix. In addition, these estimates match the estimates in Output 26.1.5 of Example 26.1, where only the covariance structures are analyzed.

Output 26.2.4 Variance and Covariance Estimates for the Sales Data

MSTRUCT _COV_ Matrix: Estimate/StdErr/t-value

	q1	q2	q3	q4
q1	0.3383	0.000198	0.0361	0.2214
	0.1327	0.0765	0.1260	0.2704
	2.5495	0.002587	0.2865	0.8186
q2	0.000198	0.2247	0.1265	0.2443
	0.0765	0.0881	0.1082	0.2251
	0.002587	2.5495	1.1693	1.0853
q3	0.0361	0.1265	0.6063	0.6301
	0.1260	0.1082	0.2378	0.3935
	0.2865	1.1693	2.5495	1.6012
q4	0.2214	0.2443	0.6301	2.6655
	0.2704	0.2251	0.3935	1.0455
	0.8186	1.0853	1.6012	2.5495

These estimates are essentially the same as the sample means, variances, and covariances. This kind of analysis is much easier using PROC CORR with the NOMISS option. However, the main purpose of Example 26.1 and Example 26.2 is to introduce the MSTRUCT modeling language and some basic but important options in PROC CALIS. You can apply the MSTRUCT modeling language to more sophisticated situations that are beyond the saturated mean and covariance structure models. Example 26.3 and Example 26.4 fit some patterned covariance models that are nonsaturated. Also, options such as NOSE, NOSTAND, and MEANSTR are useful for all modeling languages in PROC CALIS.

Example 26.3: Testing Uncorrelatedness of Variables

This example uses the sales data in Example 26.1 and tests the uncorrelatedness of the variables in the model by using the MSTRUCT model specification. With the multivariate normality assumption, this is also the test of independence of the variables. The MATRIX statement defines the parameters in the model.

The uncorrelatedness model assumes that the correlations or covariances among the four variables are zero. Therefore, only the four diagonal elements of the covariance matrix, which represent the variances of the variables, are free parameters in the covariance structure model. To specify these parameters, use the MATRIX statement with the MSTRUCT model specification:

```
proc calis data=sales;
   mstruct var=q1-q4;
   matrix _cov_ [1,1], [2,2], [3,3], [4,4];
   run;
```

Example 26.1 specifies exactly the same MSTRUCT statement for the four variables. The difference here is the addition of the MATRIX statement. Without a MATRIX statement, the MSTRUCT model assumes that all nonredundant elements in the covariance matrix are model parameters. This assumption is not the case in the current specification. The MATRIX statement specification for the covariance matrix (denoted by the _cov_ keyword) specifies four free parameters on the diagonal of the covariance matrix: `[1,1]`, `[2,2]`, `[3,3]`, and `[4,4]`. All other unspecified elements in the covariance matrix are fixed zeros by default.

The uncorrelatedness model is displayed in the output for the initial model specification. Output 26.3.1 shows that all off-diagonal elements of the covariance matrix are fixed zeros while the diagonal elements are missing and labeled with _Parm1–_Parm4. PROC CALIS generates these parameter names automatically and estimates these four parameters in the analysis.

Output 26.3.1 Initial Uncorrelatedness Model for the Sales Data

Initial MSTRUCT _COV_ Matrix

	q1	q2	q3	q4
q1	. [_Parm1]	0	0	0
q2	0	. [_Parm2]	0	0
q3	0	0	. [_Parm3]	0
q4	0	0	0	. [_Parm4]

Output 26.3.2 shows the model fit chi-square test of the uncorrelatedness model. The chi-square is 6.528 (df=6, p=0.3667), which is not significant. This means that you fail to reject the uncorrelatedness model. In other words, the data is consistent with the uncorrelatedness model (zero covariances or correlations among the quarterly sales).

Output 26.3.2 Fit Summary of the Uncorrelatedness Model for the Sales Data

Fit Summary

Chi-Square	6.5280
Chi-Square DF	6
Pr > Chi-Square	0.3667

Output 26.3.3 shows the estimates of the covariance matrix under the uncorrelatedness model, together with standard error estimates and t values. All off-diagonal elements are fixed zeros in the estimation results.

Output 26.3.3 Estimates of Variance under the Uncorrelatedness Model for the Sales Data

MSTRUCT _COV_ Matrix: Estimate/StdErr/t-value

	q1	q2	q3	q4
q1	0.3383 0.1327 2.5495 [_Parm1]	0	0	0
q2	0	0.2247 0.0881 2.5495 [_Parm2]	0	0
q3	0	0	0.6063 0.2378 2.5495 [_Parm3]	0
q4	0	0	0	2.6655 1.0455 2.5495 [_Parm4]

This example shows how to specify free parameters in the MSTRUCT model by using the MATRIX statement. To specify the covariance matrix, use the _COV_ keyword in the MATRIX statement. To specify the parameters in the mean structures, you need use an additional MATRIX statement with the _MEAN_ keyword.

Two important notes regarding the MSTRUCT model specification are now in order:

- When you use the MSTRUCT statement without any MATRIX statements, *all* elements in the covariance matrix are *free parameters* in the model (for example, see Example 26.1). However, if the MATRIX statement includes at least one free or fixed parameter in the covariance matrix, PROC CALIS assumes that all other unspecified elements in the covariance matrix are *fixed zeros* (such as the current example).
- Using parameter names in the MATRIX statement specification is optional. In the context of the current example, naming the parameters is optional because there is no need to refer to them anywhere in the specification. PROC CALIS automatically generates unique names for these parameters. Alternatively, you can specify your own parameter names in the MATRIX statement. Naming parameters is not only useful for references, but is also indispensable when you need to constrain model parameters by referring to their names. See Example 26.4 to use parameter names to define a covariance pattern.

Example 26.4: Testing Covariance Patterns

In the test for sphericity, a covariance matrix is hypothesized to be a constant multiple of an identity matrix. That is, the null hypothesis for the population covariance matrix is

$$\mathbf{\Sigma} = \sigma^2 \mathbf{I}$$

where σ^2 is an unknown positive constant and $\mathbf{I}$ is an identity matrix. When this covariance pattern is applied to the sales data in Example 26.1, this hypothesis states that all four variables have the same variance σ^2 and are uncorrelated with each other. This model is more restricted than the uncorrelatedness model in Example 26.3, which requires uncorrelatedness but does not require equal variances. Use the following specification to conduct a sphericity test for the sales data:

```
proc calis data=sales;
   mstruct var=q1-q4;
   matrix _cov_ [1,1] = 4*sigma_sq;
   run;
```

This specification is similar to that of Example 26.3. The major difference is the MATRIX statement specification. The current example uses a parameter name sigma_sq to represent the unknown variance parameter σ^2, whereas Example 26.3 specifies only the locations of the four free variance parameters.

The current MATRIX statement specification uses a shorthand notation. On the left-hand side of the equal sign, `[1,1]` indicates the starting location of the covariance matrix. The matrix entries automatically proceed to `[2,2]`, `[3,3]` and so on, depending on the length of the parameter list specified on the right-hand sign of the equal sign. For example, if there is just one parameter on the right-hand side, the matrix specification contains only `[1,1]`. In the current example, the specification **`4*sigma_sq`** means that sigma_sq appears four times in the specification. As a result, the preceding MATRIX statement specification is equivalent to the following statement:

```
matrix _cov_ [1,1] = sigma_sq,
             [2,2] = sigma_sq,
             [3,3] = sigma_sq,
             [4,4] = sigma_sq;
```

This matrix is what is required by the sphericity test. Use either the expanded notation or the shorthand notation for specifying the covariance pattern. For details about various types of shorthand notation for parameter specifications, see the MATRIX statement.

Output 26.4.1 shows the initial model specification under the test of sphericity. All the diagonal elements are labeled with the same name sigma_sq, indicating that they are the same parameter.

Output 26.4.1 Covariance Model under Sphericity for the Sales Data

Initial MSTRUCT _COV_ Matrix

	q1	q2	q3	q4
q1	. [sigma_sq]	0	0	0
q2	0	. [sigma_sq]	0	0
q3	0	0	. [sigma_sq]	0
q4	0	0	0	. [sigma_sq]

Output 26.4.2 shows that the model fit chi-square is 31.5951 (df=9, p=0.0002). This means that the covariance pattern under the sphericity hypothesis is not supported.

Output 26.4.2 Fit Summary of the Sphericity Test for the Sales Data

Fit Summary

Chi-Square	31.5951
Chi-Square DF	9
Pr > Chi-Square	0.0002

Output 26.4.3 shows the estimated covariance matrix under the sphericity hypothesis. The variance estimate for all four diagonal elements is 0.9587 (standard error=0.1880).

Output 26.4.3 Fitted Covariance Matrix under the Sphericity Hypothesis for the Sales Data

MSTRUCT _COV_ Matrix: Estimate/StdErr/t-value

	q1	q2	q3	q4
q1	0.9587 0.1880 5.0990 [sigma_sq]	0	0	0
q2	0	0.9587 0.1880 5.0990 [sigma_sq]	0	0
q3	0	0	0.9587 0.1880 5.0990 [sigma_sq]	0
q4	0	0	0	0.9587 0.1880 5.0990 [sigma_sq]

This example shows how you can specify a simple covariance pattern by using the MATRIX statement. Use the same parameter names to constrain variance parameters that are supposed to be the same under the model. Constraining parameters by using the same parameter names is applicable not only to the MSTRUCT models, but also to more complicated covariance structure models, such as multiple-group modeling (see Example 26.18 and Example 26.20).

The MSTRUCT modeling language is handy when you can directly specify the covariance pattern or structures in your model. However, in most applications of structural equation modeling, it is difficult to specify such direct covariance structures. Instead, the covariance structures are usually implied from the functional relationships among the variables in the model. Using the MSTRUCT modeling language in such a situation is not easy. Fortunately, PROC CALIS supports other modeling languages that enable you to specify the functional relationships among variables. The functional relationships can be in the form of a set of path-like descriptions, a system of linear equations, or parameter specifications in matrices. See Example 26.6 for an introduction to using the PATH modeling language for specifying path models.

Example 26.5: Testing Some Standard Covariance Pattern Hypotheses

In Example 26.3, you test the uncorrelatedness of variables by using the MSTRUCT model specification. In Example 26.4, you test the sphericity of the covariance matrix by using the same model specification technique. In both examples, you need to specify the parameters in the covariance structure model explicitly by using the MATRIX statements.

Some covariance patterns are well-known in multivariate statistics, including the two tests in Example 26.3 and Example 26.4. To facilitate the tests of these "standard" covariance patterns, PROC CALIS provides the COVPATTERN= option to specify those standard covariance patterns more efficiently. With the COVPATTERN=option, you do not need to use the MSTRUCT and MATRIX statements to specify the covariance patterns explicitly. See the COVPATTERN= option for the supported covariance patterns. This example illustrates the use of the COVPATTERN= option.

In Example 26.3, you conduct a test of uncorrelatedness for the four variables in the sales data (see Example 26.1 for the data set). That is, the variables are hypothesized to be uncorrelated and only the four variances on the diagonal of the covariance matrix are population parameters of interest. The null hypothesis for the population covariance matrix is

$$\mathbf{\Sigma} = \begin{pmatrix} \mathrm{x} & 0 & 0 & 0 \\ 0 & \mathrm{x} & 0 & 0 \\ 0 & 0 & \mathrm{x} & 0 \\ 0 & 0 & 0 & \mathrm{x} \end{pmatrix}$$

where each x represents a distinct parameter (that is, the diagonal elements are not constrained with each other). You can test the diagonal covariance pattern easily by the following specification:

```
proc calis data=sales covpattern=diag;
   run;
```

The COVPATTERN=DIAG option specifies the required diagonal covariance pattern for the test. PROC CALIS then sets up the covariance structures automatically. Output 26.5.1 shows the initial specification of the covariance pattern. As required, only the diagonal elements are parameters in the test and the other elements are fixed to zero. PROC CALIS names the variance parameters automatically—that is, _varparm_1–_varparm_4 are the four parameters for the variances. This is the same pattern as shown in Output 26.3.1 of Example 26.3, although the parameter names are different.

Output 26.5.1 Initial Diagonal Pattern for the Covariance Matrix of the Sales Data

Initial MSTRUCT _COV_ Matrix

	q1	q2	q3	q4
q1	. [_varparm_1]	0	0	0
q2	0	. [_varparm_2]	0	0
q3	0	0	. [_varparm_3]	0
q4	0	0	0	. [_varparm_4]

Output 26.5.2 shows the results of the chi-square test of the diagonal covariance pattern. The chi-square is 5.44 (df=6, p=0.4887), which is not significant. You fail to reject the null hypothesis of the diagonal covariance pattern in the population.

Output 26.5.2 Fit Summary of the Diagonal Covariance Pattern Test for the Sales Data

Fit Summary	
Chi-Square	5.4400
Chi-Square DF	6
Pr > Chi-Square	0.4887

The numerical results shown in Output 26.5.2 are different from those of the same test by using the MSTRUCT model specification, which is shown in Output 26.3.2 of Example 26.3, although you do not reject the null hypothesis in both cases. The reason is that with the use of COVPATTERN= option, PROC CALIS applies the appropriate chi-square correction to the test statistic automatically. In the current example, the chi-square correction due to Bartlett (1950) has been applied. Test results with chi-square corrections are theoretically more accurate.

To obtain the same numerical results as those in Output 26.3.2, you can turn off the chi-square correction by using the CHICORRECT=0 option, as shown in the following specification:

```
proc calis data=sales covpattern=diag chicorrect=0;
   run;
```

Output 26.5.3 shows the fit summary results without any chi-square correction. The numerical results match exactly to those shown in Output 26.3.2 of Example 26.3.

Output 26.5.3 Fit Summary of the Diagonal Covariance Pattern Test for the Sales Data: No Chi-Square Correction

Fit Summary	
Chi-Square	6.5280
Chi-Square DF	6
Pr > Chi-Square	0.3667

Example 26.4 tests the sphericity of the covariance matrix of the same data set. The null hypothesis for the population covariance matrix is

$$\mathbf{\Sigma} = \sigma^2 \mathbf{I}$$

where σ^2 is an unknown positive constant and $\mathbf{I}$ is an identity matrix. You can use the following specification to test this hypothesis easily:

```
proc calis data=sales covpattern=sigsqi;
   run;
```

Output 26.5.4 shows the initial specification of the covariance pattern. As required, the diagonal elements are all the same parameter named _varparm, and all the off-diagonal elements are fixed to zero. This is the same pattern as shown in Output 26.4.1 of Example 26.4.

Output 26.5.4 Initial Covariance Pattern for the Sphericity Test on the Sales Data

Initial MSTRUCT _COV_ Matrix

	q1	q2	q3	q4
q1	. [_varparm]	0	0	0
q2	0	. [_varparm]	0	0
q3	0	0	. [_varparm]	0
q4	0	0	0	. [_varparm]

Output 26.5.5 shows the fit summary of the sphericity test. The chi-square is 27.747 (df=9, p=0.0011), which is statistically significant. You reject the sphericity hypothesis for the population covariance matrix.

Output 26.5.5 Fit Summary of the Sphericity Test on the Sales Data

Fit Summary

Chi-Square	27.7470
Chi-Square DF	9
Pr > Chi-Square	0.0011

Again, the numerical results in Output 26.5.5 are different from those shown in Output 26.4.2 of Example 26.4. This is because with the COVPATTERN=SIGSQI option, the chi-square correction due to Box (1949) has been applied in the current example. To turn off the automatic chi-square correction, you can use the following specification:

```
proc calis data=sales covpattern=sigsqi chicorrect=0;
   run;
```

As expected, the numerical results in Output 26.5.6 match exactly to those of in Output 26.4.2 of Example 26.4.

Output 26.5.6 Fit Summary of the Sphericity Test on the Sales Data: No Chi-Square Correction

Fit Summary

Chi-Square	31.5951
Chi-Square DF	9
Pr > Chi-Square	0.0002

This example shows that for the tests of some standard covariance patterns, you can use the COVPATTERN= option directly. As compared with the use of the explicit MSTRUCT model specifications, which are shown in Example 26.3 and Example 26.4, the use of COVPATTERN= option is more efficient and less error-prone in coding. In addition, it can apply chi-square corrections in appropriate situations.

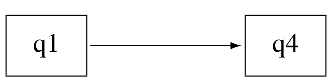
PROC CALIS also provides the test of some "standard" mean patterns by the MEANPATTERN= option. You can use the COVPATTERN= and MEANPATTERN= options together to define the desired combinations of covariance and mean patterns. See these two options for details. See Example 26.21 for a multiple-group analysis with the simultaneous use of the COVPATTERN= and MEANPATTERN= options. Certainly, the COVPATTERN= and MEANPATTERN= options are limited to the standard covariance and mean patterns provided by PROC CALIS. When you need to fit some specific (nonstandard) covariance or mean patterns, the MSTRUCT model specification would be indispensable. See Example 26.18 and Example 26.21 for applications.

Example 26.6: Linear Regression Model

This example shows how you can use PROC CALIS to fit the basic regression models. Unlike the preceding examples (Example 26.1, Example 26.2, Example 26.3, and Example 26.4) where you specify the covariance structures directly, in this example the covariance structures being analyzed are implied by the functional relationships specified in the model. The PATH modeling language introduced in the current example requires you to specify only the functional or path relationships among variables. PROC CALIS analyzes the implied covariance structures that are derived from the specified functional or path relationships.

Consider the same sales data as in Example 26.1. This example demonstrates a simple linear regression that uses q1 (the sales in the first quarter) to predict q4 (the sales in the fourth quarter).

In covariance structural analysis, or in general structural equation modeling, relationships among variables are usually represented by the so-called path diagram. For example, you can represent the linear regression of q4 on q1 by the following simple path diagram:



In the path diagram, q1 is an exogenous (or independent) variable and q4 is an endogenous (or dependent) variable. Formally, a variable in a path diagram is endogenous if there is at least one single-headed arrow pointing to it. Otherwise, the variable is exogenous. In some situations, researchers apply "causal" interpretations among variables in the path diagram, with the single-headed arrows indicating the causal directions. However, causal interpretations are not a requirement for using covariance structure analysis or structural equation modeling.

It is easy to transcribe the preceding path diagram into the PATH model specification in PROC CALIS, as shown in the following statements:

```
proc calis data=sales;
   path   q1  --->  q4;
   run;
```

Output 26.6.1 shows the modeling information of the linear regression model. It shows that all 14 observations are used and the model type is PATH. PROC CALIS analyzes the (implied) covariance structure model for the data. In the next table of Output 26.6.1, PROC CALIS shows the nature of the variables in the model: q4 is an endogenous manifest variable and q1 is an exogenous manifest variable. There is no latent variable in this simple path model.

Output 26.6.1 Modeling Information of the Linear Regression Model for the Sales Data

```
        Simple Linear Regression Model by the PATH Modeling Language

                          The CALIS Procedure
          Covariance Structure Analysis: Model and Initial Values

                         Modeling Information

                   Data Set              WORK.SALES
                   N Records Read        14
                   N Records Used        14
                   N Obs                 14
                   Model Type            PATH
                   Analysis              Covariances

                        Variables in the Model

                Endogenous    Manifest    q4
                              Latent
                Exogenous     Manifest    q1
                              Latent

                 Number of Endogenous Variables = 1
                 Number of Exogenous Variables  = 1
```

Output 26.6.2 shows the initial model specification. The path is in the first table. A parameter name is attached to the path. The name _Parm1, which is generated automatically by PROC CALIS, denotes the effect parameter of q1 on q4. In the context of linear regression, _Parm1 also denotes the regression coefficient.

Output 26.6.2 Initial Specification of the Linear Regression Model for the Sales Data

```
                   Initial Estimates for PATH List

          --------Path--------     Parameter      Estimate

          q1       --->     q4       _Parm1                  .
```

Output 26.6.2 *continued*

Initial Estimates for Variance Parameters

Variance Type	Variable	Parameter	Estimate
Exogenous	q1	_Add1	.
Error	q4	_Add2	.

NOTE: Parameters with prefix '_Add' are added by PROC CALIS.

Next, Output 26.6.2 shows the variance parameters in the model. You do not need to specify any of these parameters in the preceding PATH model specification—because PROC CALIS adds these parameters by default. _Add1 denotes the variance parameter for the exogenous variable q1. _Add2 denotes the error variance parameter for the endogenous variable q4.

In the PATH model of PROC CALIS, all variances of exogenous variables and all error variances of endogenous variables are free parameters by default. In most practical applications, these parameters are usually free parameters in models and it would be laborious to specify them each time when you fit a covariance structure model. Therefore, to make the PATH model specification more efficient and easier, PROC CALIS sets these free parameters by default. In fact, with these default parameters in the PATH model, PROC CALIS produces essentially the same regression analysis results as those produced by common linear regression procedures such as PROC REG. This consistency is shown in the subsequent estimation results for the current example.

You can also explicitly specify those otherwise default parameters of the PATH model in PROC CALIS. Depending on the modeling situation, you can set any parameter in the PATH model as a free, fixed, or constrained parameter. You can also provide names for the parameters. Naming parameters is very useful for parameter referencing and for setting up parameter constraints. See Example 26.4. For details, see the PATH statement and the section "The PATH Model" on page 1223.

Output 26.6.3 shows some fit statistics from the linear regression model. The model fit chi-square is 0 with 0 degrees of freedom. This is a perfect model fit. The fit is perfect because the covariance model contains three distinct elements (variance of q1, variance of q4, and covariance between q1 and q4) that are fitted perfectly by three parameters: _Parm1 for the effect of q1 on q4, _Add1 for the variance of variable q1, and _Add2 for the error variance of variable q4. Thus, the unconstrained linear regression model estimates are simply a transformation of the covariance elements. Hence, the model is saturated with a perfect fit and zero degrees of freedom.

Output 26.6.3 Model Fit of the Linear Regression Model for the Sales Data

Fit Summary

Chi-Square	0.0000
Chi-Square DF	0
Pr > Chi-Square	.
Standardized RMSR (SRMSR)	0.0000
RMSEA Estimate	.

Output 26.6.4 shows the estimates of the model. The effect of q1 on q4 is 0.6544 (standard error=0.7571). The associated t value is 0.86433, which is not significantly different from zero. The estimated variance of q1 is 0.3383 and the estimated error variance for q4 is 2.5207. Both estimates are significant.

Output 26.6.4 Parameter Estimates of the Linear Regression Model for the Sales Data

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
q1	--->	q4	_Parm1	0.65436	0.75707	0.86433

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Exogenous	q1	_Add1	0.33830	0.13269	2.54951
Error	q4	_Add2	2.52066	0.98869	2.54951

For a simple linear regression such as this one, you could have used PROC REG. You get essentially the same estimates by specifying the following statements:

```
proc reg data=sales;
   model q4 = q1;
   run;
```

Output 26.6.5 shows the parameter estimates from PROC REG. The intercept estimate is 2.7604 (standard error=1.1643) and the regression coefficient is 0.6544 (standard error=0.7880). The regression coefficient estimate matches PROC CALIS. However, the corresponding standard error estimate in PROC CALIS is 0.7571, which is slightly different from PROC REG. This difference is due to the different variance divisors that are used in calculating the standard error estimates. PROC CALIS uses $(N - 1)$ as the divisor (by default) while PROC REG uses $(N - q - 1)$, where N is the number of observations and q is the number of regression coefficients. In the current example, q is 1 so that the variance divisor in PROC REG is 1 less than the divisor in PROC CALIS. If you have at least a moderate sample size and the number of regression parameters is relatively small compared to the sample size, the discrepancy due to using different variance divisors is of little consequence.

Output 26.6.5 Parameter Estimates from PROC REG for the Sales Data

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	2.76040	1.16430	2.37	0.0353
q1	1	0.65436	0.78798	0.83	0.4225

By default, PROC CALIS analyzes only the covariance structures, which are properties of the second-order moments of the data. PROC CALIS does not automatically produce intercept estimates, which are properties of the first-order moments of the data.

In order to produce the intercept estimate in the linear regression context, you can add the MEANSTR (mean structures) option in the PROC CALIS statement, as shown in the following statements:

```
proc calis data=sales meanstr;
   path   q1  --->  q4;
   run;
```

Output 26.6.6 shows the parameter estimates of the model with the MEANSTR option added. Compared with Output 26.6.4, Output 26.6.6 produces one more table: estimates of the mean and intercept. The intercept estimate for q4 is 2.7604, which matches the intercept estimate from PROC REG. The estimated mean of q1 is 1.3671. All other estimates are the same for the analyses with and without the MEANSTR option.

Output 26.6.6 Parameter Estimates of the Linear Regression Model with the MEANSTR option for the Sales Data

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
q1	--->	q4	_Parm1	0.65436	0.75707	0.86433

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Exogenous	q1	_Add1	0.33830	0.13269	2.54951
Error	q4	_Add2	2.52066	0.98869	2.54951

Means and Intercepts

Type	Variable	Parameter	Estimate	Standard Error	t Value
Intercept	q4	_Add3	2.76040	1.12480	2.45413
Mean	q1	_Add4	1.36714	0.16132	8.47491

Linear regression estimates from PROC CALIS are comparable to those obtained from PROC REG, although the two procedures have different default treatments of the variance divisor in calculating the standard error estimates. With the MEANSTR option in the PROC CALIS statement, you can analyze the mean and covariance structures simultaneously. PROC CALIS prints the estimates of the intercepts and means when you model the mean structures.

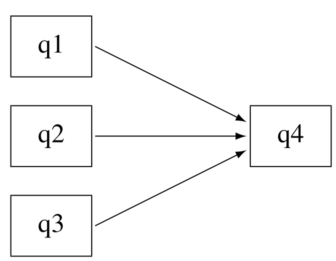
This example shows how you can fit the linear regression model as a PATH model in PROC CALIS. You need to specify only path relationships among the variables in the PATH statement, because the implied covariance structures are generated and analyzed by PROC CALIS. To make model specification more efficient, PROC CALIS sets default variance parameters for exogenous variables and default error variance parameters for endogenous variables. You can also overwrite these default parameters by explicit specifications. See Example 26.7 for some sophisticated regression models that you can specify with PROC CALIS. See Example 26.16 for a more elaborate path model specification.

Example 26.7: Multivariate Regression Models

This example shows how to analyze different types of multivariate regression models with PROC CALIS. Example 26.6 fits a simple linear regression model to the sales data that are described in Example 26.1. The simple linear regression model predicts the fourth quarter sales (q4) from the first quarter sales (q1). There is only one dependent (outcome) variable (q4) and one independent (predictor) variable (q1) in the analysis. Also, there are no constraints on the parameters. This example fits more sophisticated regression models. The models include more than one predictor. Some variables can serve as outcome variables and predictor variables at the same time. This example also illustrates the use of parameter constraints in model specifications and the use of the model fit statistics to search for a "best" model for the sales data.

Multiple Regression Model for the Sales Data

Consider a multiple regression model for q4. Instead of using just q1 as the predictor in the model as in Example 26.6, use all previous sales q1–q3 to predict the fourth-quarter sale (q4). The path model representation is shown in the following path diagram:



You can transcribe this path diagram into the following PATH model specification:

```
proc calis data=sales;
   path   q1 q2 q3 --->  q4;
   run;
```

In the path statement, the shorthand path specification

```
   path   q1 q2 q3 --->  q4;
```

is equivalent to the following specification:

```
   path   q1  --->  q4,
          q2  --->  q4,
          q3  --->  q4;
```

The shorthand notation provides a more convenient way to specify the path model. Some of the model fit statistics are shown in Output 26.7.1. This is a saturated model with perfect fit and zero degrees of freedom. Because the chi-square statistic is always smallest in a saturated model (with a zero chi-square value), it does not makes much sense to judge the model quality solely by looking at the chi-square value. However, a saturated model is useful for serving as a baseline model with which other nonsaturated competing models are compared.

Output 26.7.1 Model Fit of the Multiple Regression Model for the Sales Data

Fit Summary

Chi-Square	0.0000
Chi-Square DF	0
Pr > Chi-Square	.
Standardized RMSR (SRMSR)	0.0000
RMSEA Estimate	.
Akaike Information Criterion	20.0000
Bozdogan CAIC	36.3906
Schwarz Bayesian Criterion	26.3906

In addition to the model fit chi-square statistic, Output 26.7.1 also shows Akaike's information criterion (AIC), Bozdogan's CAIC, and Schwarz's Bayesian criterion (SBC) of the saturated model. The AIC, CAIC, and SBC are derived from information theory and henceforth they are referred to as the information-theoretic fit indices. These information-theoretic fit indices measure the model quality by taking the model parsimony into account. The root mean square error of approximation (RMSEA) also takes the model parsimony into account, but it is not an information-theoretic fit index. The values of these information-theoretic fit indices themselves do not indicate the quality of the model. However, when you fit several different models to the same data, you can order the models by these fit indices. The better the model, the smaller the fit index values. Unlike the chi-square statistic, these fit indices do not always favor a saturated model because a saturated model lacks model parsimony (the saturated model uses the most parameters to explain the data). The subsequent discussion uses these fit indices to select the "best" model for the sales data.

Output 26.7.2 shows the parameter estimates of the multiple regression model. In the first table, all path effect estimates are not statistically significant—that is, all t values are less than 1.96. The next table in Output 26.7.2 shows the variance estimates of q1–q3 and the error variance estimate for q4. All of these estimates are significant. The last table in Output 26.7.2 shows the covariances among the exogenous variables q1–q3. These covariance estimates are small and are not statistically significant.

Output 26.7.2 Parameter Estimates of the Multiple Regression Model for the Sales Data

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
q1	--->	q4	_Parm1	0.55980	0.64938	0.86205
q2	--->	q4	_Parm2	0.58946	0.84558	0.69711
q3	--->	q4	_Parm3	0.88290	0.51635	1.70988

Output 26.7.2 *continued*

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Exogenous	q1	_Add1	0.33830	0.13269	2.54951
	q2	_Add2	0.22466	0.08812	2.54951
	q3	_Add3	0.60633	0.23782	2.54951
Error	q4	_Add4	1.84128	0.72221	2.54951

Covariances Among Exogenous Variables

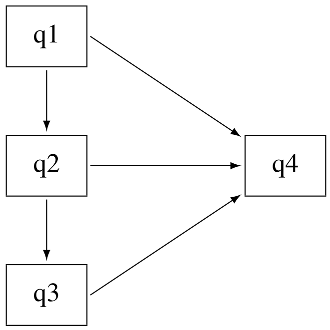
Var1	Var2	Parameter	Estimate	Standard Error	t Value
q2	q1	_Add5	0.0001978	0.07646	0.00259
q3	q1	_Add6	0.03610	0.12601	0.28649
q3	q2	_Add7	0.12653	0.10821	1.16931

In Output 26.7.2, the total number of parameter estimates is 10 (_Parm1–_Parm3 and _Add1–_Add7). Under the covariance structure model, these 10 parameters explain the 10 nonredundant elements in the covariance matrix for the sales data. That is why the model has a perfect fit with zero degrees of freedom.

In Output 26.7.2, notice that some parameters have the prefix '_Parm', while others have the prefix '_Add'. Both types of parameter names are generated by PROC CALIS. The parameters named with the '_Parm' prefix are those that were specified in the model, but were not named. In the current example, the parameters specified but not named are the path coefficients (effects) for the three paths in the PATH statement. The parameters named with the '_Add' prefix are default parameters added by PROC CALIS. In the current multiple regression example, the variances and covariances among the predictors (q1–q3) and the error variance for the outcome variable (q4) are default parameters in the model. In general, variances and covariances among exogenous variables and error variances of endogenous variables are default parameters in the PATH model. Avoid using parameter names with the '_Parm' and '_Add' prefixes to avoid confusion with parameters that are generated by PROC CALIS.

Direct and Indirect Effects Model for the Sales Data

In the multiple regression model, q1–q3 are all predictors that have direct effects on q4. This example considers the possibility of adding indirect effects into the multiple regression model. Because of the time ordering, it is reasonable to assume that there is a causal sequence q1 —> q2 —> q3. To implement this idea into the model, put two more paths into the preceding path diagram to form the following new path diagram:



With the q1 —> q2 and q2 —> q3 paths, q2 and q3 are no longer exogenous in the model. They become endogenous. The only exogenous variable in the model is q1, which has a direct effect in addition to indirect effects on q4. The direct effect is indicated by the q1 —> q4 path. The indirect effects are indicated by the following two causal chains: q1 —> q2 —> q4 and q1 —> q2 —> q3 —> q4. Similarly, q2 has a direct and an indirect effect on q4. However, q3 has only a direct effect on q4. You can use the following statements to specify this *direct and indirect effects* model:

```
proc calis data=sales;
   path   q1       --->  q2,
          q2       --->  q3,
          q1 q2 q3 --->  q4;
   run;
```

Although the *direct and indirect effects* model has two more paths in the PATH statement than does the preceding multiple regression model, the current model is more precise because it has one fewer parameter. By introducing the causal paths q1 —> q2 and q2—> q3, the six variances and covariances among q1–q3 are explained by: the two causal effects, the exogenous variance of q1, and the error variances for q2 and q3 (that is, five parameters in the model). Hence, the current *direct and indirect effects* model has one fewer parameter than the preceding multiple regression model.

Output 26.7.3 shows some model fit indices of the direct and indirect effects model. The model fit chi-square is 0.0934 with one degree of freedom. It is not significant. Therefore, you cannot reject the model on statistical grounds. The standardized root mean squares of residuals (SRMSR) is 0.028 and the root mean square error of approximation (RMSEA) is close to zero. Both indices point to a very good model fit. The AIC, CAIC, and SBC are all smaller than those of the saturated model, as shown in Output 26.7.1. This suggests that the *direct and indirect effects* model is better than the saturated model.

Output 26.7.3 Model Fit of the Direct and Indirect Effects Model for the Sales Data

Fit Summary

Chi-Square	0.0934
Chi-Square DF	1
Pr > Chi-Square	0.7600
Standardized RMSR (SRMSR)	0.0280
RMSEA Estimate	0.0000
Akaike Information Criterion	18.0934
Bozdogan CAIC	32.8449
Schwarz Bayesian Criterion	23.8449

Output 26.7.4 shows the parameter estimates of the *direct and indirect effects* model. All the path effects are not significant, while all the variance or error variance estimates are significant. Unlike the saturated model where you have covariance estimates among several exogenous variables (as shown in Output 26.7.2), in the *direct and indirect effects* model there is only one exogenous variable (q1) and hence there is no covariance estimate in the results.

Output 26.7.4 Parameter Estimates of the Direct and Indirect Effects Model for the Sales Data

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
q1	--->	q2	_Parm1	0.0005847	0.22602	0.00259
q2	--->	q3	_Parm2	0.56323	0.42803	1.31587
q1	--->	q4	_Parm3	0.55980	0.64705	0.86515
q2	--->	q4	_Parm4	0.58946	0.84524	0.69739
q3	--->	q4	_Parm5	0.88290	0.51450	1.71603

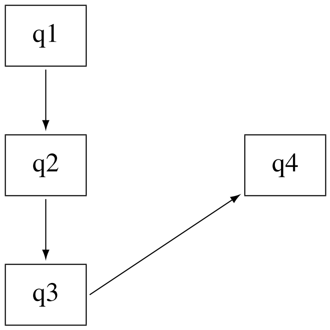
Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Exogenous	q1	_Add1	0.33830	0.13269	2.54951
Error	q2	_Add2	0.22466	0.08812	2.54951
	q3	_Add3	0.53506	0.20987	2.54951
	q4	_Add4	1.84128	0.72221	2.54951

Although the current *direct and indirect effects* model is better than the saturated model and both the SRMSR and RMSEA indicate a good model fit, the nonsignificant path effect estimates are unsettling. You continue to explore alternative models for the data.

Indirect Effects Model for the Sales Data

The saturated model includes only the direct effects of q1–q3 on q4, while the *direct and indirect effects* model includes both the direct and indirect effects of q1 and q2 on q4. An alternative model with only the indirect effects of q1 and q2 on q4, but without their direct effects, is possible. Such an *indirect effects* model is represented by the following path diagram:



You can easily transcribe this path diagram into the following PATH model specification:

```
proc calis data=sales;
   path   q1   --->  q2,
          q2   --->  q3,
          q3   --->  q4;
   run;
```

Output 26.7.5 shows some model fit indices for the *indirect effects* model. The chi-square model fit statistic is not statistically significant, so the model is not rejected. The standardized RMSR is 0.0905, which is a bit higher than the conventional value of 0.05 for an acceptable good model fit. However, the RMSEA is close to zero, which shows a very good model fit. The AIC, CAIC and SBC are all smaller than the *direct and indirect effects* model. These information-theoretic fit indices suggest that the *indirect effects* model is better.

Output 26.7.5 Model Fit of the Indirect Effects Model for the Sales Data

Fit Summary

Chi-Square	1.2374
Chi-Square DF	3
Pr > Chi-Square	0.7440
Standardized RMSR (SRMSR)	0.0905
RMSEA Estimate	0.0000
Akaike Information Criterion	15.2374
Bozdogan CAIC	26.7108
Schwarz Bayesian Criterion	19.7108

Output 26.7.6 shows the parameter estimates of the *indirect effects* model. All the variance and error variance estimates are statistically significant. However, only the path effect of q3 on q4 is statistically significant, and all other path effects are not. Having significant variances with nonsignificant paths raises some concerns about accepting the current model even though the AIC, CAIC, and SBC values suggest that it is the best model so far.

Output 26.7.6 Parameter Estimates of the Indirect Effects Model for the Sales Data

PATH List

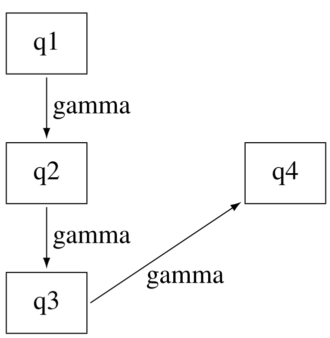
--------Path--------			Parameter	Estimate	Standard Error	t Value
q1	--->	q2	_Parm1	0.0005847	0.22602	0.00259
q2	--->	q3	_Parm2	0.56323	0.42803	1.31587
q3	--->	q4	_Parm3	1.03924	0.50506	2.05765

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Exogenous	q1	_Add1	0.33830	0.13269	2.54951
Error	q2	_Add2	0.22466	0.08812	2.54951
	q3	_Add3	0.53506	0.20987	2.54951
	q4	_Add4	2.01067	0.78865	2.54951

Constrained Indirect Effects Model for the Sales Data

In the preceding *indirect effects* model, some path effects are not significant. In the current model, all the path effects are constrained to be equal. The following path diagram represents the *constrained indirect effects* model:



Except for one notable difference, this path diagram is the same as the path diagram for the preceding *indirect effects* model. The current path diagram labels all the paths with the same name (gamma) to signify that they are the same parameter. You can specify this *constrained indirect effects* model with this chosen constraint on the path effects by the using following statements:

```
proc calis data=sales;
   path    q1    --->   q2       = gamma,
           q2    --->   q3       = gamma,
           q3    --->   q4       = gamma;
   run;
```

In the PATH statement, append an equal sign and a parameter name gamma in each of the path entries. This specification means that all the associated path effects are the same parameter named gamma.

Output 26.7.7 shows some fit indices for the *constrained indirect effects* model. Again, the model fit chi-square statistic is not significant. However, the SRMSR is 0.2115, which is too large to accept as a good model. The RMSEA is 0.0499, which still indicates a good model fit. The AIC, CAIC, and SBC values are a bit smaller than those of the preceding unconstrained *indirect effects* model. Therefore, it seems that constraining the path effects leads to a slightly better model.

Output 26.7.7 Model Fit of the Constrained Indirect Effects Model for the Sales Data

Fit Summary

Chi-Square	5.1619
Chi-Square DF	5
Pr > Chi-Square	0.3964
Standardized RMSR (SRMSR)	0.2115
RMSEA Estimate	0.0499
Akaike Information Criterion	15.1619
Bozdogan CAIC	23.3572
Schwarz Bayesian Criterion	18.3572

Output 26.7.8 shows the parameter estimates of the *constrained indirect effects* model. Again, all variance and error variance estimates are significant, and all path effects are not significant. The effect estimate is 0.24 (standard error=0.19, t=1.25).

Output 26.7.8 Parameter Estimates of the Constrained Indirect Effects Model for the Sales Data

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
q1	--->	q2	gamma	0.24014	0.19152	1.25390
q2	--->	q3	gamma	0.24014	0.19152	1.25390
q3	--->	q4	gamma	0.24014	0.19152	1.25390

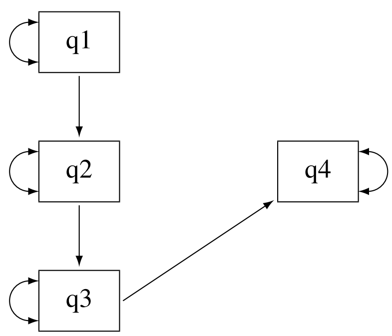
Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Exogenous	q1	_Add1	0.33830	0.13269	2.54951
Error	q2	_Add2	0.24407	0.09573	2.54951
	q3	_Add3	0.55851	0.21907	2.54951
	q4	_Add4	2.39783	0.94051	2.54951

Constrained Indirect Effects and Error Variances Model for the Sales Data

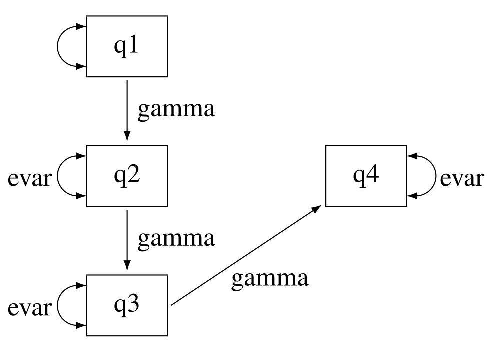
In addition to constraining all the path effects in the preceding model, the current model constrains all the error variances. Before using a path diagram to represent the current constrained indirect effects and constrained error variances, it is important to realize that you have not manually defined variances and covariances in the path diagrams for all of the preceding models. The default parameterization in PROC CALIS defined those parameters.

Represent the variances and covariances in a path diagram with double-headed arrows. When a double-headed arrow points to a single variable, it represents the variance parameter. When a double-headed arrow points to two distinct variables, it represents the covariance between the two variables. Consider the unconstrained *indirect effects* model for the sales data as an example. A more complete path diagram representation is as follows:



In this path diagram, a double-headed arrow on each variable represents variance or error variance. For q1, the double-headed arrow represents the variance parameter of q1. For other variables, the double-headed arrows represent error variances because those variables are endogenous (that is, they are predicted from other variables) in the model.

In order to represent the equality-constrained parameters in the model, you can put parameter names in the respective parameter locations in the path diagram. For the current *constrained indirect effects and error variances* model, you can represent the model by the following path diagram:



In the path diagram, label all the path effects by the parameter gamma and all error variances by the parameter evar. The double-headed arrow attached to q1 is not labeled by any name. This means that it is an unnamed free parameter in the model.

You can transcribe the path diagram into the following statements:

```
proc calis data=sales;
   path   q1   --->  q2      = gamma,
          q2   --->  q3      = gamma,
          q3   --->  q4      = gamma;
   pvar   q2 q3 q4 = 3 * evar;
   run;
```

The specification in the PATH statement is the same as the preceding PATH model specification for the *constrained indirect effects* model. The new specification here is the PVAR statement. You use the PVAR statement to specify partial variances, which include the (total) variances of exogenous variables and the error variances of the endogenous variables. In the PVAR statement, you specify the variables for which you intend to define variances. If you do not specify anything after the list of variables, the variances of these variables are unnamed free parameters. If you put an equal sign after the variable lists, you can specify parameter names, initial values, or fixed parameters for the variances of the variables. See the PVAR statement for details. In the current model, **3*evar** means that you want to specify evar three times (for the error variance parameters of q2, q3, and q4).

Note that you did not specify the variance of q1 in the PVAR statement. This variance is a default parameter in the model, and therefore you do not need to specify it in the PVAR statement. Alternatively, you can specify it explicitly in the PVAR statement by giving it a parameter name. For example, you can specify the following:

```
pvar   q2 q3 q4 = 3 * evar,
       q1       = MyOwnName;
```

Or, you can specify it explicitly without giving it a parameter name, as shown in following statement:

```
pvar   q2 q3 q4 = 3 * evar,
       q1 ;
```

All these specifications lead to the same estimation results. The difference between the two specifications is the explicit parameter name for the variance of q1. Without putting q1 in the PVAR statement, the variance parameter is named with the prefix _Add, which is generated as a default parameter by PROC CALIS. With the explicit specification of q1, the variance parameter is named MyOwnName. With the explicit specification of q1, but without giving it a parameter name in the PVAR statement, the variance parameter is named with the prefix _Parm, which PROC CALIS generates for unnamed free parameters.

Output 26.7.9 shows some fit indices for the *constrained indirect effects and error variances* model. The model fit chi-square is 19.7843, which is significant at the 0.05 α-level. In practice, the model fit chi-square statistic is not the only criterion for judging model fit. In fact, it might not even be the most commonly used criterion for measuring model fit. Other criteria such as the SRMSR and RMSEA are more popular or important. Unfortunately, the values of these two fit indices do not support the current constrained model either. The SRMSR is 1.5037 and the RMSEA is 0.3748. Both are much greater than the commonly accepted 0.05 criterion.

Output 26.7.9 Model Fit of the Constrained Indirect Effects and Error Variances Model for the Sales Data

Fit Summary

Chi-Square	19.7843
Chi-Square DF	7
Pr > Chi-Square	0.0061
Standardized RMSR (SRMSR)	1.5037
RMSEA Estimate	0.3748
Akaike Information Criterion	25.7843
Bozdogan CAIC	30.7015
Schwarz Bayesian Criterion	27.7015

The AIC, CAIC, and SBC values are all much greater than those of the preceding *constrained indirect effects* model. Therefore, constraining the error variances in addition to the constrained indirect effects does not lead to a better model.

Output 26.7.10 shows the parameter estimates of the *constrained indirect effects and error variances* model. All estimates are significant in the model, which is often desirable. However, because of the bad model fit, this model is not acceptable.

Output 26.7.10 Parameter Estimates of the Constrained Indirect Effects and Error Variances Model for the Sales Data

PATH List

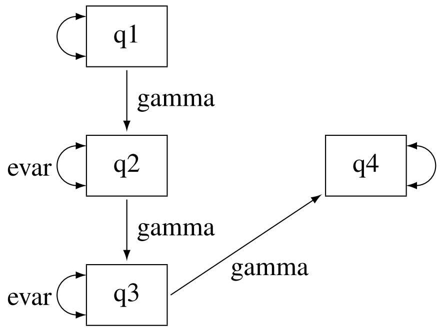
--------Path--------			Parameter	Estimate	Standard Error	t Value
q1	--->	q2	gamma	0.64733	0.16128	4.01368
q2	--->	q3	gamma	0.64733	0.16128	4.01368
q3	--->	q4	gamma	0.64733	0.16128	4.01368

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	q2	evar	1.00220	0.22695	4.41588
	q3	evar	1.00220	0.22695	4.41588
	q4	evar	1.00220	0.22695	4.41588
Exogenous	q1	_Add1	0.33830	0.13269	2.54951

Partially Constrained Model for the Sales Data

In the preceding model, constraining all error variances to be same shows that the model fit is unacceptable, even though all parameter estimates are significant. Relaxing those constraints a little might improve the model. The following path diagram represents such a *partially constrained* model:



The only difference between the current *partially constrained* model and the preceding *constrained indirect effects and error variances* model is that the error variance for q4 is no longer constrained to be equal to the error variances of q2 and q3. In the path diagram, evar is no longer attached to the double-headed arrow that is associated with the error variance of q4. You can transcribe this path diagram representation into the following PATH model specification:

```
proc calis data=sales;
   path   q1   --->  q2      = gamma,
          q2   --->  q3      = gamma,
          q3   --->  q4      = gamma;
   pvar   q2 q3 = 2 * evar,
          q4 q1;
   run;
```

Now, the PVAR statement has only the error variances of q2 and q3 constrained to be equal. The error variance of q4 and the variance of q1 are free parameters without constraints.

Output 26.7.11 shows some fit indices for the *partially constrained* model. The chi-square model fit test statistic is not significant. The SRMSR is 0.3877 and the RMSEA is 0.1164. These are far from the conventional acceptance level of 0.05. However, the AIC, CAIC, and SBC values are all slightly smaller than the *constrained indirect effects* model, as shown in Output 26.7.7. In fact, these information-theoretic fit indices suggest that the *partially constrained* model is the best model among all models that have been considered.

Output 26.7.11 Model Fit of the Partially Constrained Model for the Sales Data

Fit Summary

Chi-Square	7.0575
Chi-Square DF	6
Pr > Chi-Square	0.3156
Standardized RMSR (SRMSR)	0.3877
RMSEA Estimate	0.1164
Akaike Information Criterion	15.0575
Bozdogan CAIC	21.6138
Schwarz Bayesian Criterion	17.6138

Output 26.7.12 shows the parameter estimates of the *partially constrained* model. Again, all variance and error variance parameters are statistically significant. However, the path effects are only marginally significant.

Output 26.7.12 Parameter Estimates of the Partially Constrained Model for the Sales Data

```
                              PATH List

                                                           Standard
--------Path--------     Parameter      Estimate             Error        t Value

q1       --->     q2     gamma            0.35546          0.18958        1.87497
q2       --->     q3     gamma            0.35546          0.18958        1.87497
q3       --->     q4     gamma            0.35546          0.18958        1.87497

                         Variance Parameters

Variance                                                    Standard
Type          Variable     Parameter      Estimate             Error        t Value

Error         q2           evar            0.40601          0.11261        3.60555
              q3           evar            0.40601          0.11261        3.60555
              q4           _Parm1          2.29415          0.89984        2.54951
Exogenous     q1           _Parm2          0.33830          0.13269        2.54951
```

Which Model Should You Choose?

You fit various models in this example for the sales data. The fit summary of the models is shown in the following table:

	1	**2**	**3**	**4**	**5**	**6**
	Saturated	**Direct and Indirect Effects**	**Indirect Effects**	**Constrained Indirect Effects**	**Constrained Indirect Effects and Error Variances**	**Partially Constrained**
df	0	1	3	5	7	6
p-value	.	0.76	0.74	0.40	0.01	0.32
SRMSR	0	0.03	0.09	0.21	1.50	0.39
RMSEA	.	0.00	0.00	0.05	0.37	0.12
AIC	20.00	18.09	15.24	15.16	25.78	15.06
CAIC	36.39	32.84	26.71	23.36	30.70	21.61
SBC	26.39	23.84	19.71	18.36	27.70	17.61

As discussed previously, the model fit chi-square test statistic always favors models with a lot of parameters. It does not take model parsimony into account. In particular, a saturated model (Model 1) always has a perfect fit. However, it does not explain the data in a concise way. Therefore, the model fit chi-square statistic is not used here for comparing the competing models.

The standardized root mean square residual (SRMSR) also does not take the model parsimony into account. It tells you how the fitted covariance matrix is different from the observed covariance matrix in a certain standardized way. Again, it always favors models with a lot of parameters. As shown in the preceding table, the more parameters (the fewer degrees of freedom) the model has, the smaller the SRMSR is. A conventional criterion is to accept a model with SRMSR less than 0.05. Applying this criterion, only the saturated model (Model 1) and the *direct and indirect effects* (Model 2) models are acceptable. The *indirect effects* model (Model 3) is marginally acceptable.

The root mean square error of approximation (RMSEA) fit index does take model parsimony into account. With the 'RMSEA less than 0.05 criterion', the *constrained indirect effects and error variances* model (Model 5) and the *partially constrained* model (Model 6) are not acceptable.

The information-theoretic fit indices such as the AIC, CAIC, and SBC also take model parsimony into account. All of these indices point to the *partially constrained* model (Model 6) as the best model among the competing models. However, because this model has a relatively bad absolute fit, as indicated by the large SRMSR value (0.39), accepting this model is questionable. In addition, the information-theoretic fit indices of the *indirect effects* model (Model 3) and of the *constrained indirect effects* model (Model 4) are not too different from those of the *partially constrained* model (Model 6). The indirect effects model is especially promising because it has relatively small SRMSR and RMSEA values. The drawback is that some path effect estimates in the indirect effects model are not significant. Perhaps collecting and analyzing more data might confirm these promising models with significant path effects.

You might not be able to draw a unanimous conclusion about the best model for the sales data of this example. Different fit indices in structural equation modeling do not always point to the same conclusions. The analyses in the current example show some of the complexity of structural equation modeling. Some interesting questions about model selections are:

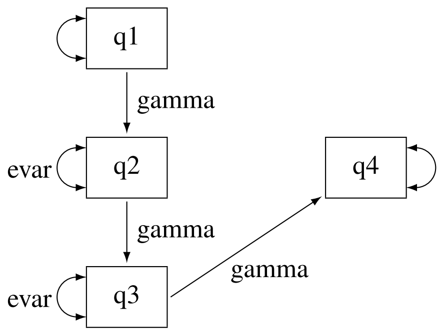
- Do you choose a model based on a single fit criterion? Or, do you consider a set of model fit criteria to weigh competing models?
- Which fit index criterion is the most important for judging model fit?
- In selecting your "best" model, how do you take "chance" into account?
- How would you use your substantive theory to guide your model search?

The answers to these interesting research questions might depend on the context. Nonetheless, PROC CALIS can help you in the model selection process by computing various kinds of fit indices. (Only a few of these fit indices are shown in the output of this example. See the FITINDEX statement for a wide variety of fit indices that you can obtain from PROC CALIS.)

Alternative PATH Model Specifications for Variances and Covariances

The PATH modeling language of PROC CALIS is designed to map the path diagram representation into the PATH statement syntax efficiently. For any path that is denoted by a single-headed arrow in the path diagram, you can specify a path entry in the PATH statement. You can also specify double-headed arrows in the PATH statement.

Consider the preceding path diagram for the *partially constrained* model for the sales data. You use double-headed arrows to denote variances or error variances of the variables. The path diagram is shown in the following:



As discussed previously, you can use the PVAR statement to specify these variances or error variances as in following syntax:

```
pvar   q2 q3 = 2 * evar,
       q4 q1;
```

Alternatively, you can specify these double-headed arrows directly as paths in the PATH statement, as shown in the following statements:

```
proc calis data=sales;
   path   q1   --->  q2      = gamma,
          q2   --->  q3      = gamma,
          q3   --->  q4      = gamma,
          <-->    q2 q3      = 2 * evar,
          <-->    q4 q1;
   run;
```

To specify the double-headed paths pointing to individual variables, you begin with the double-headed arrow notation <-->, followed by the list of variables. For example, in the preceding specification, the error variance of q4 and the variance of q1 are specified in the last path entry of the PATH statement. If you want to define the parameter names for the variances, you can add a parameter list after an equal sign in the path entries. For example, the error variances of q2 and q3 are denoted by the free parameter evar in a path entry in the PATH statement.

Alternatively, you can specify the double-headed arrow paths literally in a PATH statement, as shown in the following equivalent specification:

```
proc calis data=sales;
   path   q1   --->  q2      = gamma,
          q2   --->  q3      = gamma,
          q3   --->  q4      = gamma,
          q2   <-->  q2      = evar,
          q3   <-->  q3      = evar,
          q4   <-->  q4,
          q1   <-->  q1;
   run;
```

For example, the path entry q1 <--> q1 specifies the variance of q1. It is an unnamed free parameter in the model.

Output 26.7.13 show the parameter estimates for this alternative specification method. All these estimates match exactly those with the PVAR statement specification, as shown in Output 26.7.12. The only difference is that all estimation results are now presented under one PATH List, as shown in Output 26.7.13, instead of as two tables as shown in Output 26.7.12.

Output 26.7.13 Path Estimates of the Partially Constrained Model for the Sales Data

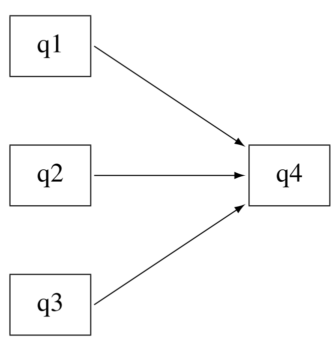
PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
q1	--->	q2	gamma	0.35546	0.18958	1.87497
q2	--->	q3	gamma	0.35546	0.18958	1.87497
q3	--->	q4	gamma	0.35546	0.18958	1.87497
q2	<-->	q2	evar	0.40601	0.11261	3.60555
q3	<-->	q3	evar	0.40601	0.11261	3.60555
q4	<-->	q4	_Parm1	2.29415	0.89984	2.54951
q1	<-->	q1	_Parm2	0.33830	0.13269	2.54951

The double-headed arrow path syntax applies to covariance specification as well. For example, the following PATH statement specifies the covariances among variables x1–x3:

```
path    x2 <--> x1,
        x3 <--> x1,
        x3 <--> x2;
```

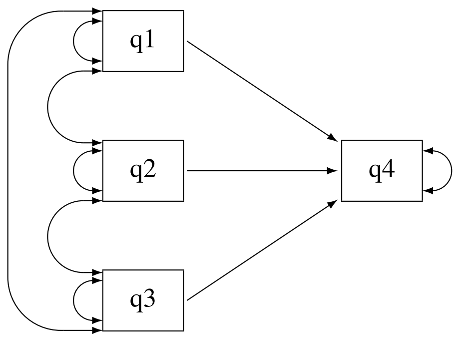
In the beginning of the current example, you use the following path diagram to represent the multiple regression model for the sales data:



The following statements specify the multiple regression model:

```
proc calis data=sales;
   path   q1 q2 q3 --->  q4;
   run;
```

You do not represent the covariances and variances among the exogenous variables explicitly in the path diagram, nor in the PATH statement specification. However, PROC CALIS generates them as free parameters by default. Some researchers might prefer to represent the exogenous variances and covariances explicitly in the path diagram, as shown in the following path diagram:



In the path diagram, there are three single-head arrows and seven double-headed arrows. These 10 paths represent the 10 parameters in the covariance structure model. To represent all these parameters in the PATH model specification, you can use the following statements:

```
proc calis data=sales;
   path   q1   --->  q4  ,
          q2   --->  q4  ,
          q3   --->  q4  ,
          q1   <-->  q1  ,
          q2   <-->  q2  ,
          q3   <-->  q3  ,
          q1   <-->  q2  ,
          q2   <-->  q3  ,
          q1   <-->  q3  ,
          q4   <-->  q4  ;
run;
```

The first three path entries in the PATH statement reflect the single-headed paths in the path diagram. The next six path entries in the PATH statement reflect the double-headed paths among the exogenous variables q1–q3 in the path diagram. The last path entry in the PATH statement reflects the double-headed path attached to the endogenous variable q4 in the path diagram. With this specification, the parameter estimates for the multiple regression model are all shown in Output 26.7.14.

Output 26.7.14 Path Estimates of the Multiple Regression Model for the Sales Data

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
q1	--->	q4	_Parm01	0.55980	0.64938	0.86205
q2	--->	q4	_Parm02	0.58946	0.84558	0.69711
q3	--->	q4	_Parm03	0.88290	0.51635	1.70988
q1	<-->	q1	_Parm04	0.33830	0.13269	2.54951
q2	<-->	q2	_Parm05	0.22466	0.08812	2.54951
q3	<-->	q3	_Parm06	0.60633	0.23782	2.54951
q1	<-->	q2	_Parm07	0.0001978	0.07646	0.00259
q2	<-->	q3	_Parm08	0.12653	0.10821	1.16931
q1	<-->	q3	_Parm09	0.03610	0.12601	0.28649
q4	<-->	q4	_Parm10	1.84128	0.72221	2.54951

These estimates are the same as those in Output 26.7.2, where the estimates are shown in three different tables, instead of in one table for all paths as in Output 26.7.14.

Sometimes, specification of some single-headed and double-headed paths can become very laborious. Fortunately, PROC CALIS provides shorthand notation for the PATH statement to make the specification more efficient. For example, a more concise way to specify the preceding multiple regression model is shown in the following statements:

```
proc calis data=sales;
   path   q1 q2 q3  --->  q4  ,
          <-->  [q1-q3]  ,
          <-->  q4  ;
run;
```

The first path entry q1 q2 q3 ---> q4 in the PATH statement represents the three single-headed arrows in the path diagram. The second path entry <--> [q1-q3] generates the variances and covariances for the set of variables specified in the rectangular brackets. The last path entry represents the error variance of q4. Consequently, expanding the preceding shorthand specification generates the following specification:

```
proc calis data=sales;
   path   q1   --->  q4  ,
          q2   --->  q4  ,
          q3   --->  q4  ,
          q1   <-->  q1  ,
          q2   <-->  q1  ,
          q2   <-->  q2  ,
          q3   <-->  q1  ,
          q3   <-->  q2  ,
          q3   <-->  q3  ,
          q4   <-->  q4  ;
run;
```

Notice that the third through ninth path entries correspond to the lower triangular elements of the covariance matrix for q1–q3.

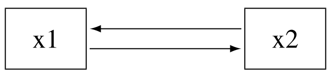
CAUTION: The double-headed path specification does *not* represent a reciprocal relationship. That is, the following statement specifies the covariance between x2 and x1:

```
path    x2 <--> x1,
```

But the following statement specifies that x2 and x1 have reciprocal causal effects:

```
path    x2 <--- x1,
        x1 ---> x2;
```

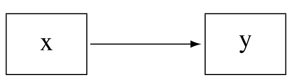
The reciprocal causal effects specification reflects the following path diagram:



Example 26.8: Measurement Error Models

In this example, you use PROC CALIS to fit some measurement error models. You use latent variables to define "true" scores variables that are measured without errors. You constrain parameters by using parameter names or fixed values in the PATH model specification.

Consider a simple linear regression model with dependent variable y and predictor variable x. The path diagram for this simple linear regression model is depicted as follows:



Suppose you have the following SAS data set for the regression analysis of y on x:

```
data measures;
   input x y @@;
   datalines;
 7.91736    13.8673    6.10807    11.7966    6.94139    12.2174
 7.61290    12.9761    6.77190    11.6356    6.33328    11.7732
 7.60608    12.8040    6.65642    12.8866    6.26643    11.9382
 7.32266    13.2590    5.76977    10.7654    5.62881    11.5041
 7.57418    13.2502    7.17305    13.3416    8.23123    13.9876
 7.17199    13.1750    8.04604    14.5968    5.77692    11.5077
 5.72741    11.3299    6.66033    12.5159    7.14944    12.4988
 7.51832    12.3588    5.48877    11.2211    7.50323    13.3735
 7.15814    13.1556    7.35485    13.8457    8.91648    14.4929
 5.37445     9.6366    6.00419    11.7654    6.89546    13.1493
;
```

This data set contains 30 observations for the x and y variables. You can fit the simple linear regression model to the measures data by the PATH model specification of PROC CALIS, as shown in the following statements:

```
proc calis data=measures;
   path
      x ---> y;
   run;
```

Output 26.8.1 shows that the regression coefficient estimate (denoted as _Parm1 in the PATH List) is 1.1511 (standard error = 0.1002).

Output 26.8.1 Estimates of the Linear Regression Model for the Measures Data

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
x	--->	y	_Parm1	1.15112	0.10016	11.49241

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Exogenous	x	_Add1	0.79962	0.20999	3.80789
Error	y	_Add2	0.23265	0.06110	3.80789

You can also do the simple linear regression by PROC REG by the following statement:

```
proc reg data=measures;
   model y = x;
run;
```

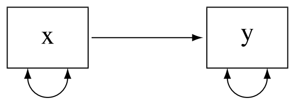
Output 26.8.2 shows that PROC REG essentially gives the same regression coefficient estimate with a similar standard error estimate. The discrepancy in the standard error estimates produced by the two procedures is due to the different variance divisors in computing standard errors in the two procedures. But the discrepancy is negligible when the sample size becomes large.

Output 26.8.2 PROC REG Estimates of the Linear Regression Model for the Measures Data

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	4.62455	0.70790	6.53	<.0001
x	1	1.15112	0.10194	11.29	<.0001

There are two main differences between PROC CALIS and PROC REG regarding the parameter estimation results. First, PROC CALIS does not give the estimate of the intercept because by default PROC CALIS analyzes only the covariance structures. Therefore, it does not estimate the intercept. To obtain the intercept estimate, you can add the MEANSTR option in the PROC CALIS statement, as is shown in Example 26.9. Second, in Output 26.8.1 of PROC CALIS, the variance estimate of x and the error variance estimate of y are shown. The corresponding results are not shown as parameter estimates in the PROC REG results. In PROC CALIS, these two variances are model parameters in covariance structure analysis. PROC CALIS adds these variances as default parameters. You can also represent these two variance parameters by double-headed arrows in the path diagram, as shown in the following:



The two double headed-arrows attached to x and y represent the variances. Although it is not necessary to specify these default parameters, you can use the PVAR statement to specify them explicitly, as shown in the following statements:

```
proc calis data=measures meanstr;
   path
      x ---> y;
   pvar
      x y;
run;
```

In the PROC CALIS statement, you specify the MEANSTR option to request the analysis of mean structures together with covariance structures. Output 26.8.3 shows the estimation results.

Output 26.8.3 Estimates of the Measurement Error Model with Error in x

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
x	--->	y	_Parm1	1.15112	0.10016	11.49241

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Exogenous	x	_Parm2	0.79962	0.20999	3.80789
Error	y	_Parm3	0.23265	0.06110	3.80789

Means and Intercepts

Type	Variable	Parameter	Estimate	Standard Error	t Value
Intercept	y	_Add1	4.62455	0.69578	6.64658
Mean	x	_Add2	6.88865	0.16605	41.48504

The regression coefficient estimate and the variance estimates are the same as those in Output 26.8.1. However, in Output 26.8.3, there is an additional table for the mean and intercept estimates. The intercept estimate for y is 4.6246 (standard error=0.6958), which match closely to the results obtained from PROC REG, as shown in Output 26.8.2.

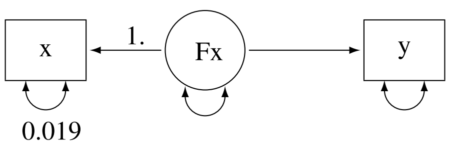
Measurement Error in x

PROC CALIS can also handle more complicated regression situations where the variables are measured with errors. This is beyond the application of PROC REG.

Suppose that the predictor variable x is measured with error and from prior studies you know that the size of the measurement error variance is about 0.019. You can use PROC CALIS to incorporate this information into the model. First, think of the measured variable x as composed of two components: one component is the "true" score measure Fx and the other is the measurement error e1. Both of these components are not observed (that is, latent) but they sum up to yield x. That is,

$$x = \mathrm{Fx} + \mathrm{e1}$$

Because x is contaminated with measurement error, what you are interested in knowing is the regression effect of the true score Fx on x. The following path diagram represents this regression scenario:



In path diagrams, latent variables are usually represented by circles or ovals, while observed variables are represented by rectangles. In the current path diagram, Fx is a latent variable and is represented by a circle. The other two variables are observed variables and are represented by rectangles. There are five arrows in the path diagram. Two of them are single-headed arrows that represent functional relationships, while the other three are double-headed arrows that represent variances or error variances. Two paths are labeled with fixed values. The path effect from Fx to x is fixed at 1, as assumed in the measurement error model. The error variance for measuring x is fixed at 0.019 due to the prior knowledge about the measurement error variance. The remaining three arrows represent free parameters in the model: the regression coefficient of y on Fx, the variance of Fx, and the error variance of y. The following statements specify the model for this path diagram:

```
proc calis data=measures;
   path
      x  <--- Fx   = 1.,
      Fx ---> y;
   pvar
      x  = 0.019,
      Fx, y;
run;
```

You specify all the single-headed paths in the PATH statement and all the double-headed arrows in the PVAR statement. For paths with fixed values, you put the equality at the back of the specifications to tell PROC CALIS about the fixed values. For example, the path coefficient in the path x <--- Fx is fixed at 1 and the error variance for x is fixed at 0.019. All other specifications represent unnamed free parameters in the model.

Output 26.8.4 shows the estimation results. The effect of Fx on y is 1.1791 (standard error=0.1029). This effect is slightly greater than the corresponding effect (1.1511) of x on y in the preceding model where the measurement error of x has not been taken into account, as shown in Output 26.8.3.

Output 26.8.4 Estimates of the Measurement Error Model with Error in x

PATH List

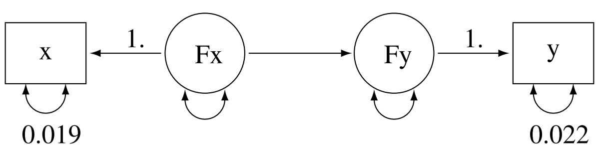
--------Path--------			Parameter	Estimate	Standard Error	t Value
x	<---	Fx		1.00000		
Fx	--->	y	_Parm1	1.17914	0.10288	11.46153

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	x		0.01900		
Exogenous	Fx	_Parm2	0.78062	0.20999	3.71741
Error	y	_Parm3	0.20686	0.06126	3.37667

Measurement Errors in x and y

Measurement error can occur in the y variable too. Suppose that both x and y are measured with errors. From prior studies, the measurement error variance of x is known to be 0.019 (as in the preceding modeling scenario) and the measurement error variance of y is known to be 0.022. The following path diagram represents the current modeling scenario:



In the current path diagram the true score variable Fy and its measurement indicator y have the same kind of relationship as the relationship between the true score variable Fx and its measurement indicator x in the previous description. The error variance for measuring y is treated as a known constant 0.022. You can transcribe this path diagram easily to the following PROC CALIS specification:

```
proc calis data=measures;
   path
      x  <--- Fx  = 1.,
      Fx ---> Fy    ,
      Fy ---> y   = 1.;
   pvar
      x  = 0.019,
      y  = 0.022,
      Fx Fy;
run;
```

Again, you specify all the single-headed paths in the PATH statement and the double-headed paths in the PVAR statement. You provide the fixed parameter values by appending the required equalities after the individual specifications.

Output 26.8.5 shows the estimation results of the model with measurement errors in both x and y. The effect of Fx on Fy is 1.1791 (standard error=0.1029). This is essentially the same effect of Fx on y as in the preceding measurement model in which no measurement error in y is assumed.

Output 26.8.5 Estimates of the Measurement Error Model with Errors in x and y

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
x	<---	Fx		1.00000		
Fx	--->	Fy	_Parm1	1.17914	0.10288	11.46153
Fy	--->	y		1.00000		

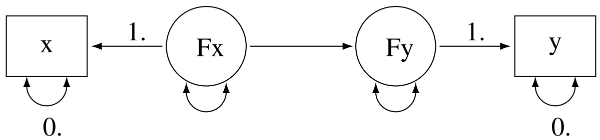
Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	x		0.01900		
	y		0.02200		
Exogenous	Fx	_Parm2	0.78062	0.20999	3.71741
Error	Fy	_Parm3	0.18486	0.06126	3.01755

The estimated error variance for Fy in the current model is 0.1849 and the measurement error variance of y is fixed at 0.022, as shown in the last table of Output 26.8.5. The sum is 0.2069, which is the same amount of error variance for y in the preceding model with measurement error assumed only in x. Hence, the assumption of the measurement error in y does not change the structural effect of Fx on y (same amount of effect Fx on Fy, which is 1.1791). It only changes the variance components of y. In the preceding model with measurement error assumed only in x, the total error variance in y is 0.2069. In the current model, this total error variance is partitioned into the measurement error variance (which is fixed at 0.022) and the error variance in the regression on Fx (which is estimated at 0.1849).

Linear Regression Model as a Special Case of Structural Equation Model

By using the current measurement error model as an illustration, it is easy to see that the structural equation model is a more general model that includes the linear regression model as a special case. If you restrict the measurement error variances in x and y to zero, the measurement error model (which represents the structural equation model in this example) reduces to the linear regression model. That is, the path diagram becomes:



You can then specify the PATH model by the following statements:

```
proc calis data=measures;
   path
      x  <--- Fx  = 1.,
      Fx ---> Fy    ,
      Fy ---> y   = 1.;
   pvar
      x  = 0.,
      y  = 0.,
      Fx Fy;
run;
```

Output 26.8.6 shows the estimation results of this measurement error model with zero measurement errors. The estimate of the regression coefficient is 1.1511, which is essentially the same result as in Output 26.8.2 by using PROC REG.

Output 26.8.6 Estimates of the Measurement Error Model with Zero Measurement Errors

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
x	<---	Fx		1.00000		
Fx	--->	Fy	_Parm1	1.15112	0.10016	11.49241
Fy	--->	y		1.00000		

Variance Parameters

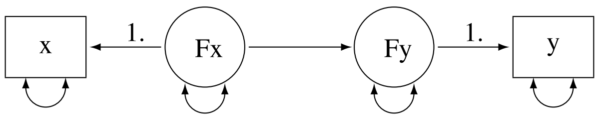
Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	x		0		
	y		0		
Exogenous	Fx	_Parm2	0.79962	0.20999	3.80789
Error	Fy	_Parm3	0.23265	0.06110	3.80789

This example shows that you can apply PROC CALIS to fit measurement error models. You treat true scores variables as latent variables in the structural equation model. The linear regression model is a special case of the structural equation model (or measurement error model) where measurement error variances are assumed to be zero. Structural equation modeling by PROC CALIS is not limited to this simple modeling scenario. PROC CALIS can treat more complicated measurement error models. In Example 26.9 and Example 26.10, you fit measurement error models with parameter constraints and with more than one predictor. You can also fit measurement error models with correlated errors.

Example 26.9: Testing Specific Measurement Error Models

In Example 26.8, you used the PATH modeling language of PROC CALIS to fit some basic measurement error models. In this example, you continue to fit the same kind of measurement error models but you restrict some model parameters to test some specific hypotheses.

This example uses the same data set as is used in Example 26.8. This data set contains 30 observations for the x and y variables. The general measurement error model with measurement errors in both x and y is shown in the following path diagram:

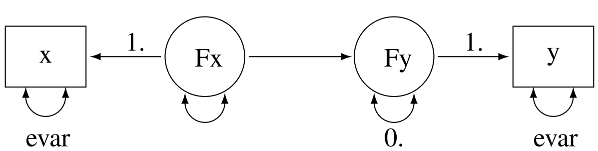


In the path diagram, two paths are fixed with a path coefficient of 1. They are required in the model for representing the relationships between true scores (latent) and measured indicators (observed). In Example 26.8, you consider several different modeling scenarios, all of which require you to make some parameter restrictions to estimate the models. You fix the measurement error variances to certain values that are based on prior knowledge or studies. Without those fixed error variances, those models would have been overparameterized and the parameters would not have been estimable.

For example, in the current path diagram, five of the single- or double-headed paths are not labeled with fixed numbers. Each of these paths represents a free parameter in the model. However, in the covariance structure model analysis, you fit these free parameters to the three nonredundant elements of the sample covariance matrix, which is a 2×2 symmetric matrix. Hence, to have an identified model, you can at most have three free parameters in your covariance structure model. However, the path diagram shows that you have five free parameters in the model. You must introduce additional parameter constraints to make the model identified.

If you do not have prior knowledge about the measurement error variances (as those described in Example 26.8), then you might need to make some educated guesses about how to restrict the overparameterized model. For example, if x and y are of the same kind of measurements, perhaps you can assume that they have an equal amount of measurement error variance. Furthermore, if the measurement errors have been taken into account, in some physical science studies you might be able to assume that the relationship between the true scores Fx and Fy is almost deterministic, resulting in a near zero error variance of Fy.

The assumptions here are not comparable to prior knowledge or studies about the measurement error variances. If you suppose they are reasonable enough in a particular field, you can use these assumptions to give you an identified model to work with (at least as an exploratory study) when the required prior knowledge is lacking. The following path diagram incorporates these two assumptions in the measurement error model:



In the path diagram, you use evar to denote the error variances of x and y. This implicitly constrains the two error variances to be equal. The error variance of Fy is labeled zero, indicating a fixed parameter value and a deterministic relationship between x and y. You can transcribe this path diagram into the following PATH modeling specification:

```
proc calis data=measures;
   path
      x  <--- Fx  = 1.,
      Fx ---> Fy   ,
      Fy ---> y   = 1.;
   pvar
      x  = evar,
      y  = evar,
      Fy = 0.,
      Fx;
run;
```

In the PVAR statement, you specify the same parameter name evar for the error variances of x and y. This way their estimates are constrained to be the same in the estimation. In addition, the error variance for Fy is fixed at zero, which reflects the "near-deterministic" assumption about the relationship between Fx and Fy. These two assumptions effectively reduce the overparameterized model by two parameters so that the new model is just-identified and estimable.

Output 26.9.1 shows the estimation results. The estimated effect of Fx on Fy is 1.3028 (standard error = 0.1134). The measurement error variances for x and y are both estimated at 0.0931 (standard error = 0.0244).

Output 26.9.1 Estimates of the Measurement Error Model with Equal Measurement Error Variances

PATH List

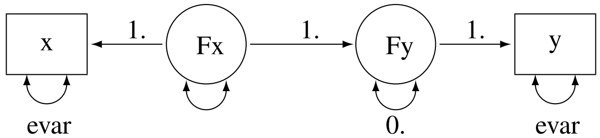
--------Path--------			Parameter	Estimate	Standard Error	t Value
x	<---	Fx		1.00000		
Fx	--->	Fy	_Parm1	1.30275	0.11336	11.49241
Fy	--->	y		1.00000		

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	x	evar	0.09307	0.02444	3.80789
	y	evar	0.09307	0.02444	3.80789
	Fy		0		
Exogenous	Fx	_Parm2	0.70655	0.20962	3.37057

Testing the Effect of Fx on Fy

Suppose you are interested in testing the hypothesis that the effect of Fx on Fy (that is, the regression slope) is 1. The following path diagram represents the model under the hypothesis:



Now you label the path from Fx to Fy with a fixed constant 1, which reflects the hypothesis you want to test. You can transcribe the current path diagram easily into the following PROC CALIS specification:

```
proc calis data=measures;
   path
      x  <--- Fx  = 1.,
      Fx ---> Fy  = 1.,   /* Testing a fixed constant effect */
      Fy ---> y   = 1.;
   pvar
      x  = evar,
      y  = evar,
      Fy = 0.,
      Fx;
run;
```

Output 26.9.2 shows the model fit chi-square statistic. The model fit chi-square test here essentially is a test of the null hypothesis of the constant effect at 1 because the alternative hypothesis is a saturated model. The chi-square value is 8.1844 (df=1, p=.0042), which is statistically significant. This means that the hypothesis of constant effect at 1 is rejected.

Output 26.9.2 Fit Summary for Testing Constant Effect

Fit Summary

Chi-Square	8.1844
Chi-Square DF	1
Pr > Chi-Square	0.0042

Output 26.9.3 shows the estimates under this restricted model. In the first table of Output 26.9.3, all path effects or coefficients are fixed at 1. In the second tale of Output 26.9.3, estimates of the error variances are 0.1255 (standard error = 0.0330) for both x and y. The error variance of Fy is a fixed zero, as required in the hypothesis. The variance estimate of Fx is 0.9205 (standard error = 0.2587).

Output 26.9.3 Estimates of Constant Effect Measurement Error Model

PATH List

--------Path--------			Estimate	Standard Error	t Value
x	<---	Fx	1.00000		
Fx	--->	Fy	1.00000		
Fy	--->	y	1.00000		

Output 26.9.3 *continued*

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	x	evar	0.12545	0.03295	3.80789
	y	evar	0.12545	0.03295	3.80789
	Fy		0		
Exogenous	Fx	_Parm1	0.92046	0.25872	3.55771

Testing a Zero Intercept

Suppose you are interested in testing the hypothesis that the intercept for the regression of Fy on Fx is zero, while the regression effect is freely estimated. Because the intercept parameter belongs to the mean structures, you need to specify this parameter in PROC CALIS to test the hypothesis.

There are two ways to include the mean structure analysis. First, you can include the MEANSTR option in the PROC CALIS statement. Alternatively, you can use the MEAN statement to specify the means and intercepts in the model. The following statements specify the model under the zero intercept hypothesis:

```
proc calis data=measures;
   path
      x  <--- Fx  = 1.,
      Fx ---> Fy  ,        /* regression effect is freely estimated */
      Fy ---> y   = 1.;
   pvar
      x  = evar,
      y  = evar,
      Fy = 0.,
      Fx;
   mean
      x y = 0. 0., /* Intercepts are zero in the measurement error model */
      Fy  = 0.,    /* Fixed to zero under the hypothesis */
      Fx;          /* Mean of Fx is freely estimated */
run;
```

In the PATH statement, the regression effect of Fx on Fy is freely estimated. In the MEAN statement, you specify the means or intercepts of the variables. Each variable in your measurement error model has either a mean or an intercept (but not both) to specify. If a variable is exogenous (independent), you can specify its mean in the MEAN statement. Otherwise, you can specify its intercept in the MEAN statement. Variables x and y in the measurement error model are both endogenous. They are measured indicators of their corresponding true scores Fx and Fy. Under the measurement error model, their intercepts are fixed zeros. The intercept for Fy is zero under the current hypothesized model. The mean of Fx is freely estimated under the model. This parameter is specified in the MEAN statement but is not named.

Output 26.9.4 shows the model fit chi-square statistic. The chi-square value is 10.5397 (df=1, p=.0012), which is statistically significant. This means that the zero intercept hypothesis for the regression of Fy on Fx is rejected.

Output 26.9.4 Fit Summary for Testing Zero Intercept

Fit Summary

Chi-Square	10.5397
Chi-Square DF	1
Pr > Chi-Square	0.0012

Output 26.9.5 shows the estimates under the hypothesized model. The effect of Fx on Fy is 1.8169 (standard error = 0.0206). In the last table of Output 26.9.5, the estimate of the mean of Fx is 6.9048 (standard error = 0.1388). The intercepts for all other variables are fixed at zero under the hypothesized model.

Output 26.9.5 Estimates of the Zero Intercept Measurement Error Model

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
x	<---	Fx		1.00000		
Fx	--->	Fy	_Parm1	1.81689	0.02055	88.42473
Fy	--->	y		1.00000		

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	x	evar	0.13684	0.03594	3.80789
	y	evar	0.13684	0.03594	3.80789
	Fy		0		
Exogenous	Fx	_Parm2	0.42280	0.11990	3.52614

Means and Intercepts

Type	Variable	Parameter	Estimate	Standard Error	t Value
Intercept	x		0		
	y		0		
	Fy		0		
Mean	Fx	_Parm3	6.90483	0.13881	49.74314

Measurement Model with Means and Intercepts Freely Estimated

In the preceding model, you fit a restricted regression model with a zero intercept. You reject the null hypothesis and conclude that this intercept is significantly different from zero. The alternative hypothesis is a saturated model with the intercept freely estimated. The model under the alternative hypothesis is specified in the following statements:

```
proc calis data=measures;
   path
      x  <--- Fx  = 1.,
      Fx ---> Fy  ,
      Fy ---> y   = 1.;
   pvar
      x  = evar,
      y  = evar,
      Fy = 0.,
      Fx;
   mean
      x y = 0. 0.,
      Fy Fx;
run;
```

Output 26.9.6 shows that model fit chi-square statistic is zero. This is expected because you are fitting a measurement error model with saturated mean and covariance structures.

Output 26.9.6 Fit Summary of the Saturated Measurement Model with Mean Structures

Fit Summary

Chi-Square	0.0000
Chi-Square DF	0
Pr > Chi-Square	.

Output 26.9.7 shows the estimates under the measurement model with saturated mean and covariance structures. The effect of Fx on Fy is 1.3028 (standard error=0.1134), which is considerably smaller than the corresponding estimate in the restricted model with zero intercept, as shown in Output 26.9.5. The intercept estimate of Fy is 3.5800 (standard error = 0.7864), with a significant t value of 4.55.

Output 26.9.7 Estimates of the Saturated Measurement Model with Mean Structures

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
x	<---	Fx		1.00000		
Fx	--->	Fy	_Parm1	1.30275	0.11336	11.49241
Fy	--->	y		1.00000		

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	x	evar	0.09307	0.02444	3.80789
	y	evar	0.09307	0.02444	3.80789
	Fy		0		
Exogenous	Fx	_Parm2	0.70655	0.20962	3.37057

Output 26.9.7 *continued*

Means and Intercepts

Type	Variable	Parameter	Estimate	Standard Error	t Value
Intercept	x		0		
	y		0		
	Fy	_Parm3	3.57998	0.78641	4.55234
Mean	Fx	_Parm4	6.88865	0.16605	41.48504

In this example, you fit some measurement error models with some parameter constraints that reflect the hypothesized models of interest. You can set equality constraints by simply providing the same parameter names in the PATH model specification of PROC CALIS. You can also fix parameters to constants. In the MEAN statement, you can specify the intercepts and means of the variables in the measurement error models. You can apply all these techniques to more complicated measurement error models with multiple predictors, as shown in Example 26.10, where you also fit measurement error models with correlated errors.

Example 26.10: Measurement Error Models with Multiple Predictors

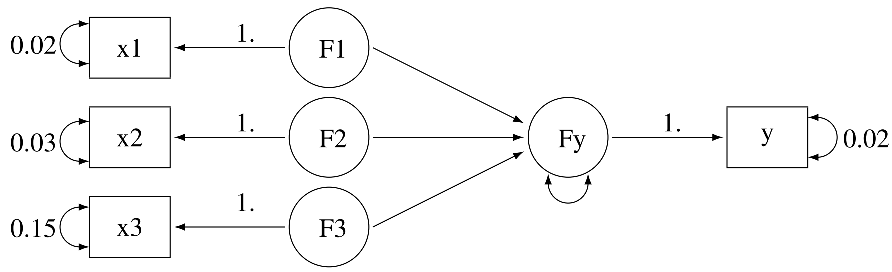
In Example 26.8 and Example 26.9, you fit various measurement error models with only one predictor. This example illustrates the case in which you have more than one predictor, all measured with errors. The measurement errors might also be correlated.

The data from 37 observations are summarized in a covariance matrix as shown in the following SAS DATA step:

```
data multiple(type=cov);
   input _type_ $ 1-4  _name_ $ 6-8 @10 y x1 x2 x3;
   datalines;
mean      0.93   1.33   1.34   4.11
cov   y   1.31    .      .      .
cov   x1  1.24   1.42    .      .
cov   x2  0.21   0.18   1.15    .
cov   x3  3.91   4.21   0.58  14.11
;
```

In this data set, four variables are measured. Variables x1–x3 are predictors of y. Instead of the raw data, you can input the sample covariance matrix in the form of a SAS data set for PROC CALIS to analyze.

You assume all of these variables in the data set are measured with errors. From prior studies, you establish the knowledge about the measurement errors of these variables. You create the true score counterparts for each of these variables in the same manner as you do in Example 26.8 and Example 26.9. The following path diagram represents your measurement error model for the data:



In the path diagram, variables F1–F3 and Fy are latent variables that represent the true score for the measured indicators x1–x3 and y, respectively. All paths from the true scores to the corresponding measured indicators are labeled with the fixed constant 1, as required by the measurement model. Each measured indicator is attached with a double-headed arrow that indicates the error variance. Because you have knowledge about these measurement error variances, you put fixed constant values adjacent to these double-headed arrows. For example, the measurement error variance of y is 0.02 and the measurement error variance of x3 is 0.15. The path digram also indicates that the paths from F1–F3 to Fy and the error variance for Fy are free parameters to estimate in the model.

Notice that for brevity the variances and covariances among the three exogenous true score variables F1–F3 are not represented in the path diagram. These six variance and covariance parameters could have been represented by double-headed arrows in the path diagram. However, because PROC CALIS always assumes the exogenous variances and covariances as default model parameters, this information is not represented to reduce clutter in the path diagram.

You can transcribe the path diagram easily to the following PATH model specification:

```
proc calis data=multiple nobs=37;
   path
      Fy  <---   F1 F2 F3,
      F1  --->   x1   = 1.,
      F2  --->   x2   = 1.,
      F3  --->   x3   = 1.,
      Fy  --->   y    = 1.;
   pvar
      x1 x2 x3 y = .02 .03 .15 .02,
      Fy;
run;
```

In the first entry of the PATH statement, you specify that F1–F3 predicts Fy. In the next four path entries you specify the measurement model for the true scores and how they are related to the observed variables. In the PVAR statement, you specify all the known measurement error variances for the observed variables. They are all fixed constants in the model. In the last entry in the PVAR statement, you specify the error variance of Fy as a free (unnamed) parameter. You could have omitted this entry because error variances for all endogenous variables in the PATH model are free parameters by default. Setting these default parameters explicitly as free parameters would not affect model fitting.

Output 26.10.1 shows the parameter estimates of the model. The path coefficient or effect from F2 to Fy is not significant, while the other two path coefficients are at least marginally significant.

Output 26.10.1 Parameter Estimates of the Measurement Model with Multiple Predictors

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
Fy	<---	F1	_Parm1	0.46507	0.22682	2.05035
Fy	<---	F2	_Parm2	0.04123	0.07069	0.58323
Fy	<---	F3	_Parm3	0.13812	0.07175	1.92490
F1	--->	x1		1.00000		
F2	--->	x2		1.00000		
F3	--->	x3		1.00000		
Fy	--->	y		1.00000		

Variance Parameters

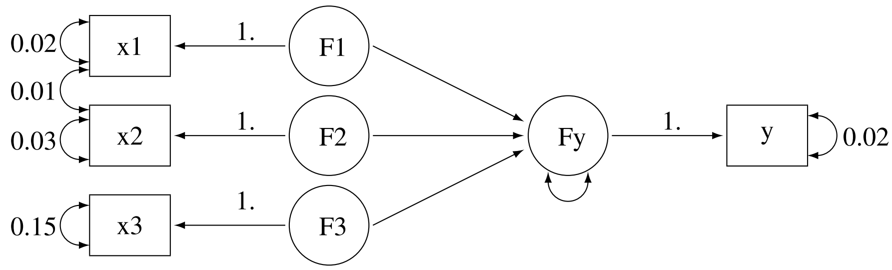
Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	x1		0.02000		
	x2		0.03000		
	x3		0.15000		
	y		0.02000		
	Fy	_Parm4	0.16461	0.04522	3.64028
Exogenous	F1	_Add1	1.40000	0.33470	4.18289
	F2	_Add2	1.12000	0.27106	4.13196
	F3	_Add3	13.96000	3.32576	4.19754

Covariances Among Exogenous Variables

Var1	Var2	Parameter	Estimate	Standard Error	t Value
F2	F1	_Add4	0.18000	0.21508	0.83688
F3	F1	_Add5	4.21000	1.02416	4.11070
F3	F2	_Add6	0.58000	0.67829	0.85509

The second table of Output 26.10.1 shows the variance estimates. As specified in the model, all measurement error variances for the observed variables are fixed constants. The error variance of Fy is 0.1646 (standard error =0.0452). Although you do not specify them in the PATH model specification, variances of F1–F3 are free parameters in the model. The second table of Output 26.10.1 shows their estimates. The last table of Output 26.10.2 shows the covariances among the latent true scores. Only the covariance between F3 and F1 is significant.

PROC CALIS not only can handle measurement error variance with multiple true score predictors, but it also can handle correlated errors. Suppose that the measurement errors for variables x1 and x2 are correlated. From prior studies, you determine that their covariance is 0.01. The path diagram with this new piece of information added is shown in the following:



In the path diagram, the double-headed arrow that connects x1 and x2 represents the covariance between the error terms for the two variables. The value attached to this double-headed arrow is 0.01, which represents a fixed constant in the model. The PATH model specification is similar to the preceding specification, with one more entry added in the PCOV statement, as shown in the following statements:

```
proc calis data=multiple nobs=37;
   path
      Fy  <---   F1 F2 F3,
      F1  --->   x1   = 1.,
      F2  --->   x2   = 1.,
      F3  --->   x3   = 1.,
      Fy  --->   y    = 1.;
   pvar
      x1 x2 x3 y = .02 .03 .15 .02,
      Fy;
   pcov
      x1 x2 = 0.01;
run;
```

Except for the PCOV statement specification, everything else is the same as in the preceding specification. In the PCOV statement, you can specify covariance or error covariances between exogenous or endogenous variables. In the current model, because both x1 and x2 are endogenous in the model, the specification is for their error covariance, which is fixed at 0.01 as required.

Output 26.10.2 shows the parameter estimates of the measurement model with correlated errors. The estimates do not change much from the preceding analysis in which correlated errors is not assumed. Perhaps the correlation between the errors in the current model is so small that it is ignorable. The last table in Output 26.10.2 shows the covariance estimates among errors. This table is unique to the current model. It shows that the measurement errors for x1 and x2 have a covariance of 0.01, which is treated as a fixed constant in the current model.

Output 26.10.2 Parameter Estimates of the Measurement Model with Multiple Predictors: Correlated Errors

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
Fy	<---	F1	_Parm1	0.46839	0.22695	2.06386
Fy	<---	F2	_Parm2	0.04549	0.07074	0.64306
Fy	<---	F3	_Parm3	0.13694	0.07194	1.90351
F1	--->	x1		1.00000		
F2	--->	x2		1.00000		
F3	--->	x3		1.00000		
Fy	--->	y		1.00000		

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	x1		0.02000		
	x2		0.03000		
	x3		0.15000		
	y		0.02000		
	Fy	_Parm4	0.16421	0.04523	3.63046
Exogenous	F1	_Add1	1.40000	0.33470	4.18289
	F2	_Add2	1.12000	0.27106	4.13196
	F3	_Add3	13.96000	3.32576	4.19754

Covariances Among Exogenous Variables

Var1	Var2	Parameter	Estimate	Standard Error	t Value
F2	F1	_Add4	0.17000	0.21508	0.79039
F3	F1	_Add5	4.21000	1.02416	4.11070
F3	F2	_Add6	0.58000	0.67829	0.85509

Covariances Among Errors

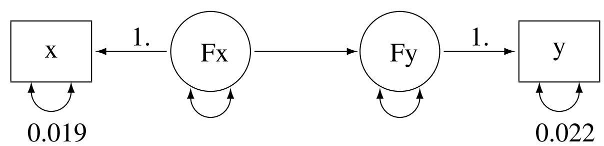
Error of	Error of	Estimate	Standard Error	t Value
x1	x2	0.01000		

This example shows how you can use PROC CALIS to fit measurement error models with multiple true score predictors. You can also fit models with correlated errors. The model specification tool is the PATH modeling language, which ties closely to the path diagram representations.

However, some researchers might prefer to use linear equations to represent the measurement error models. PROC CALIS provides the LINEQS modeling language for specifying the measurement error models, or mean and covariance structure models in general. Example 26.11 illustrates the LINEQS model specification of the measurement error models.

Example 26.11: Measurement Error Models Specified As Linear Equations

In Example 26.8, you fit a simple measurement error model with errors in both of the predictor variable x and the outcome variable y. From prior studies, the measurement error variance of x is 0.019 and the measurement error variance of y is 0.022. You use the following path diagram to represent the model:

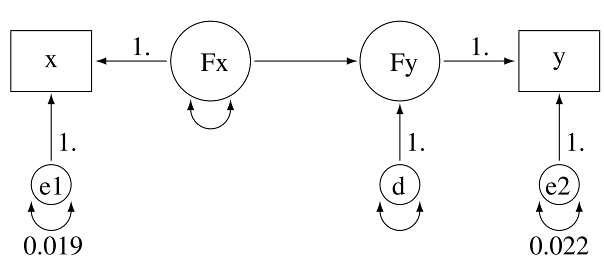


With this path diagram, you use the PATH modeling language of PROC CALIS to specify the model, as shown in the following:

```
proc calis data=measures;
   path
      x  <--- Fx  = 1.,
      Fx ---> Fy    ,
      Fy ---> y   = 1.;
   pvar
      x  = 0.019,
      y  = 0.022,
      Fx Fy;
run;
```

In the path diagram and in the PATH model specification, there are no explicit representations of the error terms in the model. You express the error variances of x, y, and Fy as partial variances of the endogenous variables. In the path diagram, you represent these partial variances by the double-headed arrows. Correspondingly, in the PATH statement of PROC CALIS, you specify these partial variances in the PVAR statement.

In practice, some researchers might prefer to express the error terms in the model explicitly. For example, with the error terms added to the preceding measurement error model, the new path diagram becomes:



In the path diagram, you add paths from error variables e1, e2, and d to the endogenous variables x, y, and Fy, respectively. All these paths from the error terms have a fixed path coefficient of 1. The error variances are represented by double-headed arrows directly attached to them. For example, the variance of e1 is fixed at 0.019, and the variance of e2 is fixed at 0.022. The variance of d, which is sometime called the disturbance, is a free unnamed parameter in the path diagram. Similarly, the variance of Fx is a free unnamed parameter in the model.

Corresponding to this new path diagram, you can use the LINEQS modeling language for specifying your model in PROC CALIS, as shown in the following statements:

```
proc calis data=measures;
   lineqs
      x   = 1. * Fx  +  e1,
      y   = 1. * Fy  +  e2,
      Fy  =    * Fx  +   d;
   variance
      e1  = 0.019,
      e2  = 0.022,
      Fx d;
run;
```

The LINEQS model specification in PROC CALIS emphasizes the linear equation input. In each of the linear equations in the LINEQS statement, you specify an endogenous variable and how it is related to other variables. An endogenous variable in the path diagram is a variable that has at least one single-headed arrow pointing to it. You need to list all endogenous variables on the left-hand side of the linear equations of the LINEQS statement. In the current model, variables x, y, and Fy are endogenous, and therefore you specify three linear equations in the LINEQS statement. The first two equations represent the measurement model for the observed variables, while the third equation represents the structural equation of the model. Notice that in the third equation, you do not specify the path coefficient that is attached to Fx. PROC CALIS treats unspecified path coefficients as free parameters. The effect of Fx on Fy is freely estimated, as required in the path diagram representation.

In the VARIANCE statement, you specify the variances of the exogenous variables in the model. The specifications in the VARIANCE statement of the LINEQS model are very similar to those in the PVAR statement of the PATH model. The main difference is the use of error variable names in the VARIANCE statement. With the LINEQS model specification, you can only specify exogenous variables in the VARIANCE statement. Hence, you must specify the error variables e1, e2, and d in the VARIANCE statement of the LINEQS model, instead of the corresponding endogenous variables x, y, and Fy in the PVAR statement of the PATH model.

Output 26.11.1 shows the parameter estimates of the LINEQS model.

Output 26.11.1 LINEQS Parameter Estimates of the Measurement Model for the Measures Data

```
                    Linear Equations

x         =     1.0000 Fx      +   1.0000 e1
y         =     1.0000 Fy      +   1.0000 e2
Fy        =     1.1791*Fx      +   1.0000 d
Std Err         0.1029 _Parm1
t Value        11.4615
```

Output 26.11.1 *continued*

Estimates for Variances of Exogenous Variables

Variable Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	e1		0.01900		
	e2		0.02200		
Latent	Fx	_Parm2	0.78062	0.20999	3.71741
Disturbance	d	_Parm3	0.18486	0.06126	3.01755

All these estimates are essentially the same as those obtained from the PATH model specification, as shown in Output 26.11.2.

Output 26.11.2 PATH Parameter Estimates of the Measurement Model for the Measures Data

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
x	<---	Fx		1.00000		
Fx	--->	Fy	_Parm1	1.17914	0.10288	11.46153
Fy	--->	y		1.00000		

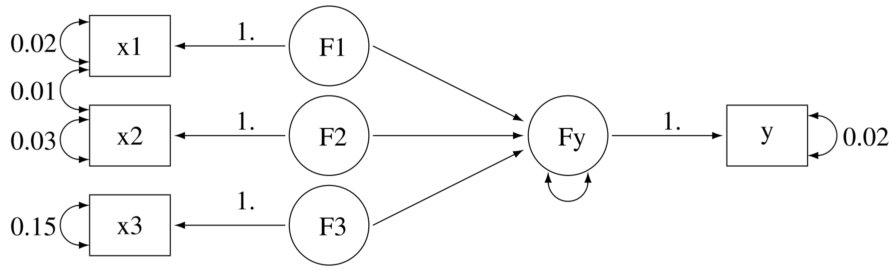
Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	x		0.01900		
	y		0.02200		
Exogenous	Fx	_Parm2	0.78062	0.20999	3.71741
Error	Fy	_Parm3	0.18486	0.06126	3.01755

You can use either the LINEQS or PATH model specification in PROC CALIS for your analysis problems. They give you the same estimation results.

So far the measurement error model is concerned with one predictor. With more predictors in the model, you might also want to model the correlated measurement errors in the x variables. You can analyze this kind of model by using the PATH model specification, as shown in Example 26.10. With measurement error terms explicitly assumed, you can also use the LINEQS model specification. This example illustrates how you can do that by using the same data set and the measurement error model with correlated errors in Example 26.10.

In the data set, you have four observed variables: x1–x3 and y. All are measured with errors, as represented by the following path diagram:

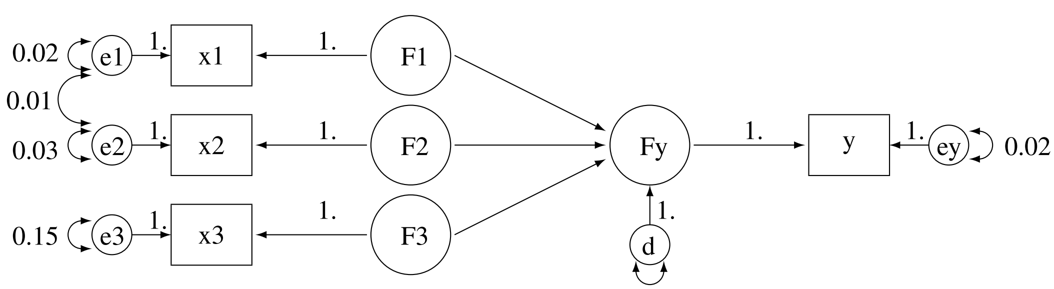


In the path diagram, F1–F3 and Fy represent true scores for the measurement indicators x1–x3 and y, respectively. You predict Fy by F1–F3, which represents the structural relationships in the model. Measurement error variances of the observed variables are treated as known and are represented by the double-headed arrows attached to the observed variables. For example, the error variance of x3 is 0.15. In addition, the error covariance between x1 and x2 is treated as known. The double-headed arrow that connects x1 and x2 represents the error covariance, and this covariance is fixed at 0.01 in the model.

You transcribe this path diagram representation into the following PATH model specification:

```
proc calis data=multiple nobs=37;
   path
      Fy  <---   F1 F2 F3,
      F1  --->   x1   = 1.,
      F2  --->   x2   = 1.,
      F3  --->   x3   = 1.,
      Fy  --->   y    = 1.;
   pvar
      x1 x2 x3 y = .02 .03 .15 .02,
      Fy;
   pcov
      x1 x2 = 0.01;
run;
```

To represent the error terms explicitly, you can add the error terms to the path diagram with some modifications, as shown in the following:



In the path diagram, you attach error variables e1–e3, ey, and d to the associated endogenous variables in the model. The error variances and covariances, which are attached to the endogenous variables directly, are now attached to the corresponding error variables. With this new path diagram, you can use the following LINEQS model specification for the model:

```
proc calis data=multiple nobs=37;
   lineqs
      Fy =     * F1 +   * F2 +   * F3 + d,
      x1 =  1. * F1 + e1,
      x2 =  1. * F2 + e2,
      x3 =  1. * F3 + e3,
       y =  1. * Fy + ey;
    variance
       e1-e3 ey = .02 .03 .15 .02,
       d;
    cov
       e1 e2 = 0.01;
run;
```

Again, in each linear equation of the LINEQS statement, you specify the functional relationship of an endogenous variable with other variables, including the error variable. The first equation is the structural equation in the model. You want to estimate the effects of F1, F2, and F3 on Fy. The error or disturbance variable is d. In the next four equations, you relate the observed variables with their true scores counterparts.

In the VARIANCE statement, you specify the error variances with reference to the error variables in the path diagram. Four of the error variances are fixed constants, as required in the model. The last specification represents a free parameter for the variance of d. The specifications in the VARIANCE statement of the LINEQS model are similar to those in the PVAR statement of the PATH model specification. The difference is that in the PATH model specification the reference variables are the endogenous variables in the PATH model, while in the LINEQS model specification the reference variables are the associated error variables.

In the COV statement, you specify the covariance between the error variables e1 and e2. Again, this is similar to the corresponding specification of the PATH model, where the same error covariance is specified as the partial covariance between x1 and x2 in the PCOV statement.

Output 26.11.3 shows the parameter estimates that result from using the LINEQS model specification. Estimates in the equations, variances, and covariances are shown respectively.

Output 26.11.3 Parameter Estimates of the Measurement Model with Multiple Predictors: LINEQS Model

```
                                  Linear Equations

Fy        =     0.4684*F1     +  0.0455*F2     +  0.1369*F3     +  1.0000 d
Std Err         0.2269 _Parm1    0.0707 _Parm2    0.0719 _Parm3
t Value         2.0639           0.6431           1.9035
x1        =     1.0000 F1     +  1.0000 e1
x2        =     1.0000 F2     +  1.0000 e2
x3        =     1.0000 F3     +  1.0000 e3
y         =     1.0000 Fy     +  1.0000 ey
```

Output 26.11.3 *continued*

Estimates for Variances of Exogenous Variables

Variable Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	e1		0.02000		
	e2		0.03000		
	e3		0.15000		
	ey		0.02000		
Disturbance	d	_Parm4	0.16421	0.04523	3.63046
Latent	F1	_Add1	1.40000	0.33470	4.18289
	F2	_Add2	1.12000	0.27106	4.13196
	F3	_Add3	13.96000	3.32576	4.19754

Covariances Among Exogenous Variables

Var1	Var2	Parameter	Estimate	Standard Error	t Value
e1	e2		0.01000		
F2	F1	_Add4	0.17000	0.21508	0.79039
F3	F1	_Add5	4.21000	1.02416	4.11070
F3	F2	_Add6	0.58000	0.67829	0.85509

Output 26.11.4 shows the parameter estimates that result from using the PATH model specification. The estimates in the path list shown in Output 26.11.4 correspond to those of the equation output in Output 26.11.3. The variance estimates in Output 26.11.4 correspond to those variance estimates of the exogenous variables of the LINEQS model, as shown in Output 26.11.3. Finally, the last two tables in Output 26.11.4 correspond to the covariance estimates among the exogenous variables of the LINEQS model, as shown in Output 26.11.3. Again, the LINEQS and PATH model specification give you exactly the same estimation results, but in different output formats.

Output 26.11.4 Parameter Estimates of the Measurement Model with Multiple Predictors: PATH Model

PATH List

--------Path--------			Parameter	Estimate	Standard Error	t Value
Fy	<---	F1	_Parm1	0.46839	0.22695	2.06386
Fy	<---	F2	_Parm2	0.04549	0.07074	0.64306
Fy	<---	F3	_Parm3	0.13694	0.07194	1.90351
F1	--->	x1		1.00000		
F2	--->	x2		1.00000		
F3	--->	x3		1.00000		
Fy	--->	y		1.00000		

Output 26.11.4 *continued*

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	x1		0.02000		
	x2		0.03000		
	x3		0.15000		
	y		0.02000		
	Fy	_Parm4	0.16421	0.04523	3.63046
Exogenous	F1	_Add1	1.40000	0.33470	4.18289
	F2	_Add2	1.12000	0.27106	4.13196
	F3	_Add3	13.96000	3.32576	4.19754

Covariances Among Exogenous Variables

Var1	Var2	Parameter	Estimate	Standard Error	t Value
F2	F1	_Add4	0.17000	0.21508	0.79039
F3	F1	_Add5	4.21000	1.02416	4.11070
F3	F2	_Add6	0.58000	0.67829	0.85509

Covariances Among Errors

Error of	Error of	Estimate	Standard Error	t Value
x1	x2	0.01000		

In this example, you fit measurement error models by using the LINEQS and PATH model specifications of PROC CALIS. The two different model specification languages give you essentially the same estimation results. The measurement models can have multiple true scores predictors and correlated errors. The measurement error models considered so far have only one measured indicator for each true score latent variable. This is usually not the case in many psychometric or sociological applications where latent factors usually have several observed indicators. The confirmatory factor model is a typical example of this kind of applications. See Example 26.12 for an application of PROC CALIS to fit confirmatory factor models. See Example 26.16 for an application of PROC CALIS to fit a general structural equation model where latent variables have more than one measured indicators.

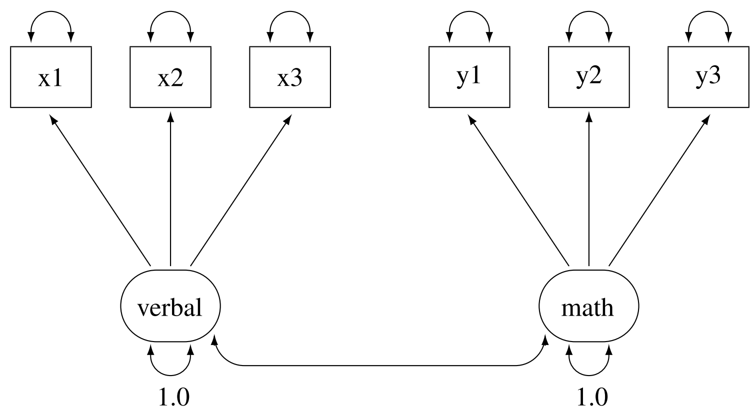
Example 26.12: Confirmatory Factor Models

This example shows how you can fit a confirmatory factor analysis model by the FACTOR modeling language. Thirty-two students take tests of their verbal and math abilities. Six tests are administered separately. Tests x1–x3 test their verbal skills and tests y1–y3 test their math skills.

The data are shown in the following DATA step:

```
data scores;
   input x1 x2 x3 y1 y2 y3;
   datalines;
 23  17  16  15  14  16
 29  26  23  22  18  19
 14  21  17  15  16  18
 20  18  17  18  21  19
 25  26  22  26  21  26
 26  19  15  16  17  17
 14  17  19   4   6   7
 12  17  18  14  16  13
 25  19  22  22  20  20
  7  12  15  10  11   8
 29  24  30  14  13  16
 28  24  29  19  19  21
 12   9  10  18  19  18
 11   8  12  15  16  16
 20  14  15  24  23  16
 26  25  21  24  23  24
 20  16  19  22  21  20
 14  19  15  17  19  23
 14  20  13  24  26  25
 29  24  24  21  20  18
 26  28  26  28  26  23
 20  23  24  22  23  22
 23  24  20  23  22  18
 14  18  17  13  16  14
 28  34  27  25  21  21
 17  12  10  14  12  16
  8   1  13  14  15  14
 22  19  19  13  11  14
 18  21  18  15  18  19
 12  12  10  13  13  16
 22  14  20  20  18  19
 29  21  22  13  17  12
;
```

Because of the unambiguous nature of the tests, you hypothesize that this is a confirmatory factor model with two factors: one is the verbal ability factor and the other is the math ability factor. You can represent such a confirmatory factor model by the following path diagram:



In the path diagram, there are two clusters of variables. One cluster is for the verbal factor and the other is for the math factor. The single-headed arrows in the path diagram represent functional relationships between factors and the observed variables. The double-headed arrows that point to single variables represent variances of the factors or error variances of the observed variables. The double-headed arrow that connect the two factors represents their covariance. All but two of these arrows are not labeled with numbers. Each of the unlabeled arrows represents a free parameter in the confirmatory factor model. You label the double-headed arrows that attach to the two factors with the constant 1. This means that the variances of the factors are fixed at 1.0 in the model.

You can specify the confirmatory factor model by the FACTOR model language of PROC CALIS, as shown in the following statements:

```
proc calis data=scores;
   factor
      verbal ---> x1-x3,
      math   ---> y1-y3;
   pvar
      verbal = 1.,
      math   = 1.;
run;
```

In each of the entry of the FACTOR statement, you specify a latent factor, followed by a list of observed variables that are functionally related to the latent factor. For example, in the first entry, the verbal factor is related to variables x1–x3, as shown by the single-headed arrows in the path diagram. In fact, all single-headed arrows in the path digram are specified in the FACTOR statement. Notice that each entry of the FACTOR statement must take the format of

```
factor_name ---> variable_list
```

You cannot reverse the arrow specification as in the following:

```
variable_list <--- factor_name
```

Nor you can have a specification such as the following:

```
variable_list ---> factor_name
```

However, you can specify the functional relationships between factors and variables in different entries. For example, you can specify the same confirmatory factor model by the following statements:

```
title "Basic Confirmatory Factor Model: Separate Path Entries";
title2 "FACTOR Model Specification";
proc calis data=scores;
   factor
      verbal ---> x1,
      verbal ---> x2,
      verbal ---> x3,
      math   ---> y1,
      math   ---> y2,
      math   ---> y3;
   pvar
      verbal = 1.,
      math   = 1.;
   fitindex  noindextype on(only)=[chisq df probchi rmsea  srmsr bentlercfi];
run;
```

In the PVAR statement, which is for the specification of variances or error variances, you fix the variances of the latent factors to 1. This completes the model specification of the confirmatory factor model, although you do not specify other arrows in the path diagram as free parameters in these statements. The reason is that in the FACTOR modeling language, the variances and covariances among factors and the error variances of the observed variables are default parameters in the confirmatory factor model. It is not necessary to specify these parameters (or the corresponding arrows in the path diagram) explicitly if they are free parameters in the model. You can also specify these free parameters explicitly without affecting the estimation. However, if these parameters (or the corresponding double-headed arrows in the path diagram) are intended to be constrained parameters or fixed values, you must specify them explicitly. For example, in the current confirmatory factor model, you must provide explicit specifications for the variances of the verbal and the math factors because these parameters are fixed at 1.

Output 26.12.1 shows the modeling information and the variables in the confirmatory factor model.

Output 26.12.1 Modeling Information and Variables of the CFA Model: Scores Data

```
                 Simple Confirmatory Factor Model
                    FACTOR Model Specification

                       The CALIS Procedure
       Covariance Structure Analysis: Model and Initial Values

                       Modeling Information

                 Data Set              WORK.SCORES
                 N Records Read        32
                 N Records Used        32
                 N Obs                 32
                 Model Type            FACTOR
                 Analysis              Covariances
```

Output 26.12.1 *continued*

```
                    Variables in the Model

              Variables    x1  x2  x3  y1  y2  y3
              Factors      verbal  math

                    Number of Variables = 6
                    Number of Factors   = 2
```

In the beginning of the output, PROC CALIS shows the data set, the number of observations, the model type, and the analysis type. The default analysis type in PROC CALIS is covariances (that is, covariance structures). If you want to analyze the correlation structures instead, you can use the CORR option in the PROC CALIS statement. Next, PROC CALIS shows the list of variables and factors in the model. As expected, the number of variables is 6 and the number of factors is 2.

Output 26.12.2 shows the initial model specifications of the confirmatory factor model.

Output 26.12.2 Initial Specification of the CFA Model: Scores Data

```
                 Initial Factor Loading Matrix

                          verbal            math

             x1                .               0
                        [_Parm1]

             x2                .               0
                        [_Parm2]

             x3                .               0
                        [_Parm3]

             y1                0               .
                                        [_Parm4]

             y2                0               .
                                        [_Parm5]

             y3                0               .
                                        [_Parm6]

                Initial Factor Covariance Matrix

                          verbal            math

           verbal         1.0000               .
                                         [_Add1]

           math                .          1.0000
                         [_Add1]
```

Output 26.12.2 *continued*

```
                    Initial Error Variances

            Variable     Parameter      Estimate

            x1           _Add2                 .
            x2           _Add3                 .
            x3           _Add4                 .
            y1           _Add5                 .
            y2           _Add6                 .
            y3           _Add7                 .

      NOTE: Parameters with prefix '_Add' are added by PROC CALIS.
```

The first table of Output 26.12.2 shows the pattern of factor loadings of the variables on the two latent factors. As expected, x1–x3 have nonzero loadings only on the verbal factor, while y1–y3 have nonzero loadings on the math factor. PROC CALIS names these free parameters automatically with the "_Parm" prefix and unique numerical suffixes. There are six parameters in the factor loading matrix with six different parameter names.

The next table of Output 26.12.2 shows the covariance matrix of the factors. The variances of the factors are fixed at one, as shown on the diagonal of the covariance matrix. The covariance between the two factors is a free parameter named _Add1. You did not specify this covariance parameter explicitly in the factor model specification. By default, PROC CALIS assumes that latent factors are correlated. Default free parameters added by PROC CALIS have the _Add prefix for their names. If you do not want to assume the covariances among the factors, you must specify zero covariances in the COV statement. For example, the following statement specifies that the math and verbal factors have zero covariance:

```
cov
  math verbal = 0.;
```

The last table of Output 26.12.2 shows the error variance parameters of the observed variables. By default PROC CALIS assumes these error variances are free parameters in the confirmatory factor model. These added parameters are named with the _Add prefix. However, as all other default parameters that are assumed by PROC CALIS, you can overwrite the default by using explicit specifications. You can specify the error variances of a confirmatory factor model explicitly in the PVAR statement. See specifications in Example 26.13.

Output 26.12.3 shows the fit summary of the confirmatory factor model for the scores data.

Output 26.12.3 Fit Summary of the CFA Model: Scores Data

```
                         Fit Summary

        Chi-Square                           9.8052
        Chi-Square DF                             8
        Pr > Chi-Square                      0.2790
        Standardized RMSR (SRMSR)            0.0571
        RMSEA Estimate                       0.0853
        Bentler Comparative Fit Index        0.9887
```

The model fit chi-square is 9.805 (df=8, p=0.279). This shows that statistically you cannot reject the confirmatory factor model for the test scores. However, the root mean square error of approximation (RMSEA) estimate is 0.0853, which is greater than the conventional 0.05 value for a good model fit. The standardized root mean square residual (SRMSR) is 0.0571, which is close to the conventional 0.05 value for a good model fit. Bentler's comparative fit index is 0.9887, which indicates a very good model fit. Overall, the model seems to be quite reasonable for the data.

Output 26.12.4 shows the loading and factor covariance estimates of the confirmatory factor model for the scores data. The first table shows the loading estimates, together with the standard error estimates and the t values. In structural equation modeling, the significance of the parameter estimates is usually inferred by comparing the t values with the critical value of a standardized normal variate (that is, the z-table). Therefore, estimates with associated (absolute) t values greater than 1.96 are significant at α=.05. In Output 26.12.4, all the t values for the loading estimates are greater than 2. This indicates that the prescribed relationships between the variables and the factors are significant.

Output 26.12.4 Loading and Factor Covariance Estimates of the CFA Model: Scores Data

```
              Factor Loading Matrix: Estimate/StdErr/t-value

                              verbal            math

              x1              5.8406               0
                              0.9962
                              5.8629
                             [_Parm1]

              x2              5.8182               0
                              0.9537
                              6.1004
                             [_Parm2]

              x3              4.6619               0
                              0.7814
                              5.9662
                             [_Parm3]

              y1                   0          5.2804
                                              0.6998
                                              7.5455
                                             [_Parm4]

              y2                   0          4.2003
                                              0.6220
                                              6.7532
                                             [_Parm5]

              y3                   0          3.7596
                                              0.6341
                                              5.9289
                                             [_Parm6]
```

Output 26.12.4 *continued*

Factor Covariance Matrix: Estimate/StdErr/t-value

	verbal	math
verbal	1.0000	0.5175 0.1429 3.6221 [_Add1]
math	0.5175 0.1429 3.6221 [_Add1]	1.0000

The second table of Output 26.12.4 shows the covariance matrix of the verbal and the math factors. Because the factor variances are fixed at one, the covariance estimate is also the correlation between the two factors. Output 26.12.4 shows that the two factors are moderately correlated with a correlation estimate of 0.5175, which is statistically significant.

Output 26.12.5 shows the estimates of the error variances. All but the error variance of y1 are significant. This suggests that y1 might have an almost perfect relationship with the math factor.

Output 26.12.5 Error Variance Estimates of the CFA Model: Scores Data

Error Variances

Variable	Parameter	Estimate	Standard Error	t Value
x1	_Add2	11.52376	4.26398	2.70259
x2	_Add3	9.14503	3.83219	2.38637
x3	_Add4	6.68169	2.59770	2.57216
y1	_Add5	0.78580	1.29440	0.60708
y2	_Add6	2.88069	1.09395	2.63329
y3	_Add7	5.15573	1.46854	3.51080

Output 26.12.6 echoes this same fact. The R-squares in this table shows the percentages of variance of the variables that are overlapped with the factors. While all these percentages (0.74 – 0.97) are quite high for all variables, the percentage is especially high for y1. It shares 97% of the variance with the math factor. So, it appears that the observed variable y1 is almost a perfect indicator of the math factor.

Output 26.12.6 Squared Multiple Correlations of the CFA Model: Scores Data

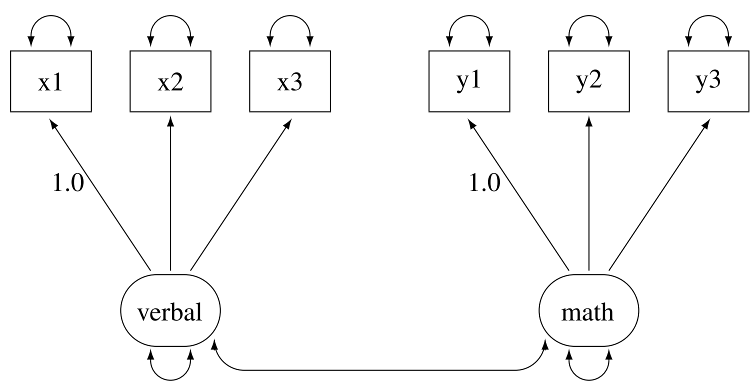
Squared Multiple Correlations

Variable	Error Variance	Total Variance	R-Square
x1	11.52376	45.63609	0.7475
x2	9.14503	42.99597	0.7873
x3	6.68169	28.41532	0.7649
y1	0.78580	28.66835	0.9726
y2	2.88069	20.52319	0.8596
y3	5.15573	19.29032	0.7327

Alternative Identification Constraints

Setting the variances of the latent factors to 1 in the preceding FACTOR model specification makes the model identified. This is necessary because the scales of the latent factors are arbitrary and the constraints imposed on the factor variances fix the scales of the factors.

In practice, there is another way to fix the scales of the factors. For each factor, you can fix the loading of one of its measured indicators to a constant. This fixed loading value is usually set at 1. For example, you can represent the confirmatory factor model for the scores data by the following alternative path diagram:



This path diagram is essentially the same as the preceding one. However, the fixed constants adjacent to the double-headed arrows that attach to the two factors in the preceding path diagram are now moved to two of the single-headed paths in the current path diagram.

You can specify this path diagram by the following FACTOR model specification of PROC CALIS:

```
proc calis data=scores;
   factor
      verbal ---> x1-x3    = 1. ,
      math   ---> y1-y3    = 1. ;
run;
```

In the FACTOR statement, you assign a fixed constant to each of the path entries. In the first entry, the constant 1 is assigned to the loading of x1 on the verbal factor, while all other loadings in this entry are (unnamed) free parameters. Similarly, in the second entry, the fixed constant 1 is assigned to the loading of y1 on the math factor, while all other loadings in this entry are (unnamed) free parameters. This completes the specification of the confirmatory factor model because all the double-headed arrows in the path diagram correspond to default free parameters in the FACTOR modeling language of PROC CALIS.

Output 26.12.7 shows some fit indices for the current confirmatory factor model for the scores data.

Output 26.12.7 Fit Summary of the CFA Model with Alternative Identification Constraints: Scores Data

Fit Summary

Chi-Square	9.8052
Chi-Square DF	8
Pr > Chi-Square	0.2790
Standardized RMSR (SRMSR)	0.0571
RMSEA Estimate	0.0853
Bentler Comparative Fit Index	0.9887

The model fit chi-square is 9.805 (df=8, p=0.279). This is the same model fit chi-square as that for the preceding CFA model specification with factor variances constrained to 1. In fact, all fit information in Output 26.12.7 are identical to Output 26.12.3.

Output 26.12.8 shows the parameter estimates under the current model specification. The loading of x1 on the verbal factor is a fixed at 1, as required for the identification of the scale of the verbal factor. Similarly, the loading of y1 on the math factor is a fixed at 1 for the identification of the scale of the math factor. All other loading estimates in Output 26.12.8 are not the same as those in the preceding model specification, as shown in Output 26.12.4. The reason is that the scales of the factors (as measured by the estimated standard deviations of the factors) in the two specifications are not the same. In the current model specification, the verbal factor has an estimated variance of 34.1123 and the math factor has an estimated variance of 27.8825, as shown in the second table of Output 26.12.8. Hence, the estimated standard deviations of these two factors are 5.8406 and 5.2804, respectively. But the standard deviations of the factors in the preceding confirmatory factor model specification are fixed at 1.

Output 26.12.8 Loading and Factor Covariance Estimates of the CFA Model with Alternative Identification Constraints: Scores Data

```
                Factor Loading Matrix: Estimate/StdErr/t-value

                              verbal              math

             x1               1.0000                 0



             x2               0.9962                 0
                              0.1576
                              6.3194
                            [_Parm1]

             x3               0.7982                 0
                              0.1286
                              6.2083
                            [_Parm2]

             y1                    0            1.0000



             y2                    0            0.7955
                                                0.0718
                                               11.0820
                                              [_Parm3]

             y3                    0            0.7120
                                                0.0858
                                                8.3027
                                              [_Parm4]

               Factor Covariance Matrix: Estimate/StdErr/t-value

                                verbal              math

           verbal              34.1123           15.9585
                               11.6366            6.7270
                                2.9315            2.3723
                              [_Add1]           [_Add3]

           math                15.9585           27.8825
                                6.7270            7.3905
                                2.3723            3.7727
                              [_Add3]           [_Add2]
```

However, if you multiply the loading estimates in Output 26.12.8 by the corresponding estimated factor standard deviation, you get the same set of loading estimates as in Output 26.12.4. For example, the loading of x1 on the verbal factor is 1.0 in Output 26.12.8. Multiplying this loading by the estimated standard deviation 5.8406 of the verbal factor gives you the same corresponding loading as in Output 26.12.4. Another example is the loading of y3 on the math factor. This loading is 0.7120 in Output 26.12.8. Multiplying this estimate by the estimated standard deviation 5.2804 of the verbal factor gives an estimate of 3.7596, which matches the corresponding loading estimate in Output 26.12.4. Therefore, the discrepancies in the loading estimates are due to different factor scales in the two specifications. The loading estimates in Output 26.12.8 are simply rescaled version of the loading estimates in Output 26.12.4.

However, the scales of the factors do not affect the estimates of the error variances, as shown in Output 26.12.9. These estimates are the same as those for the preceding model specification, as shown in the Output 26.12.5.

Output 26.12.9 Error Variance Estimates of the CFA Model with Alternative Identification Constraints: Scores Data

Error Variances

Variable	Parameter	Estimate	Standard Error	t Value
x1	_Add4	11.52376	4.26398	2.70259
x2	_Add5	9.14503	3.83219	2.38637
x3	_Add6	6.68169	2.59770	2.57216
y1	_Add7	0.78580	1.29440	0.60708
y2	_Add8	2.88069	1.09395	2.63329
y3	_Add9	5.15573	1.46854	3.51080

This example shows how you can fit a basic confirmatory factor model by the FACTOR modeling language of PROC CALIS. You can set the identification constraints and get statistically equivalent estimation results in two different ways. By setting up additional parameter constraints, you can also fit some variations of the basic confirmatory factor model. See Example 26.13 for illustrations of some restricted confirmatory factor models for the scores data.

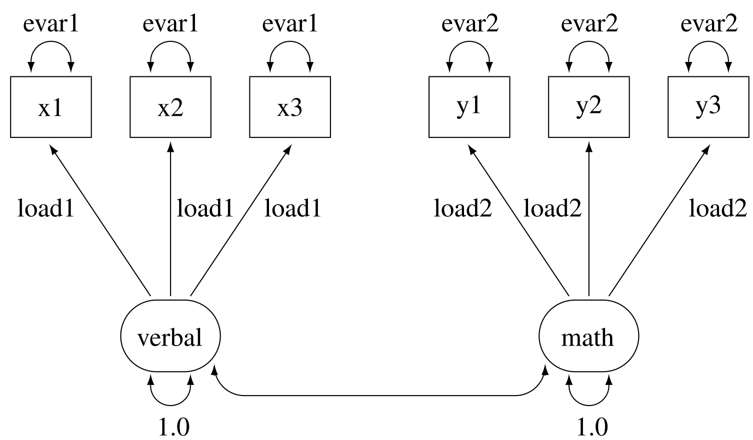
When your data have missing values, with the default ML estimation method PROC CALIS deletes all observations with missing values for the analysis. This might result in a serious loss of information. Example 26.14 considers a hypothetical situation where some observations in the scores data have missing values in the observed variables. Only 16 observations have complete data. By using the full information maximum likelihood (FIML) method for treating the missing data, Example 26.14 shows how you can fully use the information from the scores data set with missing values.

Example 26.13: Confirmatory Factor Models: Some Variations

This example shows how you can fit some variations of the basic confirmatory factor analysis model by the FACTOR modeling language. You apply these models to the scores data set that is described in Example 26.12. The data set contains six test scores of verbal and math abilities. Thirty-two students take the tests. Tests x1–x3 test their verbal skills and tests y1–y3 test their math skills.

The Parallel Tests Model

In classical measurement theory, test items for a latent factor are parallel if they have the same loadings on the factor and the same error variances (or reliability). Suppose for the scores data, the items within each of the verbal and the math factors are parallel. You can use the following path diagram to represent such a parallel tests model:



In the path diagram, the variances of the verbal and the math are both fixed at 1, as indicated by the constants 1.0 adjacent to the double-headed arrows that are attached to factors. You label all the single-headed paths in the path diagram by parameter names. For the three paths (loadings) from the verbal factor, you use the same parameter name load1. This means that these loadings are the same parameter. You also label the double-headed arrows that are attached to x1–x3 by the parameter name evar1. This means that the corresponding error variances for these three observed variables are exactly the same. Hence, x1–x3 are parallel tests for the verbal factor, as required by the current confirmatory factor model.

Similarly, you define parallel tests y1–y3 for the math factor by using load2 as the common factor loading parameter and evar2 as the common error variances for the observed variables.

Corresponding to this path diagram, you can specify the model by the following FACTOR model specification of PROC CALIS:

```
proc calis data=scores;
   factor
      verbal ---> x1-x3    = load1 load1 load1,
      math   ---> y1-y3    = load2 load2 load2;
   pvar
      verbal = 1.,
      math   = 1.,
      x1-x3  = 3*evar1,
      y1-y3  = 3*evar2;
run;
```

In each entry of the FACTOR statement, you specify the factor-variables relationships, followed by a list of parameters. For example, the three loading parameters of x1–x3 on the verbal factor are all named load1. This effectively constrains the corresponding loading estimates to be the same. Similarly, in the next entry you set equality constraints on the loading estimates y1–y3 on the math factor by using the same parameter name load2.

To make the tests parallel, you also need to constrain the error variances for each variable cluster. In the PVAR statement, in addition to setting the factor variances to 1 for identification, you set all the error variances of x1–x3 to be the same by using the same parameter name evar1. The notation 3*evar1 means that you want to specify evar1 three times, one time each for the error variances for the three observed variables in the variable list of the entry. Similarly, you set the equality of the error variances of y1–y3 by using the same parameter name evar2.

Output 26.13.1 shows some fit indices of the parallel tests model for the scores data. The model fit chi-square is 26.128 (df=16, p=0.0522). The SRMSR value is 0.1537 and the RMSEA value is 0.1429. All these indices show that the model does not fit very well. However, Bentler's CFI is 0.9366, which shows a good model fit.

Output 26.13.1 Model Fit of the Parallel Tests Model: Scores Data

Fit Summary

Chi-Square	26.1283
Chi-Square DF	16
Pr > Chi-Square	0.0522
Standardized RMSR (SRMSR)	0.1537
RMSEA Estimate	0.1429
Bentler Comparative Fit Index	0.9366

Output 26.13.2 shows the parameter estimates of the parallel tests model. The first table of Output 26.13.2 shows the required factor pattern for parallel tests. Variables x1–x3 all have the same loading estimates on the verbal factor, and variables y1–y3 all have the same loading estimates on the math factor. All loading estimates are statistically significant.

Output 26.13.2 Parameter Estimates of the Parallel Tests Model: Scores Data

```
                  Factor Loading Matrix: Estimate/StdErr/t-value

                                 verbal             math

               x1                5.4226                0
                                 0.7655
                                 7.0833
                                [load1]

               x2                5.4226                0
                                 0.7655
                                 7.0833
                                [load1]

               x3                5.4226                0
                                 0.7655
                                 7.0833
                                [load1]

               y1                     0           4.4001
                                                  0.5926
                                                  7.4246
                                                 [load2]

               y2                     0           4.4001
                                                  0.5926
                                                  7.4246
                                                 [load2]

               y3                     0           4.4001
                                                  0.5926
                                                  7.4246
                                                 [load2]

                 Factor Covariance Matrix: Estimate/StdErr/t-value

                                   verbal             math

             verbal                1.0000           0.5024
                                                    0.1497
                                                    3.3569
                                                   [_Add1]

             math                  0.5024           1.0000
                                   0.1497
                                   3.3569
                                  [_Add1]
```

Output 26.13.2 *continued*

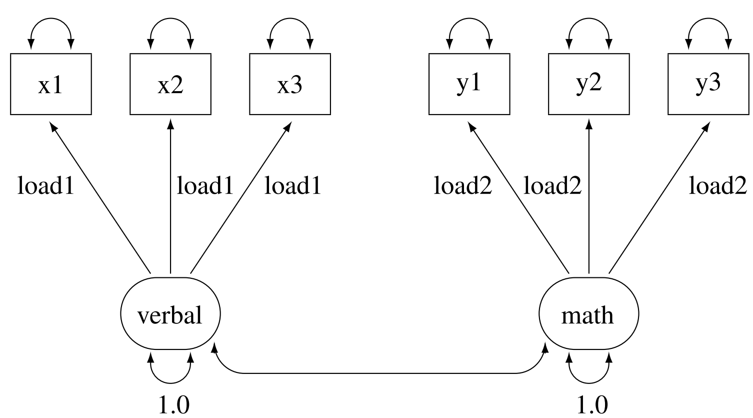
Error Variances

Variable	Parameter	Estimate	Standard Error	t Value
x1	evar1	9.61122	1.72623	5.56776
x2	evar1	9.61122	1.72623	5.56776
x3	evar1	9.61122	1.72623	5.56776
y1	evar2	3.46673	0.62264	5.56776
y2	evar2	3.46673	0.62264	5.56776
y3	evar2	3.46673	0.62264	5.56776

In the second table of Output 26.13.2, the factor covariance (or correlation) estimate is 0.5024, showing moderate relationship between the verbal and the math factors. The last table of Output 26.13.2 shows the error variances of the variables. As required by the parallel tests model, the error variance estimates of x1–x3 are all 9.6112, and the error variance estimates of y1–y3 are all 3.4667.

The Tau-Equivalent Tests Model

Because the parallel tests model does not fit well, you are looking for a less constrained model for the scores data. The tau-equivalent tests model is such a model. It requires only the equality of factor loadings but not the equality of error variances within each factor. The following path diagram represents the tau-equivalent tests model for the scores data:



This path diagram is much the same as that for the parallel tests model except that now you do not use parameter names to label the double-headed arrows that are attached to the observed variables. This means that you allow the corresponding error variances to be free parameters in the tau-equivalent tests model. You can use the following FACTOR model specification of PROC CALIS to specify the tau-equivalent tests model for the scores data:

```
proc calis data=scores;
   factor
      verbal ---> x1-x3   = load1 load1 load1,
      math   ---> y1-y3   = load2 load2 load2;
   pvar
      verbal = 1.,
      math   = 1.;
run;
```

This specification is the same as that for the parallel tests model except that you remove the specifications about the error variances in the PVAR statement in the current tau-equivalent model. This effectively allows the error variances of the observed variables to be (default) free parameters in the model.

Output 26.13.3 shows some model fit indices of the tau-equivalent tests model for the scores data. The chi-square is 22.0468 ($df = 12$, $p = 0.037$). The SRMSR is 0.1398 and the RMSEA is 0.1643. The comparative fit index (CFI) is 0.9371. Except for the CFI value, all other values do not support a good model fit. This model has a degrees of freedom of 12, which is less restrictive (has more parameters) than the parallel tests model, which has a degrees of freedom of 16, as shown in Output 26.13.1. However, it seems that the tau-equivalent tests model is still too restrictive for the data.

Output 26.13.3 Model Fit of the Tau-Equivalent Tests Model: Scores Data

Fit Summary

Chi-Square	22.0468
Chi-Square DF	12
Pr > Chi-Square	0.0370
Standardized RMSR (SRMSR)	0.1398
RMSEA Estimate	0.1643
Bentler Comparative Fit Index	0.9371

Output 26.13.4 shows the parameter estimates. The first table of Output 26.13.4 shows the required pattern of factor loadings under the tau-equivalent tests model. The third table of Output 26.13.4 shows the error variance estimates. The error variance parameters are no longer constrained under the tau-equivalent tests model. Each has a unique estimate.

Output 26.13.4 Parameter Estimates of the Tau-Equivalent Tests Model: Scores Data

Factor Loading Matrix: Estimate/StdErr/t-value

	verbal	math
x1	5.2418 0.7374 7.1085 [load1]	0
x2	5.2418 0.7374 7.1085 [load1]	0
x3	5.2418 0.7374 7.1085 [load1]	0
y1	0	4.4462 0.5932 7.4953 [load2]
y2	0	4.4462 0.5932 7.4953 [load2]
y3	0	4.4462 0.5932 7.4953 [load2]

Factor Covariance Matrix: Estimate/StdErr/t-value

	verbal	math
verbal	1.0000	0.4514 0.1569 2.8772 [_Add1]
math	0.4514 0.1569 2.8772 [_Add1]	1.0000

Output 26.13.4 *continued*

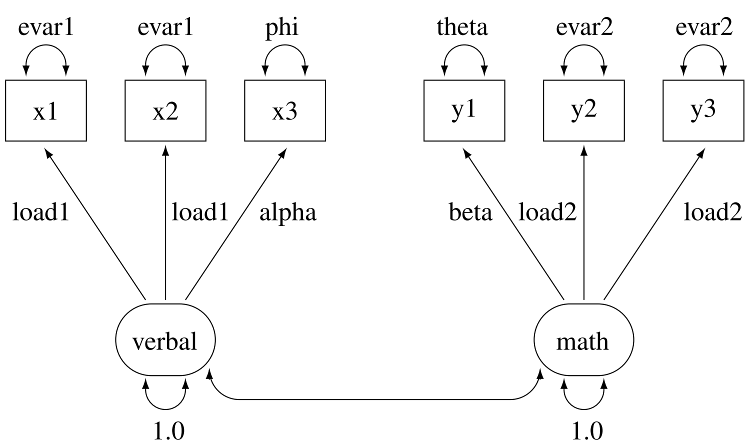
Error Variances

Variable	Parameter	Estimate	Standard Error	t Value
x1	_Add2	13.05681	4.19549	3.11210
x2	_Add3	10.80421	3.70322	2.91752
x3	_Add4	5.43527	2.72147	1.99719
y1	_Add5	3.29858	1.24673	2.64578
y2	_Add6	1.90435	1.02393	1.85984
y3	_Add7	5.09724	1.61477	3.15663

The Partially Constrained Parallel Tests Model

Because both the parallel tests and tau-equivalent tests models do not fit the data well, you can explore an alternative model for the scores data. Suppose that for each factor only two (but not all) of their measured variables (tests) are parallel. For example, suppose you know that tests x1 and x2 are very similar to each other (for example, both are speeded tests with forced-choice answers), while x3 is a little different in the way it is administered (for example, open-ended questions). Although all tests are designed for measuring the verbal factor, only x1 and x2 are parallel tests while x3 is congeneric to the verbal factor. Similarly, suppose you can argue that y2 and y3 are parallel tests while y1 is only congeneric to the math factor.

The current modeling idea is represented by the following path diagram:



In the path diagram, x1 and x2 have the same parameter load1 for the paths from the verbal factor. Their error variances are also the same, as labeled with the evar1 parameter adjacent to the double-headed arrows that are attached to the variables. The test x3 has distinct parameter names for its associated path and the attached double-headed arrow. The corresponding loading and error variance parameters are alpha and phi, respectively. Similarly, with the use of specific parameter names, you define y2 and y3 as parallel tests for the math factor, while y1 is congeneric to the same factor but with distinct loading and error variance parameters. Lastly, you fix the variances of the factors to 1.0 for identification of the factor scales.

You can specify such a partially constrained parallel tests model by the following FACTOR model specification of PROC CALIS:

```
proc calis data=scores;
   factor
      verbal ---> x1-x3   = load1 load1 alpha,
      math   ---> y1-y3   = beta  load2 load2;
   pvar
      verbal = 1.,
      math   = 1.,
      x1-x3  = evar1  evar1  phi,
      y1-y3  = theta  evar2  evar2;
run;
```

First, in the FACTOR statement, you name the loading parameters that reflect the parallel tests constraints. For example, the loading parameters of x1 and x2 on the verbal factor are both named load1. This means that they are the same. However, the loading parameter of x3 on the verbal factor is named alpha, which means that it is a separate parameter. Similarly, you apply the load2 parameter name to the loading parameters of y2 and y3 on the math factor, but the loading parameter of y1 on the math factor is a distinct parameter named beta.

In the PVAR statement, the two factor variances are set to a constant 1 for the identification of latent factor scales. Next, you use the same naming techniques as in the FACTOR statement to constrain some parts of the error variances. As a result, together with the specifications in the FACTOR statement, x1 and x2 are parallel tests for the verbal factor and y2 and y3 are parallel tests for the math factor, while x3 and y1 are only congeneric tests for their respective factors.

Output 26.13.5 shows some fit indices of the partially constrained parallel tests model. The model fit chi-square is 12.6784 ($df = 12$, $p = 0.3928$). The SRMSR is 0.0585 and the RMSEA is close to 0.0427. The comparative fit index (CFI) is 0.9958. All these fit indices point to a quite reasonable model fit for the scores data.

Output 26.13.5 Model Fit of the Partially Constrained Parallel Tests Model: Scores Data

Fit Summary

Chi-Square	12.6784
Chi-Square DF	12
Pr > Chi-Square	0.3928
Standardized RMSR (SRMSR)	0.0585
RMSEA Estimate	0.0427
Bentler Comparative Fit Index	0.9958

Notice that the current model actually has the same degrees of freedom as that of the tau-equivalent tests model, as shown in Output 26.13.3. Both models have nine parameters. But the current partially constrained parallel tests model is definitely a better model for the data. This shows that sometimes you do not have to add more parameters to improve the model fit. Structurally different models might explain the data quite differently, even though they might use the same number of parameters.

Output 26.13.6 show the parameter estimates of the partially constrained parallel tests model for the scores data. The estimates in the factor loading matrix and error variances table confirm the prescribed nature of the tests—that is, x1 and x2 are parallel tests for the verbal factor and y2 and y3 are parallel tests for the math factor.

Output 26.13.6 Parameter Estimates of the Partially Constrained Parallel Tests Model: Scores Data

```
                 Factor Loading Matrix: Estimate/StdErr/t-value

                               verbal              math

            x1                 5.8306                 0
                               0.8593
                               6.7853
                              [load1]

            x2                 5.8306                 0
                               0.8593
                               6.7853
                              [load1]

            x3                 4.6623                 0
                               0.7814
                               5.9664
                              [alpha]

            y1                      0            5.2784
                                                 0.7010
                                                 7.5294
                                                 [beta]

            y2                      0            3.9789
                                                 0.5732
                                                 6.9419
                                                [load2]

            y3                      0            3.9789
                                                 0.5732
                                                 6.9419
                                                [load2]

                Factor Covariance Matrix: Estimate/StdErr/t-value

                                 verbal              math

          verbal                 1.0000            0.5203
                                                   0.1425
                                                   3.6497
                                                  [_Add1]

          math                   0.5203            1.0000
                                 0.1425
                                 3.6497
                                [_Add1]
```

Output 26.13.6 *continued*

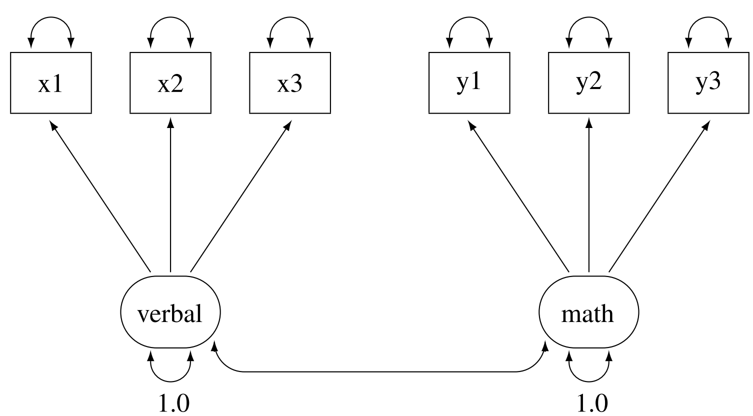
Error Variances

Variable	Parameter	Estimate	Standard Error	t Value
x1	evar1	10.31998	2.57827	4.00268
x2	evar1	10.31998	2.57827	4.00268
x3	phi	6.67832	2.59902	2.56956
y1	theta	0.80714	1.35247	0.59679
y2	evar2	4.07534	1.00371	4.06028
y3	evar2	4.07534	1.00371	4.06028

Example 26.14: The Full Information Maximum Likelihood Method

This example shows how you can fully utilize all available information from the data when there is a high proportion of observations with random missing value. You use the full information maximum likelihood method for model estimation.

In Example 26.12, 32 students take six tests. These six tests are indicator measures of two ability factors: verbal and math. You conduct a confirmatory factor analysis in Example 26.12 based on a data set without any missing values. The path diagram for the confirmatory factor model is shown the following:



Suppose now due to sickness or unexpected events, some students cannot take part in one of these tests. Now, the data test contains missing values at various locations, as indicated by the following DATA step:

```
data missing;
   input x1 x2 x3 y1 y2 y3;
   datalines;
 23   .  16  15  14  16
 29  26  23  22  18  19
 14  21   .  15  16  18
 20  18  17  18  21  19
 25  26  22   .  21  26
 26  19  15  16  17  17
  .  17  19   4   6   7
 12  17  18  14  16   .
 25  19  22  22  20  20
  7  12  15  10  11   8
 29  24   .  14  13  16
 28  24  29  19  19  21
 12   9  10  18  19   .
 11   .  12  15  16  16
 20  14  15  24  23  16
 26  25   .  24  23  24
 20  16  19  22  21  20
 14   .  15  17  19  23
 14  20  13  24   .   .
 29  24  24  21  20  18
 26   .  26  28  26  23
 20  23  24  22  23  22
 23  24  20  23  22  18
 14   .  17   .  16  14
 28  34  27  25  21  21
 17  12  10  14  12  16
  .   1  13  14  15  14
 22  19  19  13  11  14
 18  21   .  15  18  19
 12  12  10  13  13  16
 22  14  20  20  18  19
 29  21  22  13  17   .
;
```

This data set is similar to the scores data set used in Example 26.12, except that some values are replaced at random with missing values. You can still fit the same confirmatory factor analysis model described in Example 26.12 to this data set by the default maximum likelihood (ML) method, as shown in the following statement:

```
proc calis data=missing;
   factor
      verbal ---> x1-x3,
      math   ---> y1-y3;
   pvar
      verbal = 1.,
      math   = 1.;
run;
```

The data set, the number of observations, the model type, and analysis type are shown in the first table of Output 26.14.1. Although PROC CALIS reads all 32 records in the data set, only 16 of these records are used. The remaining 16 records contain at least one missing value in the tests. They are discarded from the analysis. Therefore, the maximum likelihood method only uses those 16 observations without missing values.

Output 26.14.1 Modeling Information of the CFA Model: Missing Data

```
           Confirmatory Factor Model With \Dataset{Missing} Data: ML
                          FACTOR Model Specification

                             The CALIS Procedure
            Covariance Structure Analysis: Model and Initial Values

                             Modeling Information

                      Data Set              WORK.MISSING
                      N Records Read        32
                      N Records Used        16
                      N Obs                 16
                      Model Type            FACTOR
                      Analysis              Covariances
```

Output 26.14.2 shows the parameter estimates.

Output 26.14.2 Parameter Estimates of the CFA Model: Missing Data

Factor Loading Matrix: Estimate/StdErr/t-value

	verbal	math
x1	5.1110 1.3110 3.8984 [_Parm1]	0
x2	5.6261 1.2561 4.4790 [_Parm2]	0
x3	4.8739 1.1410 4.2717 [_Parm3]	0
y1	0	4.4529 0.8530 5.2205 [_Parm4]
y2	0	3.8562 0.8303 4.6444 [_Parm5]
y3	0	2.6338 0.7416 3.5513 [_Parm6]

Factor Covariance Matrix: Estimate/StdErr/t-value

	verbal	math
verbal	1.0000	0.7050 0.1464 4.8165 [_Add1]
math	0.7050 0.1464 4.8165 [_Add1]	1.0000

Output 26.14.2 *continued*

Error Variances

Variable	Parameter	Estimate	Standard Error	t Value
x1	_Add2	11.27773	5.19739	2.16988
x2	_Add3	6.33003	4.25356	1.48817
x3	_Add4	6.47402	3.61040	1.79316
y1	_Add5	0.57143	1.51781	0.37648
y2	_Add6	2.57992	1.47618	1.74770
y3	_Add7	4.59651	1.77777	2.58555

Most of the factor loading estimates shown in Output 26.14.2 are similar to those estimated from the data set without missing values, as shown in Output 26.12.4. The loading estimate of y3 on the math factor shows the largest discrepancy. With only half of the data used in the current estimation, this loading estimate is 2.6338 in the current analysis, while it is 3.7596 if no data were missing, as shown in Output 26.12.4. Another obvious difference between the two sets of results is that the standard error estimates for the loadings are consistently larger in the current analysis than in the analysis in Example 26.12 where there are no missing data. This is expected because you have only half of the data set available in the current analysis.

Similarly, the estimates for the factor covariance and error variances are mostly similar to those in the analysis with complete data, but the standard error estimates in the current analysis are consistently higher.

The maximum likelihood method, as implemented in PROC CALIS, deletes all observations with at least one missing value in the estimation. In a sense, the partially available information of these deleted observations is wasted. This greatly reduces the efficiency of the estimation, which results in higher standard error estimates.

To fully utilize all available information from the data set with the presence of missing values, you can use the full information maximum likelihood (FIML) method in PROC CALIS, as shown in the following statements:

```
proc calis method=fiml data=missing;
   factor
      verbal ---> x1-x3,
      math   ---> y1-y3;
   pvar
      verbal = 1.,
      math   = 1.;
run;
```

In the PROC CALIS statement, you use METHOD=FIML to request the full information maximum likelihood method. Instead of deleting observations with missing values, the full information maximum likelihood method uses all available information in all observations. Output 26.14.3 shows some modeling information of the FIML estimation of the confirmatory factor model on the missing data.

Output 26.14.3 Modeling Information of the CFA Model with FIML: Missing Data

```
            Confirmatory Factor Model With Missing Data: FIML
                         FACTOR Model Specification

                            The CALIS Procedure
          Mean and Covariance Structures: Model and Initial Values

                           Modeling Information

                Data Set                    WORK.MISSING
                N Records Read              32
                N Complete Records          16
                N Incomplete Records        16
                N Complete Obs              16
                N Incomplete Obs            16
                Model Type                  FACTOR
                Analysis                    Means and Covariances
```

PROC CALIS shows you that the number of complete observations is 16 and the number of incomplete observations is 16 in the data set. All these observations are included in the estimation. The analysis type is 'Means and Covariances' because with full information maximum likelihood, the sample means have to be analyzed during the estimation.

For the full information maximum likelihood estimation, PROC CALIS outputs several tables to summarize the missing data patterns and statistics. Output 26.14.4 shows the proportions of data that are present for the variables, individually or jointly by pairs.

Output 26.14.4 Proportions of Data Present for the Variables: Missing Data

Proportions of Data Present for Means (Diagonal) and Covariances (Off-Diagonal)

	x1	x2	x3	y1	y2	y3
x1	0.9375					
x2	0.7813	0.8438				
x3	0.8125	0.7188	0.8750			
y1	0.8750	0.8125	0.8125	0.9375		
y2	0.9063	0.8125	0.8438	0.9063	0.9688	
y3	0.8125	0.7188	0.7500	0.8125	0.8750	0.8750

Average Proportion Coverage of Means	0.906250
Average Proportion Coverage of Covariances	0.816667

The diagonal elements of the table in Output 26.14.4 show the proportions of data coverage by each of the variables. The off-diagonal elements shows the proportions of joint data coverage by all possible pairs of variables. For example, the first diagonal element of the table shows that about 94% of the observations have x1 values that are not missing. This percentage value is referred to as the proportion coverage for x1 or the proportion coverage for computing the means of x1. The off-diagonal element for x1 and x2 shows that about 78% of the observations have nonmissing values for both thier x1 and x2 values. This percentage value is referred to as the joint proportion coverage of x1 and x2 or the proportion coverage for computing the covariance between x1 and x2. The larger the coverage proportions this table shows, the more relative information the data contain for estimating the corresponding moments.

To summarize the proportion coverage, Output 26.14.4 shows that on average about 91% of the data are nonmissing for computing the means, and about 82% of the data are nonmissing for computing the covariances.

Output 26.14.5 shows the lowest coverage proportions of the means and the covariances.

Output 26.14.5 Ranking the Lowest Coverage Proportions: Missing Data

Rank Order of the 3 Smallest Variable (Mean) Coverages

Variable	Coverage
x2	0.8438
x3	0.8750
y3	0.8750

Rank Order of the 7 Smallest Covariance Coverages

Var1	Var2	Coverage
x3	x2	0.7188
y3	x2	0.7188
y3	x3	0.7500
x2	x1	0.7813
x3	x1	0.8125
y1	x2	0.8125
y1	x3	0.8125

The first table of Output 26.14.5 shows that x2 has the lowest proportion coverage at about 84%, and x3 and y3 are the next at about 88%. The second table of Output 26.14.5 shows that the joint proportion coverage by the x3-x2 pair and the y3-x2 pair are the lowest at about 72%, followed by the y3-x3 pair at 75%. These two tables are useful to diagnose which variables most lack the information for estimation. For this data set, these tables show that estimation related to the moments of x2, x3, and y3 suffers the missing data problem the most. However, because the worst proportion coverage is still higher than 70%, the missingness problem does not seem to be very serious based on percentage.

In Output 26.14.6, PROC CALIS outputs two tables that show an overall picture of the missing patterns in the data set.

Output 26.14.6 The Most Frequent Missing Patterns and Their Mean Profiles: Missing Data

Rank Order of the 5 Most Frequent Missing Patterns
Total Number of Distinct Patterns with Missing Values = 7

	Pattern	NVar Miss	Freq	Proportion	Cumulative
1	x.xxxx	1	4	0.1250	0.1250
2	xx.xxx	1	4	0.1250	0.2500
3	xxxxx.	1	3	0.0938	0.3438
4	.xxxxx	1	2	0.0625	0.4063
5	xxxx..	2	1	0.0313	0.4375

NOTE: Nonmissing Pattern Proportion = 0.5000 (N=16)

Means of the Nonmissing and the Most Frequent Missing Patterns

		Missing Pattern				
Variable	Nonmissing (N=16)	1 (N=4)	2 (N=4)	3 (N=3)	4 (N=2)	5 (N=1)
x1	21.75000	18.50000	21.75000	17.66667	.	14.00000
x2	19.37500	.	22.75000	15.66667	9.00000	20.00000
x3	19.31250	17.25000	.	16.66667	16.00000	13.00000
y1	19.00000	18.75000	17.00000	15.00000	9.00000	24.00000
y2	18.12500	18.75000	17.50000	17.33333	10.50000	.
y3	17.75000	19.50000	19.25000	.	10.50000	.

The first table of Output 26.14.6 shows that “x.xxxx” and “xx.xxx” are the two most frequent missing patterns in the data set. Each has a frequency of 4. An “x” in the missing pattern denotes a nonmissing value, while a “.” denotes a missing value. Hence, the first pattern has all missing values for the second variable, and the second pattern has all missing values for the third variable. Each of these two missing patterns accounts for 12.5% of the total observations. Together, the five missing patterns shown in Output 26.14.6 account for about 43.8% of the total observations. The note after this table shows that 50% of the total observations do not have any missing values.

To determine exactly which variables are missing in the missing patterns, it is useful to consult the second table in Output 26.14.6. In this table, the variable means of the most frequent missing patterns are shown, together with the variable means of the nonmissing pattern for comparisons. Missing means in this table show that the corresponding variables are not present in the missing patterns. For example, the column labeled “Nonmissing” is for the group of 16 observations that do not have any missing values. Each of the variable means is computed based on 16 observations. The next column labeled “1” is the first missing pattern that has four observations. The variable mean for x2 is missing for this missing pattern group, while each of the other variable means is computed based on four observations. Comparing these means with those in the nonmissing group, it shows that the means for x1, x3, and y1 in the first missing pattern are smaller than those in the nonmissing group, while the means for y2 and y3 are greater. This comparison does not seem to suggest any systematic bias in the means of the first missing pattern group.

However, the nonmissing means in the third missing pattern (the column labeled “3” do show a consistent downward bias, as compared with the means in the nonmissing group. This might mean that respondents with low scores in x1–x3, y1, and y2 tend not to respond to y3 for some reason. Similarly, the fourth missing pattern shows a consistent downward bias in x2, x3, and y1–y3. Whether these patterns suggest a systematic (or nonrandom) pattern of missingness must be judged in the substantive context. Nonetheless, the numerical results if Output 26.14.6 provide some insight on this matter.

The tables shown in Output 26.14.6 do not show all the missing patterns. In general, PROC CALIS shows only the most frequent or dominant missing patterns so that the output results are more focused. By default, if the total number of missing patterns in a data set is below six, then PROC CALIS shows all the missing patterns. If the total number of missing patterns is at least six, PROC CALIS shows up to 10 missing patterns provided that each of these missing patterns accounts for at least 5% of the total observations. The 10 missing patterns is the default maximum number of missing patterns to show, and the 5% is the default proportion threshold for a missing pattern to display. You can override the default maximum number of missing patterns by the MAXMISSPAT= option and the proportion threshold by the TMISSPAT= option.

Output 26.14.7 shows the parameter estimates by the FIML estimation.

Output 26.14.7 Parameter Estimates of the CFA Model with FIML: Missing Data

Factor Loading Matrix: Estimate/StdErr/t-value

	verbal	math
x1	5.5003 1.0025 5.4867 [_Parm1]	0
x2	5.7134 0.9956 5.7385 [_Parm2]	0
x3	4.4417 0.7669 5.7918 [_Parm3]	0
y1	0	4.9277 0.6798 7.2491 [_Parm4]
y2	0	4.1215 0.5716 7.2100 [_Parm5]
y3	0	3.3834 0.6145 5.5058 [_Parm6]

Output 26.14.7 *continued*

Factor Covariance Matrix: Estimate/StdErr/t-value

	verbal	math
verbal	1.0000	0.5014 0.1473 3.4029 [_Add01]
math	0.5014 0.1473 3.4029 [_Add01]	1.0000

Error Variances

Variable	Parameter	Estimate	Standard Error	t Value
x1	_Add08	12.72770	4.77627	2.66478
x2	_Add09	9.35994	4.48806	2.08552
x3	_Add10	5.67393	2.69872	2.10246
y1	_Add11	1.86768	1.36676	1.36650
y2	_Add12	1.49942	0.97322	1.54067
y3	_Add13	5.24973	1.54121	3.40623

First, you can compare the current FIML results with the results in Example 26.12, where maximum likelihood method is used with the complete data set. Overall, the estimates of loadings, factor covariance, and error variances are similar in the two analyses. Next, you compare the current FIML results with the results in Output 26.14.2, where the default ML method is applied to the same data set with missing values. Except for the standard error estimate of the factor covariance, which are very similar with ML and FIML, the standard error estimates with FIML are consistently smaller than those with ML in Output 26.14.2. This means that with FIML, you improve the estimation efficiency by including the partial information in those observations with missing values.

When you have a data set with no missing values, the ML and FIML methods, as implemented in PROC CALIS, are theoretically the same. Both are equally efficient and produce similar estimates (see Example 26.15). FIML and ML are the same estimation technique that maximizes the likelihood function under the multivariate normal distribution. However, in PROC CALIS, the distinction between of ML and FIML concerns different treatments of the missing values. With METHOD=ML, all observations with one or more missing values are discarded from the analysis. With METHOD=FIML, all observations with at least one nonmissing value are included in the analysis.

Example 26.15: Comparing the ML and FIML Estimation

This example uses the complete data set from Example 26.12 to illustrate how the maximum likelihood (ML) and full information maximum likelihood (FIML) methods are theoretically equivalent when you apply them to data set without missing values. In Example 26.14, you apply a confirmatory factor model to a data set with missing values. You find that with METHOD=FIML, you can get more stable estimates than with METHOD=ML (which is the default estimation method). Near the end of Example 26.14, you learn that ML and FIML are theoretically equivalent estimation methods when you apply them to data sets *without* missing values.

However, the ML and FIML methods have two major computational differences in their implementations in PROC CALIS. First, with METHOD=FIML the first-order properties (that is, the means of the variables) of the data are automatically included in the analysis. However, by default you analyze only the second-order properties (that is, the covariances of the variables) with METHOD=ML. Second, the biased sample covariance formula (with N as the variance divisor) is used with METHOD=FIML, while the unbiased sample covariance formula (with DF=$N - 1$ as the variance divisor) is used with METHOD=ML. See the section "Relationships among Estimation Criteria" on page 1252 for more details about the similarities and differences between the ML and FIML methods.

If you take care of these two differences between ML and FIML in PROC CALIS, you can obtain exactly the same results with these two methods when you apply them to data sets without missing values.

For example, with the complete data set scores from Example 26.12, you specify the FIML estimation in the following statements:

```
proc calis method=fiml data=scores;
   factor
      verbal ---> x1-x3,
      math   ---> y1-y3;
   pvar
      verbal = 1.,
      math   = 1.;
run;
```

An equivalent specification with the ML method is shown in the following statements:

```
proc calis method=ml meanstr vardef=n data=scores;
   factor
      verbal ---> x1-x3,
      math   ---> y1-y3;
   pvar
      verbal = 1.,
      math   = 1.;
run;
```

In the PROC CALIS statement, you specify two options to make the ML estimation exactly equivalent to the FIML estimation in PROC CALIS. First, the MEANSTR option requests the first-order properties (the mean structures) to be analyzed with the covariance structures. Second, the VARDEF=N option defines the variance divisor to N, instead of the default DF, which is the same as N−1. These two options make the ML estimation equivalent to the FIML estimation.

Output 26.15.1 and Output 26.15.2 show some fit summary statistics under the FIML and ML methods, respectively.

Output 26.15.1 Model Fitting by the FIML Method: Scores Data

Fit Summary	
Fit Function	31.7837
Chi-Square	10.1215
Chi-Square DF	8
Pr > Chi-Square	0.2566
Standardized RMSR (SRMSR)	0.0504
RMSEA Estimate	0.0910
Bentler Comparative Fit Index	0.9872

Output 26.15.2 Model Fitting by the ML Method: Scores Data

Fit Summary	
Fit Function	0.3163
Chi-Square	10.1215
Chi-Square DF	8
Pr > Chi-Square	0.2566
Standardized RMSR (SRMSR)	0.0504
RMSEA Estimate	0.0910
Bentler Comparative Fit Index	0.9872

Except for the fit function values, both FIML and ML methods produce the same set of fit statistics. The difference in the fit function values is expected because the FIML function has a constant term which is derived from the likelihood function. This constant term does not depend on the model parameters. Hence, the FIML and ML discrepancy functions that are used in PROC CALIS are equivalent when VARDEF=N is used in the ML method for analyzing mean and covariance structures.

The parameter estimates are shown in Output 26.15.3 and Output 26.15.4 for the FIML and ML methods, respectively. Except for very tiny numerical differences in some estimates, the FIML and ML estimates match.

Output 26.15.3 Parameter Estimates by the FIML Method: Scores Data

```
                  Factor Loading Matrix: Estimate/StdErr/t-value

                                 verbal              math

                x1               5.7486                 0
                                 0.9651
                                 5.9567
                               [_Parm1]

                x2               5.7265                 0
                                 0.9239
                                 6.1980
                               [_Parm2]

                x3               4.5886                 0
                                 0.7570
                                 6.0618
                               [_Parm3]

                y1                    0            5.1972
                                                   0.6779
                                                   7.6662
                                                 [_Parm4]

                y2                    0            4.1342
                                                   0.6025
                                                   6.8612
                                                 [_Parm5]

                y3                    0            3.7004
                                                   0.6143
                                                   6.0237
                                                 [_Parm6]

                 Factor Covariance Matrix: Estimate/StdErr/t-value

                                   verbal              math

              verbal               1.0000            0.5175
                                                     0.1406
                                                     3.6804
                                                   [_Add01]

              math                 0.5175            1.0000
                                   0.1406
                                   3.6804
                                 [_Add01]
```

Output 26.15.3 *continued*

Intercepts

Variable	Parameter	Estimate	Standard Error	t Value
x1	_Add02	19.90625	1.17540	16.93575
x2	_Add03	18.81250	1.14089	16.48928
x3	_Add04	18.68750	0.92749	20.14856
y1	_Add05	17.90625	0.93161	19.22084
y2	_Add06	17.84375	0.78823	22.63773
y3	_Add07	17.75000	0.76419	23.22725

Error Variances

Variable	Parameter	Estimate	Standard Error	t Value
x1	_Add08	11.16406	4.06574	2.74589
x2	_Add09	8.85978	3.65403	2.42466
x3	_Add10	6.47248	2.47685	2.61319
y1	_Add11	0.76135	1.23420	0.61687
y2	_Add12	2.79060	1.04306	2.67539
y3	_Add13	4.99466	1.40025	3.56698

Output 26.15.4 Parameter Estimates by the ML Method: Scores Data

```
                Factor Loading Matrix: Estimate/StdErr/t-value

                              verbal              math

            x1                5.7486                 0
                              0.9651
                              5.9567
                            [_Parm1]

            x2                5.7265                 0
                              0.9239
                              6.1981
                            [_Parm2]

            x3                4.5885                 0
                              0.7570
                              6.0617
                            [_Parm3]

            y1                     0            5.1972
                                                0.6779
                                                7.6662
                                              [_Parm4]

            y2                     0            4.1341
                                                0.6025
                                                6.8612
                                              [_Parm5]

            y3                     0            3.7004
                                                0.6143
                                                6.0238
                                              [_Parm6]

              Factor Covariance Matrix: Estimate/StdErr/t-value

                                verbal              math

          verbal                1.0000            0.5175
                                                  0.1406
                                                  3.6800
                                                [_Add01]

          math                  0.5175            1.0000
                                0.1406
                                3.6800
                              [_Add01]
```

Output 26.15.4 *continued*

Intercepts

Variable	Parameter	Estimate	Standard Error	t Value
x1	_Add02	19.90625	1.17540	16.93575
x2	_Add03	18.81250	1.14089	16.48928
x3	_Add04	18.68750	0.92749	20.14856
y1	_Add05	17.90625	0.93161	19.22084
y2	_Add06	17.84375	0.78823	22.63773
y3	_Add07	17.75000	0.76419	23.22725

Error Variances

Variable	Parameter	Estimate	Standard Error	t Value
x1	_Add08	11.16365	4.06567	2.74583
x2	_Add09	8.85925	3.65397	2.42456
x3	_Add10	6.47288	2.47689	2.61331
y1	_Add11	0.76124	1.23420	0.61679
y2	_Add12	2.79066	1.04307	2.67543
y3	_Add13	4.99461	1.40024	3.56697

The equivalence between METHOD=ML and METHOD=FIML implies that if you do not have any missing data in your data, you can just use METHOD=ML because it is computationally more efficient than the FIML method.

While the equivalence between ML and FIML is established here with the use of the VARDEF= and MEANSTR options (for data without missing values), it is not necessary in practice to use these options with METHOD=ML. The VARDEF= option is used in this example only to demonstrate the theoretical equivalence between METHOD=ML and METHOD=FIML. The VARDEF= option has very little effect if you have at least a moderate sample size (for example, 30 or more observations).

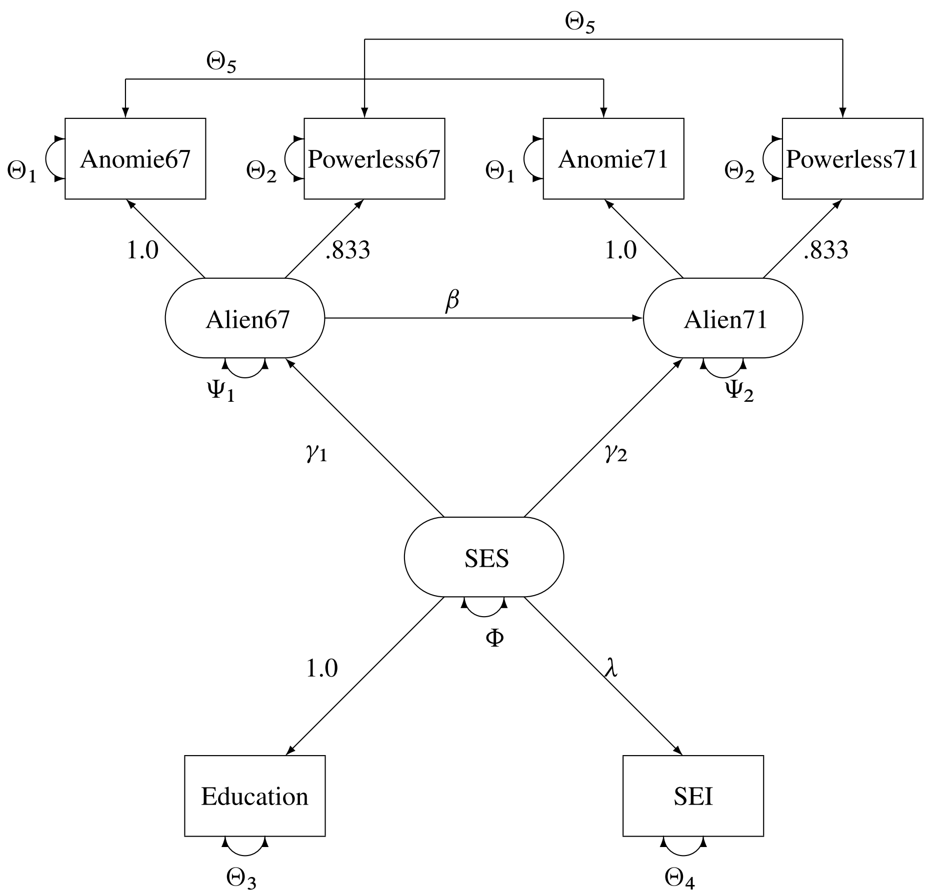
Merely adding the MEANSTR option to an analysis for data without missing values amounts to adding a saturated mean structure to a covariance structure analysis. In this case, the MEANSTR option only gives you more estimates that pertain to the mean structures, but the parameter estimates that pertain to the covariance structures do not change. Therefore, use the MEANSTR option only when you need to estimate certain mean structure parameters or when you fit models with nonsaturated mean structures.

However, use METHOD=FIML when there are missing values in your data and you need to use every bit of information from the incomplete observations with random missing values.

Example 26.16: Path Analysis: Stability of Alienation

The following covariance matrix from Wheaton et al. (1977) has served to illustrate the performance of several implementations for the analysis of structural equation models. Two different models have been analyzed by an early implementation of LISREL and are mentioned in Jöreskog (1978). You can also find a more detailed discussion of these models in the LISREL VI manual (Jöreskog and Sörbom 1985). A slightly modified model for this covariance matrix is included in the EQS 2.0 manual (Bentler 1985, p. 28). However, for the analysis with the EQS implementation, the SEI variable is rescaled by a factor of 0.1 to make the matrix less ill-conditioned. Since the Levenberg-Marquardt or Newton-Raphson optimization techniques are used with PROC CALIS, rescaling the data matrix is not necessary and, therefore, is not done here. The results reported here reflect the estimates based on the original covariance matrix.

The path diagram of this model is displayed in Figure 26.1 and is reproduced in the following:



You use the PATH modeling language of PROC CALIS to specify this path model, as shown in the following statements:

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

ods graphics off;
```

Since no METHOD= option is used in the PROC CALIS statement, maximum likelihood estimates are computed by default.

In the PATH statement, you specify the functional relationships of the variables in the model. These functional relationships are represented as single-headed paths in the path diagram. There are five entries in the PATH statement. You specify the relationships between the latent constructs and the observed variables in the first three path entries. For example, the first entry states that Anomie and Powerless67 are measured indicators of the latent variable Alien67. The path effects or coefficients from the latent factor to these measured indicators are fixed at 1.0 and 0.833, respectively. Similarly, in the next two path entries, you define the relationships between the latent factors Alien71 and SES and their measured indicators. The last two path entries in the PATH statement represent the functional relationships among the latent variables in the model. SES has effects on Alien67 and Alien71. These effect parameters are labeled or named with gamma1 and gamma2, respectively. Alien67 also has an effect on Alien71, with the effect parameter named beta.

In the PVAR statement, you specify the variance or error variance parameters in the model. These parameters correspond to the double-headed arrows pointing to the individual variables in the path diagram. In the first six entries of the PVAR statement, you specify the error variance parameters of the observed variables. You also give names to these parameters that correspond to the notation in the path diagram. Although you can choose any names for the parameters, it is important to remember that parameters with the same name are identical and will have the same estimates. For example, the error variances of Anomie67 and Anomie71 are the same parameter named theta1. Similarly, you constrain the error variances of Powerless67 and Powerless71. However, the error variance parameters of Education and SEI are unique. They are not constrained with other parameters in the model because they have unique parameter names. Next, you specify the error variance parameters of Alien67 and Alien71. They also have unique parameter names and therefore they are not constrained with any other parameters in the model. Lastly, you specify the variance parameter phi of SES.

In the PCOV statement, you specify the covariances or error covariances among variables in the model. These parameters correspond to the double-headed arrows pointing to distinct pairs of variables in the path diagram. Observed variables Anomie67 and Anomie71 have correlated errors and you specify this error covariance parameter as theta5. Similarly, observed variables Powerless67 and Powerless71 have correlated errors and you also specify this error covariance parameter as theta5. This way, the two error covariances are constrained to be equal.

PROC CALIS can produce a high-quality residual histogram that is useful for showing the distribution of residuals. Before you request the residual histogram, ODS Graphics must be enabled. For example, you can specify the ODS GRAPHICS ON statement, as shown in the preceding statements before the PROC CALIS statement. Then, the residual histogram is requested by the **`plots=residuals`** option in the PROC CALIS statement.

Output 26.16.1 displays the modeling information and variables in the analysis.

Output 26.16.1 Model Specification and Variables

```
                    PATH Model Specification

                      The CALIS Procedure
        Covariance Structure Analysis: Model and Initial Values

                      Modeling Information

                  Data Set              WORK.WHEATON
                  N Obs                 932
                  Model Type            PATH
                  Analysis              Covariances
```

Output 26.16.1 *continued*

```
                        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 26.16.1 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 26.16.2 echoes the initial specification of the PATH model.

Output 26.16.2 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                   .
```

Output 26.16.2 *continued*

Initial Estimates for Covariances Among Errors

Error of	Error of	Parameter	Estimate
Anomie67	Anomie71	theta5	.
Powerless67	Powerless71	theta5	.

In Output 26.16.2, 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 26.16.2, 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. PROC CALIS generates the initial values of these parameters automatically.

You can examine this output to ensure that the desired model is being analyzed. PROC CALIS outputs the initial specifications or the estimation results in the order you specify in the model, unless you use reordering options such as ORDERSPEC and ORDERALL. Therefore, the input order of specifications is important—it determines how your output would look.

Simple descriptive statistics are displayed in Output 26.16.3.

Output 26.16.3 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. You can provide these initial estimates or let PROC CALIS to generate them automatically. As shown in Output 26.16.2, you did not provide any initial estimates for the parameters. PROC CALIS uses a combination of well-behaved mathematical methods to complete the initial estimation. The initial estimation methods for the current analysis are shown in Output 26.16.4.

Output 26.16.4 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, the instrumental variable Method, the McDonald and Hartmann method, and the two-stage least squares method have been used for initial estimation. In the same output, the vector of initial parameter estimates and their gradients are also shown. The initial objective function value is 0.0366.

Output 26.16.5 displays the optimization information, including technical details, iteration history and convergence status.

Output 26.16.5 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
```

Output 26.16.5 *continued*

```
                        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 CALIS issues an error message.

The fit summary statistics are displayed in Output 26.16.6. By default, PROC CALIS displays all available fit indices and modeling information.

Output 26.16.6 Fit Summary

Fit Summary

Modeling Info	N Observations	932
	N Variables	6
	N Moments	21
	N Parameters	12
	N Active Constraints	0
	Baseline Model Function Value	2.2894
	Baseline Model Chi-Square	2131.4327
	Baseline Model Chi-Square DF	15
	Pr > Baseline Model Chi-Square	<.0001
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	1169
	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	0.0000
	RMSEA Upper 90% Confidence Limit	0.0470
	Probability of Close Fit	0.9705
	ECVI Estimate	0.0405
	ECVI Lower 90% Confidence Limit	0.0357
	ECVI Upper 90% Confidence Limit	0.0556
	Akaike Information Criterion	37.4851
	Bozdogan CAIC	107.5330
	Schwarz Bayesian Criterion	95.5330
	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 26.16.6. You can check the number of observations, number of variables, number of moments being fitted, number of parameters, number of active constraints in the solution, and the independent model chi-square and its degrees of freedom in this modeling information category. Next, three types of fit indices are shown: absolute, parsimony, and incremental.

The absolute indices are fit measures that you interpret them without referring to any baseline model. These indices do not adjust for model parsimony. They always favor models with a large number of parameters. The chi-square test statistic is the best-known absolute index in this category. In this example, the p-value of the chi-square is 0.1419, which is greater than the conventional 0.05 value. From the statistical hypothesis testing point of view, you cannot reject this model. The Z-test of Wilson and Hilferty is also insignificant at $\alpha = .05$, which echoes the result of the chi-square test. You can consult other absolute indices as well. Although it seems that there are no clear conventional levels for these absolute indices to indicate an acceptable model fit, you can always use these indices to compare the relative fit among competing models.

Next, the parsimony fit indices take the model parsimony into account. These indices adjust the model fit by the degrees of freedom (or the number of the parameters) of the model in certain ways. The advantage of these indices is that merely increasing the number of parameters in the model might not necessarily lead better model fit measures. These fit indices penalize models with large numbers of parameters. There is no universal way to interpret all these indices. However, for the relatively well-known RMSEA estimate, by convention values under 0.05 indicate good model fit. The RMSEA value for this example is 0.0231, and so this is a very good model fit. For interpretations of other parsimony indices, you can consult the original articles for these indices.

Last, the incremental fit indices are computed based on comparing the target model fit against the fit of a baseline model, which is usually the so-called uncorrelatedness model where all manifest variables are assumed to be uncorrelated. This is the baseline model that PROC CALIS uses. The baseline model fit statistic is shown under the Modeling Info category of the same fit summary table. In this example, the model fit chi-square of the baseline model is 2131.43, with 15 degrees of freedom. The incremental indices show how well the hypothesized model improves over the baseline model for the data. Various incremental fit indices have been proposed. In the fit summary table, there are six of such fit indices. Large values for these indices are desired. It has been suggested that values greater than .9 for these indices indicate acceptable model fit. In this example, all incremental indices but James et al. parsimonious NFI show that the hypothesized model fits well.

There is no consensus as to which fit index is the best to judge model fit. Probably, with artificial data and model, all fit indices can be shown defective in some aspects of measuring model fit. Conventional wisdom is to look at all fit indices and determine whether the majority of them are close to the desirable ranges of values. In this example, almost all fit indices are good, and so it is safe to conclude that the model fits well.

Nowadays, most researchers pay less attention to the model fit chi-square statistic because it tends to reject all meaningful models with minimum departures from the truth. Although the model fit chi-square test statistic is an impeccable statistical inference tool when the underlying statistical assumptions are satisfied, for practical purposes it is just too powerful to accept any useful and reasonable models with only tiny imperfections. Some fit indices are more popular than others. Standardized 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 CALIS can perform a detailed residual analysis. Large residuals might indicate misspecification of the model. In Output 26.16.7, raw residuals are reported and ranked.

Output 26.16.7 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	0.00000	0.00000
SEI	Occupational Status Index	-0.59248	0.00000	0.00002

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 26.16.8, for example, the table for the 10 largest asymptotically standardized residuals is displayed.

Output 26.16.8 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	0.00000	0.00000
SEI	Occupational Status Index	-0.49608	0.00000	0.00000

Average Standardized Residual	0.646672
Average Off-diagonal Standardized Residual	0.818456

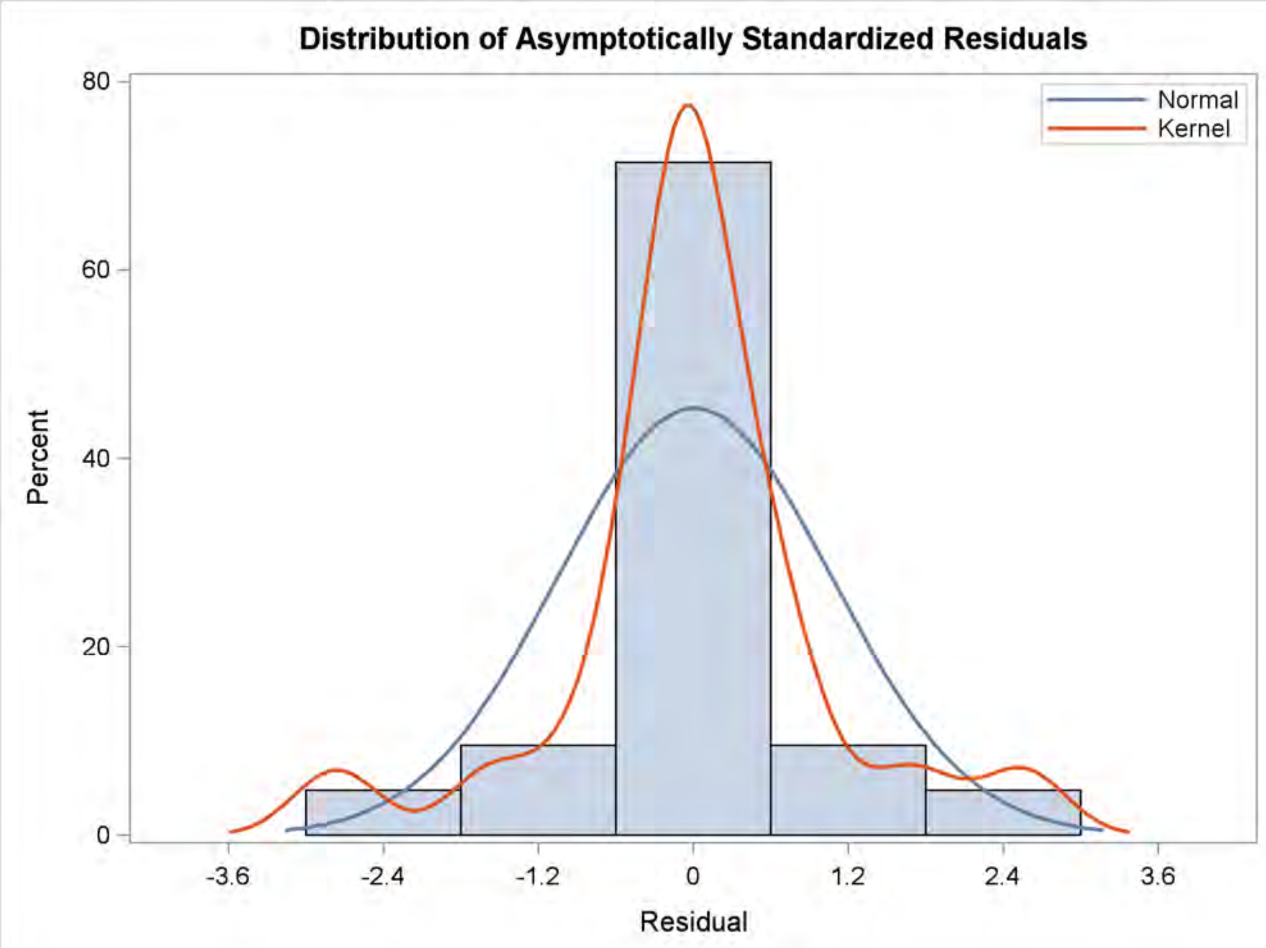
Rank Order of the 10 Largest Asymptotically Standardized Residuals

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 26.16.9, which also shows the normal and kernel approximations.

Output 26.16.9 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.

Output 26.16.10 shows the estimation results.

Output 26.16.10 Estimation Results

PATH List

------------Path------------			Parameter	Estimate	Standard 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 26.16.10 *continued*

Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard 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

Error of	Error of	Parameter	Estimate	Standard 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 26.16.10 are significantly different from zero, supporting the presence of these parameters in the model.

Squared multiple correlations are shown in Output 26.16.11.

Output 26.16.11 Squared Multiple Correlations

Squared Multiple Correlations

Variable	Error Variance	Total 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 26.16.10, 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 simple-minded to conclude that the effect of SES on SEI is greater than that SES on Education. Because SEI and Education are measured on different scales, direct comparison of the corresponding path coefficients is simply inappropriate.

In alleviating this problem, some might resort to the standardized solution for a better comparison. In a standardized solution, because the variances of manifest variables and systematic predictors are all standardized to ones, you hope the path coefficients are more comparable. In this example, PROC CALIS standardizes your results in Output 26.16.12.

Output 26.16.12 Standardized Results

Standardized Results for PATH List

Path			Parameter	Estimate	Standard 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

Standardized Results for Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard 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		

Output 26.16.12 *continued*

Standardized Results for Covariances Among Errors

Error of	Error of	Parameter	Estimate	Standard 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 CALIS the standardized estimates are computed with standard error estimates and *t* values so that you can make statistical inferences on the standardized estimates as well.

PROC CALIS might differ from other software in its standardization scheme. Unlike other software that might standardize the path coefficients that attach to the error terms (unsystematic sources), PROC CALIS keeps these path coefficients at ones (not shown in the output). Unlike other software that might also standardize the corresponding error variances to ones, the error variances in the standardized solution of PROC CALIS are rescaled so as to keep the mathematical consistency of the model.

Essentially, in PROC CALIS only variances of manifest and non-error-type latent variables are standardized to ones. The error variances are rescaled, but not standardized. For example, in the standardized solution shown in Output 26.16.12, 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 1275 for the logic of the standardization scheme adopted by PROC CALIS.

In appearance, the standardized solution is like a correlational analysis on the standardized manifest variables with standardized exogenous latent factors. Unfortunately, this statement is over-simplified, if not totally inappropriate. In standardizing a solution, the implicit equality constraints are likely destroyed. In this example, the unstandardized error variances for Anomie67 and Anomie71 are both 3.608, represented by a common parameter theta1. However, after standardization, these error variances have different values at 0.303 and 0.286, respectively. In addition, fixed parameter values are no longer fixed in a standardized solution (for example, the first five paths in the current example). The issue of standardization is common to all other SEM software and beyond the current discussion. PROC CALIS provides the standardized solution so that users can interpret the standardized estimates whenever they find them appropriate.

Example 26.17: Simultaneous Equations with Mean Structures and Reciprocal Paths

The supply-and-demand food example of Kmenta (1971, pp. 565, 582) is used to illustrate PROC CALIS for the estimation of intercepts and coefficients of simultaneous equations in econometrics. The model is specified by two simultaneous equations containing two endogenous variables Q and P, and three exogenous variables D, F, and Y:

$$Q_t(demand) = \alpha_1 + \beta_1 P_t + \gamma_1 D_t$$

$$Q_t(supply) = \alpha_2 + \beta_2 P_t + \gamma_2 F_t + \gamma_3 Y_t$$

for $t = 1, \ldots, 20$.

To analyze this model in PROC CALIS, the second equation needs to be written in another form. For instance, in the LINEQS model each endogenous variable must appear on the left-hand side of exactly one equation. To satisfy this requirement, you can rewrite the second equation as an equation for P_t as:

$$P_t = -\frac{\alpha_2}{\beta_2} + \frac{1}{\beta_2} Q_t - \frac{\gamma_2}{\beta_2} F_t - \frac{\gamma_3}{\beta_2} Y_t$$

or, equivalently reparameterized as:

$$P_t = \theta_1 + \theta_2 Q_t + \theta_3 F_t + \theta_4 Y_t$$

where

$$\theta_1 = -\frac{\alpha_2}{\beta_2}, \quad \theta_2 = \frac{1}{\beta_2}, \quad \theta_3 = -\frac{\gamma_2}{\beta_2}, \quad \theta_4 = -\frac{\gamma_3}{\beta_2}$$

This new equation for P_t together with the first equation for Q_t suggest the following LINEQS model specification in PROC CALIS:

```
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 calis data=food pshort nostand;
   lineqs
      Q = alpha1 * Intercept + beta1  * P  + gamma1 * D + E1,
      P = theta1 * Intercept + theta2 * Q  + theta3 * F + theta4 * Y + E2;
   variance
      E1-E2 = eps1-eps2;
   cov
      E1-E2 = eps3;
   bounds
      eps1-eps2 >= 0. ;
run;
```

The LINEQS modeling language is used in this example because its specification is similar to the original equations. In the LINEQS statement, you essentially input the two model equations for Q and P. Parameters for intercepts and regression coefficients are also specified in the equations. Note that Intercept in the two equations is treated as a special variable that contains ones for all observations. Intercept is not a variable in the data set, nor do you need to create such a variable in your data set. Hence, the variable Intercept does not represent the intercept parameter itself. Instead, the intercept parameters for the two equations are the coefficients attached to Intercept. In this example, the intercept parameters are alpha1 and theta1, respectively, in the two equations. As required, error terms E1 and E2 are added to complete the equation specification.

In the VARIANCE statement, you specify eps1 and eps2, respectively, for the variance parameters of the error terms. In the COV, you specify eps3 for the covariance parameter between the error terms. In the BOUNDS statement, you set lower bounds for the error variances so that estimates of eps1 and eps2 would be nonnegative.

In this example, the PSHORT and the NOSTAND options are used in the PROC CALIS statement. The PSHORT option suppresses a large amount of the output. For example, initial estimates are not printed and simple descriptive statistics and standard errors are not computed. The NOSTAND option suppresses the printing of the standardized results. Because the default printing in PROC CALIS might produce a large amount of output, using these printing options make your output more concise and readable. Whenever appropriate, you may consider using these printing options.

The estimated equations are shown in Output 26.17.1.

Output 26.17.1 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 26.17.2.

Output 26.17.2 Variance, Covariance, and Mean Parameters

Estimates for Variances of Exogenous Variables

Variable Type	Variable	Parameter	Estimate	Standard 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

Output 26.17.2 *continued*

Covariances Among Exogenous Variables

Var1	Var2	Parameter	Estimate	Standard 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 Type	Variable	Parameter	Estimate	Standard 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 CALIS. These parameters are added as free parameters to complete the model specification. In PROC CALIS, variances and covariances among the set of exogenous manifest variables must be parameters. You either specify them explicitly or let the CALIS procedure to add them. If you need to constrain or to fix these parameters, then you must specify them explicitly. When your model also fits the mean structures, the same principle applies to the means of the exogenous manifest variables. In this example, because variables D, F, and Y are all exogenous manifest variables, their associated means, variances and covariances must be parameters in the model.

The squared multiple correlations for the equations are shown in Output 26.17.3.

Output 26.17.3 Squared Multiple Correlations

Squared Multiple Correlations

Variable	Error Variance	Total 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.

Specifying the LINEQS with the Original Econometric Parameters

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 shown in the following statements:

```
proc calis data=Food pshort nostand;
   lineqs
      Q = alpha1 * Intercept + beta1  * P  + gamma1 * D + E1,
      P = theta1 * Intercept + theta2 * Q  + theta3 * F + theta4 * Y + E2;
   variance
      E1-E2 = eps1-eps2;
   cov
      E1-E2 = eps3;
   bounds
      eps1-eps2 >= 0. ;
   parameters alpha2 (50.) beta2 gamma2 gamma3 (3*.25);
      theta1  = -alpha2 / beta2;
      theta2  = 1 / beta2;
      theta3  = -gamma2 / beta2;
      theta4  = -gamma3 / beta2;
run;
```

In this new specification, only the PARAMETERS statement and the SAS programming statements following it are new. In the PARAMETERS statement, you define super-parameters alpha2, beta2, gamma2, and gamma3, and put initial values for them in parentheses. These parameters are the original econometric parameters of interest. The SAS programming statements that follow the PARAMETERS statement are used to define the functional relationships of the super-parameters with the parameters in the LINEQS model. Consequently, in this new specification, 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 26.17.4.

Output 26.17.4 Additional Parameters

Additional Parameters

Type	Parameter	Estimate	Standard 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 26.17.4, all these estimates are significant, despite the fact that one of the transformed parameter estimates in the linear equations of the LINEQS model is not. You can obtain almost equivalent results by applying the SAS/ETS procedure SYSLIN on this problem.

Alternative Ways to Specify Your LINEQS Model

In specifying the linear equations in the LINEQS model, it might become cumbersome when you need to name a lot of parameters into the equations. If the parameters in your model are unconstrained, you need to very careful to use unique parameter names to distinguish the free parameters because parameters with the same name are identical and will have the same estimate. To make model specification easier and to avoid accidental constraints, PROC CALIS provides an efficient way to specify these free parameters. That is, you can simply omit the parameter names in the specification. For example, in the first specification of the current example, except for the boundary constraints on the error variance parameters, all other parameters in the model are not constrained, as shown in the following statements:

```
proc calis data=food pshort nostand;
   lineqs
      Q = alpha1 * Intercept + beta1  * P  + gamma1 * D + E1,
      P = theta1 * Intercept + theta2 * Q  + theta3 * F + theta4 * Y + E2;
   variance
      E1-E2 = eps1-eps2;
   cov
      E1-E2 = eps3;
   bounds
      eps1-eps2 >= 0. ;
run;
```

Parameters such as alpha1, beta1, and so on are unique parameter names in the specific locations of the model. They are free parameters. Hence, you can use the following equivalent specification:

```
proc calis data=food pshort nostand;
   lineqs
      Q =  * Intercept +  *  P  +  * D        + E1,
      P =  * Intercept +  *  Q  +  * F +  * Y + E2;
   variance
      E1-E2 = eps1-eps2;
   cov
      E1 E2;
   bounds
      eps1-eps2 >= 0. ;
run;
```

Only the parameters eps1 and eps2 remain in this equivalent specification. You omit the specification of all other parameter names. But the estimation results are the same, as shown in Output 26.17.5.

Output 26.17.5 Estimation Results With Generated Parameter Names

```
                              Linear Equations

    Q        =    93.6193*Intercept + -0.2295*P        +  0.3100*D
    Std Err        7.5748 _Parm1        0.0923 _Parm2       0.0448 _Parm3
    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 _Parm4        1.7540 _Parm5       0.3960 _Parm6
    t Value       -1.5897               2.4025             -2.3500

                     + -1.5579*Y         +  1.0000 E2
                        0.6650 _Parm7
                       -2.3429

                 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       _Parm8         -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
```

The estimation results in Output 26.17.5 are the same as those in Output 26.17.2 and Output 26.17.3 with the original LINEQS model specification, only now PROC CALIS generates the parameter names with the _Parm in the results, as shown in Output 26.17.5. Note that you retain the parameter names eps1 and eps2 because you need to refer to them in the BOUNDS statement.

Example 26.18: Fitting Direct Covariance Structures

In the section "Direct Covariance Structures Analysis" on page 1011, 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:

$$\mathbf{\Sigma} = \begin{pmatrix} \phi_1 & \theta_1 & \theta_2 & \theta_1 \\ \theta_1 & \phi_2 & \theta_l & \theta_3 \\ \theta_2 & \theta_1 & \phi_1 & \theta_1 \\ \theta_1 & \theta_3 & \theta_1 & \phi_2 \end{pmatrix}$$

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 26.16), structural equations (see Example 26.17), 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 $\mathbf{\Sigma}$. 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 1011, you can use the MSTRUCT modeling language to specify the hypothesized covariance structures directly, as shown in the following statements:

```
proc calis nobs=932 data=Wheaton psummary;
   fitindex on(only)=[chisq df probchi] outfit=savefit;
   mstruct
      var = Anomie67 Powerless67 Anomie71 Powerless71;
   matrix _COV_ [1,1] = phi1,
                [2,2] = phi2,
                [3,3] = phi1,
                [4,4] = phi2,
                [2,1] = theta1,
                [3,1] = theta2,
                [3,2] = theta1,
                [4,1] = theta1,
                [4,2] = theta3,
                [4,3] = theta1;
run;
```

In the MSTRUCT statement you specify the variables in the VAR= list. The order of variables in this VAR= list is assumed to be the same as that in the row and column of the hypothesized covariance matrix. Next, in the MATRIX statement you specify parameters as entries in the hypothesized covariance matrix _COV_. Only the lower diagonal elements need to be specified because covariance matrices, by nature, are symmetric. Redundant specification of the upper triangular elements are unnecessary as PROC CALIS has the information accounted for. You can also set initial estimates by putting parenthesized numbers after the parameter names. But in this example you let PROC CALIS determine all the initial estimates.

In the PROC CALIS statement, the PSUMMARY option is used. As a global display option, this option suppresses a lot of displayed output and requests only the fit summary table be printed. This way you can eliminate quite a lot of displayed output that is not of your primary interest. In this example, the specification of the covariance structures is straightforward, and you do not need any output regarding the initial estimation or standardized solution. Suppose that you are not even concerned with the estimates of the parameters because you are not yet sure if this model is good enough for the data. All you want to know at this stage is whether the hypothesized covariance structures fit the data well. Therefore, the PSUMMARY option would serve your purpose well in this example.

In fact, even the fit summary table can be trimmed down quite a bit if you only want to look at certain specific fit indices. In the FITINDEX statement of this example, the ON(ONLY)= option turns on the printing of the model fit chi-square, its df, and p-value only. This does not mean that you must lose the information of all other fit indices. In addition to the printed output, you can save all fit indices in an output data set. To this end, you can use the OUTFIT= option in the FITINDEX statement. In this example, you save the results of all fit indices in a SAS data set called savefit.

Output 26.18.1 shows the entire printed output.

Output 26.18.1 Testing Direct Covariance Structures

Fit Summary

Chi-Square	221.5798
Chi-Square DF	5
Pr > Chi-Square	<.0001

The displayed output is very concise. It contains only a fit summary table with three statistics. The p-value for the model fit chi-square test indicates that the hypothesized structures should be rejected at $\alpha = 0.05$.

Therefore, this rather restrictive direct covariance structure model does not fit the data well. A less restrictive covariance structure model is needed to explain the variances and covariances.

All fit indices are saved in the savefit data set. To view it, you can use the following statement:

```
proc print data=savefit;
run;
```

Output 26.18.2 shows all indices, their types and values of all fit indices and information.

Output 26.18.2 Saved Fit Indices

Analysis of Direct Covariance Structures
Testing Model by the MSTRUCT Language

Obs	_TYPE_	Index Code	FitIndex	Fit Value	PrintChar
1	ModelInfo	101	N Observations	932.00	932
2	ModelInfo	103	N Variables	4.00	4
3	ModelInfo	104	N Moments	10.00	10
4	ModelInfo	105	N Parameters	5.00	5
5	ModelInfo	106	N Active Constraints	0.00	0
6	ModelInfo	111	Baseline Model Function Value	1.68	1.6799
7	ModelInfo	113	Baseline Model Chi-Square	1563.94	1563.9442
8	ModelInfo	114	Baseline Model Chi-Square DF	6.00	6
9	ModelInfo	115	Pr > Baseline Model Chi-Square	0.00	<.0001
10	Absolute	201	Fit Function	0.24	0.2380
11	Absolute	203	Chi-Square	221.58	221.5798
12	Absolute	204	Chi-Square DF	5.00	5
13	Absolute	205	Pr > Chi-Square	0.00	<.0001
14	Absolute	211	Z-Test of Wilson & Hilferty	12.25	12.2533
15	Absolute	212	Hoelter Critical N	47.00	47
16	Absolute	213	Root Mean Square Residual (RMSR)	0.76	0.7649
17	Absolute	214	Standardized RMSR (SRMSR)	0.07	0.0701
18	Absolute	215	Goodness of Fit Index (GFI)	0.90	0.9036
19	Parsimony	301	Adjusted GFI (AGFI)	0.81	0.8071
20	Parsimony	302	Parsimonious GFI	0.75	0.7530
21	Parsimony	303	RMSEA Estimate	0.22	0.2157
22	Parsimony	304	RMSEA Lower 90% Confidence Limit	0.19	0.1920
23	Parsimony	305	RMSEA Upper 90% Confidence Limit	0.24	0.2404
24	Parsimony	306	Probability of Close Fit	0.00	<.0001
25	Parsimony	307	ECVI Estimate	0.25	0.2488
26	Parsimony	308	ECVI Lower 90% Confidence Limit	0.20	0.2003
27	Parsimony	309	ECVI Upper 90% Confidence Limit	0.31	0.3053
28	Parsimony	310	Akaike Information Criterion	231.58	231.5798
29	Parsimony	311	Bozdogan CAIC	260.77	260.7665
30	Parsimony	312	Schwarz Bayesian Criterion	255.77	255.7665
31	Parsimony	313	McDonald Centrality	0.89	0.8903
32	Incremental	401	Bentler Comparative Fit Index	0.86	0.8610
33	Incremental	402	Bentler-Bonett NFI	0.86	0.8583
34	Incremental	403	Bentler-Bonett Non-normed Index	0.83	0.8332
35	Incremental	404	Bollen Normed Index Rho1	0.83	0.8300
36	Incremental	405	Bollen Non-normed Index Delta2	0.86	0.8611
37	Incremental	406	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 calis nobs=932 data=Wheaton psummary;
   mstruct
      var = Anomie67 Powerless67 Anomie71 Powerless71;
   matrix _COV_ [1,1] = phi1 phi2 phi1 phi2,
                [2, ] = theta1,
                [3, ] = theta2 theta1,
                [4, ] = theta1 theta3 theta1;
   fitindex on(only)=[chisq df probchi] outfit=savefit;
run;
```

In the first entry of the MATRIX statement, the notation [1,1] represents that the parameter list specified after the equal sign starts with the [1,1] element of the _COV_ matrix and proceeds down the diagonal. In the next three entries, the notations [2,], [3,], and [4,] represent that parameter lists start with the first elements of the second, third, and fourth rows, respectively, and proceed to the next (right) elements on the same rows. See the syntax of the MATRIX statement on page 1111 for more details about this kind of shorthand notation.

This example shows how you can use the MSTRUCT modeling language to test specific covariance patterns. You need to define the parameters of the covariance patterns explicitly by the MATRIX statements. See Example 26.4 and Example 26.20 for more applications.

However, some commonly-used covariance and mean patterns are built into PROC CALIS. For these covariance and mean patterns, you can simply use the COVPATTERN= and the MEANPATTERN= options without the need to specify the parameters in the MATRIX statements. See the COVPATTERN= and the MEANPATTERN= options for the supported covariance and mean patterns. See Example 26.5 and Example 26.21 for applications.

Example 26.19: 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:

```
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
;
```

Confirmatory Factor Model with Uncorrelated Factors

You first fit a confirmatory factor model with uncorrelated factors to the data, as shown in the following statements:

```
proc calis data=cognitive1 nobs=64 modification;
   factor
      Read_Factor   ---> reading1-reading3 ,
      Math_Factor   ---> math1-math3       ,
      Write_Factor  ---> writing1-writing3 ;
   pvar
      Read_Factor Math_Factor Write_Factor = 3 * 1.;
   cov
      Read_Factor Math_Factor Write_Factor = 3 * 0.;
run;
```

In the PROC CALIS statement, the number of observations is specified with the NOBS= option. With the MODIFICATION in the PROC CALIS statement, LM (Lagrange Multiplier) tests are conducted. The results of LM tests can suggest the inclusion of additional parameters for a better model fit.

The FACTOR modeling language is most handy when you specify confirmatory factor models. You use the FACTOR statement to invoke the FACTOR modeling language. Entries in the FACTOR statement are for specifying factor-variables relationships and are separated by commas. In each entry, you first specify a latent factor, followed by the right arrow sign ---> (you can use >, ->, -->, or --->). Then you specify the observed variables that have nonzero loadings on the factor. For example, in the first entry of FACTOR statement, you specify that latent factor Read_Factor has nonzero loadings (free parameters) on variables reading1–reading3. Optionally, you can specify the parameter list after you specify the factor-variable relationships. For example, you can name the loading parameters as in the following specification:

```
factor
   Read_Factor   ---> reading1-reading3    = load1-load3;
```

This way, you name the factor loadings with parameter names load1, load2, and load3, respectively. However, in the current example, because the loading parameters are all unconstrained, you can just let PROC CALIS to generate the parameter names for you. In this example, there are three factors: Read_Factor, Math_Factor, and Write_Factor. These factors have simple cluster structures with the nine observed variables. Each observed variable has only one loading on exactly one factor.

In the PVAR statement, you can specify the variances of the factors and the error variances of the observed variables. The factor variances in this model are all fixed at 1.0 for identification purposes. You do not need to specify the error variances of the observed variables in the current model because PROC CALIS assumes these are free parameters by default.

In the COV statement, you specify that the covariances among the factors are fixed zeros. There are three covariances among the three latent factors and therefore you put 3 * 0. for their fixed values. This means that the factors in the current model are uncorrelated. Note that you must specify uncorrelated factors explicitly in the COV statement because all latent factors are correlated by default.

In Output 26.19.1, the initial model specification is echoed in matrix form. The observed variables and factors are also displayed.

Output 26.19.1 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
```

Output 26.19.1 *continued*

Initial Factor Loading Matrix

	Read_Factor	Math_Factor	Write_Factor
reading1	. [_Parm1]	0	0
reading2	. [_Parm2]	0	0
reading3	. [_Parm3]	0	0
math1	0	. [_Parm4]	0
math2	0	. [_Parm5]	0
math3	0	. [_Parm6]	0
writing1	0	0	. [_Parm7]
writing2	0	0	. [_Parm8]
writing3	0	0	. [_Parm9]

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

Initial Error Variances

Variable	Parameter	Estimate
reading1	_Add1	.
reading2	_Add2	.
reading3	_Add3	.
math1	_Add4	.
math2	_Add5	.
math3	_Add6	.
writing1	_Add7	.
writing2	_Add8	.
writing3	_Add9	.

NOTE: Parameters with prefix '_Add' are added by PROC CALIS.

In the table for initial factor loading matrix, the nine loading parameters are shown to have simple cluster relations with the factors. In the table for initial factor covariance matrix, the diagonal matrix shows that the factors are not correlated. The diagonal elements are fixed at ones so that this matrix is also a correlation matrix for the factors. In the table for initial error variances, the nine variance parameters are shown. As described previously, these error variances are generated by PROC CALIS as default parameters.

In Output 26.19.2, initial estimates are generated by the instrumental variable method and the McDonald method.

Output 26.19.2 Optimization of the Uncorrelated Factor Model: Initial Estimates

Initial Estimation Methods

1	Instrumental Variables Method
2	McDonald Method

Optimization Start
Parameter Estimates

N	Parameter	Estimate	Gradient
1	_Parm1	7.15372	0.00851
2	_Parm2	7.80225	-0.00170
3	_Parm3	8.70856	-0.00602
4	_Parm4	7.68637	0.00272
5	_Parm5	8.01765	-0.01096
6	_Parm6	7.05012	0.00932
7	_Parm7	8.76776	-0.0009955
8	_Parm8	5.96161	-0.01335
9	_Parm9	7.23168	0.01665
10	_Add1	31.84831	-0.00179
11	_Add2	47.36790	0.0003461
12	_Add3	23.50199	0.00257
13	_Add4	23.13374	-0.0008384
14	_Add5	31.84224	0.00280
15	_Add6	38.92075	-0.00167
16	_Add7	13.86035	-0.00579
17	_Add8	61.00217	0.00115
18	_Add9	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 26.19.3.

Output 26.19.3 Optimization of the Uncorrelated Factor Model: Iteration Summary

Iter	Restarts	Func Calls	Act Con	Objective Function	Obj Fun Change	Max Abs Gradient Element	Lambda	Actual Over Pred 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 26.19.4.

Output 26.19.4 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
	Baseline Model Function Value	4.3182
	Baseline Model Chi-Square	272.0467
	Baseline Model Chi-Square DF	36
	Pr > Baseline Model Chi-Square	<.0001
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	52
	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	85.3752
	Bozdogan CAIC	142.2351
	Schwarz Bayesian Criterion	124.2351
	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 nonnormed index delta2 are both above 0.90. However, many other do not attain this 0.90 level. For example, adjusted GFI is only 0.759. It is thus safe to conclude that there could be some improvements on the model fit.

The MODIFICATION option in the PROC CALIS statement has been used to request for computing the LM test indices for model modifications. The results are shown in Output 26.19.5.

Output 26.19.5 Lagrange Multiplier Tests

Rank Order of the 10 Largest LM Stat for Factor Loadings

Variable	Factor	LM Stat	Pr > ChiSq	Parm 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

Var1	Var2	LM Stat	Pr > ChiSq	Parm 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 of	Error of	LM Stat	Pr > ChiSq	Parm 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 26.19.5 shows the new parameters in factor covariances, error variances, or error covariances that would result in a better model fit. The table for the new parameters of the factor covariance matrix indicates that adding each of the covariances among factors might lead to a statistically significant improvement in model fit. The largest LM Stat value in this table is 8.95, which is smaller than that of the largest LM Stat for the factor loading parameters. Despite this, it is more reasonable to add the covariance parameters among factors first to determine whether that improves the model fit.

Confirmatory Factor Model with Correlated Factors

To fit the corresponding confirmatory factor model with correlated factors, you can remove the fixed zeros from the COV statement in the preceding specification, as shown in the following statements:

```
proc calis data=cognitive1 nobs=64 modification;
   factor
      Read_Factor   ---> reading1-reading3 ,
      Math_Factor   ---> math1-math3       ,
      Write_Factor  ---> writing1-writing3 ;
   pvar
      Read_Factor Math_Factor Write_Factor = 3 * 1.;
   cov
      Read_Factor Math_Factor Write_Factor /* = 3 * 0. */;
run;
```

In the COV statement, you comment out the fixed zeros so that the covariances among the latent factors are now free parameters. An alternative way is to delete the entire COV statement so that the covariances among factors are free parameters by the FACTOR model default.

The fit summary of the correlated factor model is shown in Output 26.19.6.

Output 26.19.6 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
	Baseline Model Function Value	4.3182
	Baseline Model Chi-Square	272.0467
	Baseline Model Chi-Square DF	36
	Pr > Baseline Model Chi-Square	<.0001
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	78
	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	0.0000
	RMSEA Upper 90% Confidence Limit	0.1244
	Probability of Close Fit	0.3814
	ECVI Estimate	1.2602
	ECVI Lower 90% Confidence Limit	1.2453
	ECVI Upper 90% Confidence Limit	1.5637
	Akaike Information Criterion	71.4667
	Bozdogan CAIC	137.8032
	Schwarz Bayesian Criterion	116.8032
	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 29.47, which is about 20 less than the model with uncorrelated factors. The p-value is 0.20, indicating a satisfactory model fit. The RMSEA value is 0.06, which is close to 0.05, a value recommended as an indication of good model fit by Browne and Cudeck (1993) More fit indices that do not attain the 0.9 level with the uncorrelated factor model now have values close to or above 0.9. These include the goodness-of-fit index (GFI), McDonald centrality, Bentler-Bonnet NFI, and Bentler-Bonnet nonnormed index. By all counts, the correlated factor model is a much better fit than the uncorrelated factor model.

In Output 26.19.7, the estimation results for factor loadings are shown. All these loadings are statistically significant, indicating non-chance relationships with the factors.

Output 26.19.7 Estimation of the Factor Loading Matrix

Factor Loading Matrix: Estimate/StdErr/t-value

	Read_Factor	Math_Factor	Write_Factor
reading1	6.7657 1.0459 6.4689 [_Parm01]	0	0
reading2	7.8579 1.1890 6.6090 [_Parm02]	0	0
reading3	9.1344 1.0712 8.5269 [_Parm03]	0	0
math1	0	7.5488 1.0128 7.4536 [_Parm04]	0
math2	0	8.4401 1.0838 7.7874 [_Parm05]	0
math3	0	6.8194 1.0910 6.2506 [_Parm06]	0
writing1	0	0	7.9677 1.1254 7.0797 [_Parm07]
writing2	0	0	6.8742 1.1986 5.7350 [_Parm08]
writing3	0	0	7.0949 1.2057 5.8844 [_Parm09]

In Output 26.19.8, 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 26.19.8 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 0.1311 2.4955 [_Parm10]	0.4810 0.1208 3.9813 [_Parm11]
Math_Factor	0.3272 0.1311 2.4955 [_Parm10]	1.0000	0.3992 0.1313 3.0417 [_Parm12]
Write_Factor	0.4810 0.1208 3.9813 [_Parm11]	0.3992 0.1313 3.0417 [_Parm12]	1.0000

In Output 26.19.9, the error variances for variables are shown.

Output 26.19.9 Estimation of the Error Variances

Error Variances

Variable	Parameter	Estimate	Standard Error	t Value
reading1	_Add1	37.24939	8.33997	4.46637
reading2	_Add2	46.49695	10.69869	4.34604
reading3	_Add3	15.90447	9.26097	1.71737
math1	_Add4	25.22889	7.72269	3.26685
math2	_Add5	24.89032	8.98327	2.77074
math3	_Add6	42.12110	9.20362	4.57658
writing1	_Add7	27.24965	10.36489	2.62903
writing2	_Add8	49.28881	11.39812	4.32429
writing3	_Add9	48.10684	11.48868	4.18733

All t values except the one for reading3 are greater than 2, a value close to a critical t value at $\alpha = 0.05$. This means that the error variance for reading3 could have been zero in the population, or it could have been nonzero but the current sample just has this insignificant value by chance (that is, a Type 2 error). Further research is needed to confirm either way.

In addition to the parameter estimation results, PROC CALIS also outputs supplementary results that could be useful for interpretations. In Output 26.19.10, the squared multiple correlations and the factor scores regression coefficients are shown.

Output 26.19.10 Supplementary Estimation Results

Squared Multiple Correlations

Variable	Error Variance	Total 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	0.0200	0.000681	0.001985
reading2	0.0186	0.000633	0.001847
reading3	0.0633	0.002152	0.006275
math1	0.001121	0.0403	0.002808
math2	0.001271	0.0457	0.003183
math3	0.000607	0.0218	0.001520
writing1	0.003195	0.002744	0.0513
writing2	0.001524	0.001309	0.0245
writing3	0.001611	0.001384	0.0259

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 26.20: Testing Equality of Two Covariance Matrices Using a Multiple-Group Analysis

You can use PROC CALIS to do multiple-group or multiple-sample analysis. The groups in the analysis must be independent. In this example, a relatively simple multiple-group analysis is carried out. The covariance matrices of two independent groups are tested for equality. Hence, individual covariance matrices are actually not structured. Rather, they are constrained to be the same under the null hypothesis. That is, you want to test the following null hypothesis:

$$H_0 : \mathbf{\Sigma}_1 = \mathbf{\Sigma}_2$$

where $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ represent the population covariance matrices of the two independent groups in question.

In PROC CALIS, you can use two different approaches to test the equality of covariance matrices. The first approach is to define an MSTRUCT model explicitly and to fit this model to the independent groups. The second approach is to use the COVPATTERN= option to invoke the required covariance structure model for the independent groups. Some standard covariance structures or patterns with the MSTRUCT modeling language are built into PROC CALIS internally. With appropriate keywords for the COVPATTERN= option, you can invoke the target built-in covariance patterns without defining the MSTRUCT model explicitly. This example considers these two approaches successively.

This example is concerned with a reaction time experiment that was conducted on two groups of individuals. One group ($N = 20$) was considered to be an expert group with prior training related to the tasks of the experiment. Another group ($N = 18$) was a control group without prior training. Three tasks of dexterity were administered to all individuals. These tasks differed by their required complexity levels of body skills. They were labeled as high, medium, and low complexities.

Apparently, the differential performance of the two groups under different task complexities was the primary research objective. In this example, however, you are interested in testing whether the groups have the same covariance matrix for the tasks. Equality of covariance matrices might be an essential assumption in some statistical tests for comparing group means. In this example, the sample covariance matrices for the two groups are stored in the data sets Expert and Novice, as shown in the following:

```
data expert(type=cov);
   input _type_ $ _name_ $ high medium low;
   datalines;
COV   high    5.88     .       .
COV   medium  2.88    7.16     .
COV   low     3.12    4.44   8.14
;

data novice(type=cov);
   input _type_ $ _name_ $ high medium low;
   datalines;
COV   high    6.42     .       .
COV   medium  1.24    8.25     .
COV   low     4.26    2.75   7.99
;
```

These data sets are read into the analysis through the GROUP statements in the following PROC CALIS specification:

```
proc calis;
   group 1 / data=expert nobs=20 label="Expert";
   group 2 / data=novice nobs=18 label="Novice";
   model 1 / groups=1,2;
      mstruct
         var=high medium low;
   fitindex NoIndexType On(only)=[chisq df probchi]
            chicorrect=eqcovmat;
   ods select ModelingInfo MSTRUCTVariables MSTRUCTCovInit Fit;
   run;
```

The first GROUP statement defines group 1 for the expert group. The second GROUP statement defines group 2 for the novice group. You use the NOBS= option in both statements to provide the number of observations of these groups. You use the LABEL= option in these statements to provide meaningful group labels.

The MODEL statement defines MODEL 1. In the analysis, this model fits to both groups 1 and 2, as indicated by the GROUPS= option of the statement. This is done to test the null hypothesis of equality of covariance matrices in the two groups. An MSTRUCT model for MODEL 1 is defined immediately afterward. Three variables, high, medium, and low, are specified in the VAR= option of the MSTRUCT statement.

Without further specification about the MSTRUCT model, PROC CALIS assumes all non redundant elements in the covariance matrix are free parameters. This is what is required under the null hypothesis of the equality of covariance matrices in the two groups—the groups have the same covariance matrix, but the covariance matrix itself is unconstrained. Your model under the null hypothesis is now well-defined and ready to run. In addition, you use FITINDEX and ODS SELECT statements to customize or fine tune the analysis.

By using the options in the FITINDEX statement, you can customize the fit summary table and control some analytic options. In the current example, you use the NOINDEXTYPE option to suppress the printing of the index types in the fit summary table. Then, you use the ON(ONLY)= option to specify the fit indices printed in the fit summary table. In this example, you request only the model fit chi-square statistic, degrees of freedom, and the probability value of the chi-square be printed. Finally, you use the CHICORRECT=EQCOVMAT option to request a chi-square correction for the test of equality of covariance matrices. This correction is due to Box (1949) and is implemented in PROC CALIS as a built-in chi-square correction option.

In addition, because you are not interested in all displayed output for the current hypothesized model, you use the ODS SELECT statement to display only those output (or ODS tables) of interest. In this example, you request only the modeling information, the variables involved, the initial covariance matrix specification, and the fit summary table be printed. All output in PROC CALIS are named as an ODS table. To locate a particular output in PROC CALIS, you must know the corresponding ODS table name. See the section "ODS Table Names" on page 1298 for a listing of ODS tables produced by PROC CALIS.

Output 26.20.1 displays some information regarding the basic model setup.

Output 26.20.1 Modeling Information and Initial Specification

Modeling Information

Group	Label	Data Set	N Obs	Model	Type	Analysis
1	Expert	WORK.EXPERT	20	Model 1	MSTRUCT	Covariances
2	Novice	WORK.NOVICE	18	Model 1	MSTRUCT	Covariances

Model 1. Variables in the Model

high medium low

Number of Variables = 3

Model 1. Initial MSTRUCT _COV_ Matrix

	high	medium	low
high	. [_Add1]	. [_Add2]	. [_Add4]
medium	. [_Add2]	. [_Add3]	. [_Add5]
low	. [_Add4]	. [_Add5]	. [_Add6]

The modeling information table summarizes some basic information about the two groups. Both of them are fitted by Model 1. The next table shows the variables involved: high, medium, and low. The order of variables in this table is the same as that of the row and column variables of the covariance model matrix, which is shown next in Output 26.20.1. The parameters for the entries in the covariance matrix are shown. The names of parameters are displayed in parentheses. All these parameters are set by default and their names have the prefix _Add. No initial estimates are given as input, as indicated by the missing value '.'.

Output 26.20.2 shows the customized fit summary table, which has been much simplified for the current example due to the uses of some options in the FITINDEX statement.

Output 26.20.2 Model Fit

Fit Summary

Chi-Square	2.4924
Chi-Square DF	6
Pr > Chi-Square	0.8693

As shown in Output 26.20.2, the chi-square test statistic is 2.4924. With six degrees of freedom, the test statistic is not significant at $\alpha = 0.01$. Therefore, the hypothesized model is supported, which means that the equality of the covariance matrices of the groups is supported.

Instead of using the MSTRUCT modeling language explicitly for defining the hypothesized covariance patterns (or structures), you can also invoke the same covariance patterns by using the COVPATTERN= option, as shown in the following statements:

```
proc calis covpattern=eqcovmat;
   var high medium low;
   group 1 / data=expert nobs=20 label="Expert";
   group 2 / data=novice nobs=18 label="Novice";
   fitindex NoIndexType On(only)=[chisq df probchi];
   run;
```

The COVPATTERN=EQCOVMAT option in the PROC CALIS statement hypothesizes that the two population covariance matrices for the groups are the same. Next, you specify the set of variables in the covariance matrices in the VAR statement, followed by the specification of the data for the two groups. You use the FITINDEX statement to select a subset of fit indices to display in the output.

Output 26.20.3 shows the data sets and the corresponding MSTRUCT models that are generated by the COVPATTERN=EQCOVMAT option.

Output 26.20.3 Modeling Information with the COVPATTERN=EQCOVMAT Option

Modeling Information

Group	Label	Data Set	N Obs	Model	Type	Analysis
1	Expert	WORK.EXPERT	20	Model 1	MSTRUCT	Covariances
2	Novice	WORK.NOVICE	18	Model 2	MSTRUCT	Covariances

PROC CALIS generates Model 1 for the expert group and Model 2 for the novice group. Output 26.20.4 and Output 26.20.5 show the covariance matrices of these two models.

Output 26.20.4 Initial Specification of Model 1 for the Expert Group

Model 1. Variables in the Model

high medium low

Number of Variables = 3

Model 1. Initial MSTRUCT _COV_ Matrix

	high	medium	low
high	. [_cov_1_1]	. [_cov_2_1]	. [_cov_3_1]
medium	. [_cov_2_1]	. [_cov_2_2]	. [_cov_3_2]
low	. [_cov_3_1]	. [_cov_3_2]	. [_cov_3_3]

Output 26.20.5 Initial Specification of Model 2 for the Novice Group

```
                    Model 2. Variables in the Model

                          high   medium   low

                       Number of Variables = 3

                 Model 2. Initial MSTRUCT _COV_ Matrix

                           high          medium            low

         high                 .               .               .
                     [_cov_1_1]      [_cov_2_1]      [_cov_3_1]

         medium               .               .               .
                     [_cov_2_1]      [_cov_2_2]      [_cov_3_2]

         low                  .               .               .
                     [_cov_3_1]      [_cov_3_2]      [_cov_3_3]
```

In Output 26.20.4, the covariance matrix for the expert group has three variables: high, medium, and low. The second table of Output 26.20.4 shows the parameters for the corresponding covariance matrix. PROC CALIS generates the parameter names for the elements in this covariance matrix: _cov_1_1, _cov_2_1, . . . , _cov_3_3. In Output 26.20.5, the covariance matrix for the novice group has exactly the same set of three variables: high, medium, and low. The second table of Output 26.20.5 shows the parameters for the corresponding covariance matrix. These variance and covariance parameters are exactly the same as those in Output 26.20.4, as required by the testing of equality of covariance matrices.

Output 26.20.6 shows the fit summary of the test. The test results are exactly the same as those in Output 26.20.2, as expected. The chi-square value is 2.4924. With six degrees of freedom, the test statistic is not significant at $\alpha = 0.01$. The hypothesis about the equality of the covariance matrices between the groups is supported.

Output 26.20.6 Model Fit with the COVPATTERN=EQCOVMAT Option

```
                              Fit Summary

                    Chi-Square                2.4924
                    Chi-Square DF                  6
                    Pr > Chi-Square           0.8693
```

One advantage of using the built-in covariance patterns such as the current COVPATTERN=EQCOVMAT option is that it is more efficient and less error-prone than if you specify the covariance patterns manually by using the MSTRUCT and MATRIX statements. With the COVPATTERN= option, PROC CALIS generates the correct model specification internally. Another advantage is that when applicable, PROC CALIS applies the appropriate chi-square correction to the chi-square test statistic. For the current example, PROC CALIS displays the following message in the output:

```
NOTE: The chi-square correction due to Box for testing equality of
      covariance matrices was applied. Use the CHICORRECT=0 option
      if this correction is not desirable.
```

This shows that when you use the COVPATTERN=EQCOVMAT option, an appropriate chi-square correction is applied automatically to the chi-square test statistic. To turn off this automatic chi-square, you can use the CHICORRECT=0 in the PROC CALIS statement (although this should be a rare practice with the COVPATTERN= options).

To extend the test of the equality of covariance matrices to the test of the equality of mean vectors, see Example 26.4. To extend the multiple-group analysis of covariance patterns to the multiple-group analysis of a general structural equation model, see Example 26.27.

Example 26.21: Testing Equality of Covariance and Mean Matrices between Independent Groups

To make the specification of some standard MSTRUCT models for covariance and mean patterns more efficient, PROC CALIS defines these standard models internally. You can use two options to invoke these built-in covariance and mean patterns easily. For example, with the COVPATTERN= option, you can define the compound symmetry (COMPSYM) pattern for the covariance matrix or the equality of covariance matrices between groups (EQCOVMAT). With the MEANPATTERN= option, you can define uniform means (UNIFORM) for the mean vector or the equality of mean vectors between groups (EQMEANVEC). See the COVPATTERN= and the MEANPATTERN= options for details about the supported built-in covariance and mean patterns.

In Example 26.20, you test of the equality of covariance matrices between two groups. This example extends the application to the test of equality of mean vectors between three independent groups by using the COVPATTERN= and MEANPATTERN= options together. The “best” fit model for the data is explored. The following DATA steps define the covariance and mean matrices for the three independent groups, respectively:

```
data g1(type=corr);
   Input _type_ $ 1-8 _name_ $ 9-11 x1-x9;
   datalines;
corr    x1  1.     .       .      .      .      .      .      .      .
corr    x2 .721    1.      .      .      .      .      .      .      .
corr    x3 .676   .379     1.     .      .      .      .      .      .
corr    x4 .149   .403    .450    1.     .      .      .      .      .
corr    x5 .422   .384    .445   .411    1.     .      .      .      .
corr    x6 .343   .456    .243   .308   .531    1.     .      .      .
corr    x7 .115   .225    .201   .481   .373   .198   1.      .      .
corr    x8 .213   .237    .434   .503   .267   .333   .355   1.      .
corr    x9 .236   .257    .159   .246   .126   .235   .601   .512    1.
mean     . 21.3   22.3    17.2   23.4   22.1   15.6   18.7   20.1  19.7
std      .  1.2    1.4    .87    1.33    2.2    1.4    2.3    2.1   1.8
n        .   21     21      21     21     21     21     21     21    21
;
```

```
data g2(type=corr);
   Input _type_ $ 1-8 _name_ $ 9-11 x1-x9;
   datalines;
corr    x1  1.     .        .      .      .      .      .      .      .
corr    x2 .733    1.       .      .      .      .      .      .      .
corr    x3 .576   .388      1.     .      .      .      .      .      .
corr    x4 .209   .414     .425    1.     .      .      .      .      .
corr    x5 .412   .286     .461   .398    1.     .      .      .      .
corr    x6 .323   .399     .212   .302   .522    1.     .      .      .
corr    x7 .215   .295     .188   .467   .334   .232   1.      .      .
corr    x8 .204   .257     .462   .522   .298   .355  .372    1.      .
corr    x9 .245   .272     .177   .301   .156   .246  .578   .422     1.
mean     . 22.1   19.8     16.9   23.3   21.9   17.3   17.9   19.1  19.8
std      .  1.3    1.3     .99    1.25   2.1    1.3    2.2    2.0   1.5
n        .   22     22       22     22     22     22     22     22    22
;

data g3(type=corr);
   Input _type_ $ 1-8 _name_ $ 9-11 x1-x9;
   datalines;
corr    x1  1.     .        .      .      .      .      .      .      .
corr    x2 .699    1.       .      .      .      .      .      .      .
corr    x3 .488   .328      1.     .      .      .      .      .      .
corr    x4 .235   .398     .413    1.     .      .      .      .      .
corr    x5 .377   .265     .471   .376    1.     .      .      .      .
corr    x6 .335   .412     .265   .314   .503    1.     .      .      .
corr    x7 .243   .216     .192   .423   .369   .212   1.      .      .
corr    x8 .217   .292     .423   .525   .219   .317  .376    1.      .
corr    x9 .211   .283     .152   .285   .147   .135  .633   .579     1.
mean     . 22.2   20.9     15.4   25.1   22.6   16.3   19.3   20.2  19.5
std      .  1.5    1.0     1.04   1.5    1.9    1.6    2.4    2.2   1.6
n        .   20     20       20     20     20     20     20     20    20
;
```

Each of these data sets contains the information about the correlations, means, standard deviations, and sample sizes. Even though these data sets contain correlations, by default PROC CALIS analyzes the covariances and means.

The first hypothesis to test is the equality of covariance matrices and mean vectors:

$$H_0 : \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}_3 \text{ and } \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \boldsymbol{\mu}_3$$

where $\boldsymbol{\Sigma}_1$, $\boldsymbol{\Sigma}_2$, and $\boldsymbol{\Sigma}_3$ are the population covariance matrices for the three independent groups, respectively, and $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, and $\boldsymbol{\mu}_3$ are the population mean vectors for the three independent groups, respectively.

The following statements specify this test:

```
proc calis covpattern=eqcovmat meanpattern=eqmeanvec;
   var x1-x9;
   group 1 / data=g1;
   group 2 / data=g2;
   group 3 / data=g3;
   fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
   run;
```

In the PROC CALIS statement, the COVPATTERN=EQCOVMAT option specifies the same covariance matrix for the three groups and the MEANPATTERN=EQMEANVEC option specifies the same mean vector for the three groups. The VAR statement specifies that x1–9 are the variables in the hypothesis test. Next, the GROUP statements specify the data sets for the three independent groups. You use the FITINDEX statement to limit the amount of output fit statistics to the quantities specified: the chi-square test (CHISQ), the degrees of freedom (DF), the significance value of the test statistic (PROBCHI), the root mean square error approximation (RMSEA), Akaike's information criterion (AIC), consistent Akaike's information criterion (CAIC), and Schwarz's Bayesian criterion (SBC). The first three quantities are useful for the chi-square model fit test, while the rest of the fit indices are useful for comparing competing models for the data. Because there are not many quantities in this customized fit summary table, the NOINDEXTYPE option is used to suppress the printing of the fit index types.

Output 26.21.1 shows the general modeling information, including the sample sizes, the models for the groups, the model types, and the analysis types.

Output 26.21.1 Modeling Information for Testing Equality of Covariance and Mean Matrices

Modeling Information

Group	Data Set	N Obs	Model	Type	Analysis
1	WORK.G1	21	Model 1	MSTRUCT	Means and Covariances
2	WORK.G2	22	Model 2	MSTRUCT	Means and Covariances
3	WORK.G3	20	Model 3	MSTRUCT	Means and Covariances

Output 26.21.2 shows the initial mean vector and the initial covariance matrix specifications for Model 1, which fits to Group 1. PROC CALIS generates the mean parameter names _mean_1, _mean_2, ..., and _mean_9 for the nine elements in the mean vector. It also generates the covariance parameter names _cov_1_1, _cov_2_1, ..., and _cov_9_9 for the 45 nonredundant elements in the covariance matrix.

Output 26.21.2 Initial Mean Vector and Covariance Matrix for Model 1

Model 1. Initial MSTRUCT _MEAN_ Vector

Variable	Parameter	Estimate
x1	_mean_1	.
x2	_mean_2	.
x3	_mean_3	.
x4	_mean_4	.
x5	_mean_5	.
x6	_mean_6	.
x7	_mean_7	.
x8	_mean_8	.
x9	_mean_9	.

Output 26.21.2 *continued*

Model 1. Initial MSTRUCT _COV_ Matrix

	x1	x2	x3	x4	x5
x1	. [_cov_1_1]	. [_cov_2_1]	. [_cov_3_1]	. [_cov_4_1]	. [_cov_5_1]
x2	. [_cov_2_1]	. [_cov_2_2]	. [_cov_3_2]	. [_cov_4_2]	. [_cov_5_2]
x3	. [_cov_3_1]	. [_cov_3_2]	. [_cov_3_3]	. [_cov_4_3]	. [_cov_5_3]
x4	. [_cov_4_1]	. [_cov_4_2]	. [_cov_4_3]	. [_cov_4_4]	. [_cov_5_4]
x5	. [_cov_5_1]	. [_cov_5_2]	. [_cov_5_3]	. [_cov_5_4]	. [_cov_5_5]
x6	. [_cov_6_1]	. [_cov_6_2]	. [_cov_6_3]	. [_cov_6_4]	. [_cov_6_5]
x7	. [_cov_7_1]	. [_cov_7_2]	. [_cov_7_3]	. [_cov_7_4]	. [_cov_7_5]
x8	. [_cov_8_1]	. [_cov_8_2]	. [_cov_8_3]	. [_cov_8_4]	. [_cov_8_5]
x9	. [_cov_9_1]	. [_cov_9_2]	. [_cov_9_3]	. [_cov_9_4]	. [_cov_9_5]

Model 1. Initial MSTRUCT _COV_ Matrix

	x6	x7	x8	x9
x1	. [_cov_6_1]	. [_cov_7_1]	. [_cov_8_1]	. [_cov_9_1]
x2	. [_cov_6_2]	. [_cov_7_2]	. [_cov_8_2]	. [_cov_9_2]
x3	. [_cov_6_3]	. [_cov_7_3]	. [_cov_8_3]	. [_cov_9_3]
x4	. [_cov_6_4]	. [_cov_7_4]	. [_cov_8_4]	. [_cov_9_4]
x5	. [_cov_6_5]	. [_cov_7_5]	. [_cov_8_5]	. [_cov_9_5]
x6	. [_cov_6_6]	. [_cov_7_6]	. [_cov_8_6]	. [_cov_9_6]
x7	. [_cov_7_6]	. [_cov_7_7]	. [_cov_8_7]	. [_cov_9_7]
x8	. [_cov_8_6]	. [_cov_8_7]	. [_cov_8_8]	. [_cov_9_8]
x9	. [_cov_9_6]	. [_cov_9_7]	. [_cov_9_8]	. [_cov_9_9]

Although not shown here, the initial mean vector and covariance matrices for Models 2 and 3 are exactly the same as those shown in Output 26.21.2, as required by the equality of covariance and mean matrices in the null hypothesis H_0.

Output 26.21.3 shows the customized fit summary table. The chi-square test statistic is 203.2605. The degrees of freedom is 108 and the p-value is less than 0.0001. Therefore, the hypothesis H_0 of equality in covariance and mean matrices is rejected for the three independent groups. The RMSEA index is much greater than 0.05, which does not indicate a good model fit. Other fit indices such as AIC, CAIC, and SBC are not interpreted for the fit of the model itself, but are useful for comparing competing models in the later discussion.

Output 26.21.3 Fit Summary for Testing H_0: Equality of Covariance and Mean Matrices

Fit Summary	
Chi-Square	203.2605
Chi-Square DF	108
Pr > Chi-Square	<.0001
RMSEA Estimate	0.2100
Akaike Information Criterion	311.2605
Bozdogan CAIC	480.9897
Schwarz Bayesian Criterion	426.9897

A less restrictive hypothesis is now considered. This hypothesis states the equality of covariance matrices only:

$$H_1 : \mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{\Sigma}_3 (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \text{ and } \boldsymbol{\mu}_3 \text{unconstrained})$$

H_1 differs from H_0 in that the population means in H_1 are not constrained. To test this hypothesis, you need to change the MEANPATTERN= option to use the SATURATED keyword, as shown in the following statements:

```
proc calis covpattern=eqcovmat meanpattern=saturated;
   var x1-x9;
   group 1 / data=g1;
   group 2 / data=g2;
   group 3 / data=g3;
   fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
   run;
```

Output 26.21.4 shows the results of the testing H_1.

Output 26.21.4 Fit Summary for Testing H_1: Equality of Covariance Matrices but Unconstrained Means

Fit Summary	
Chi-Square	26.7897
Chi-Square DF	90
Pr > Chi-Square	1.0000
RMSEA Estimate	0.0000
Akaike Information Criterion	170.7897
Bozdogan CAIC	397.0954
Schwarz Bayesian Criterion	325.0954

The chi-square test statistic is 26.7897 (df=90, p=1.000). You cannot reject this null hypothesis about the equality of the population covariance matrices. The RMSEA value is virtually zero, which indicates a perfect fit. Comparing the models under H_0 and H_1, it is clear that the three groups are significantly different with regard to their mean vectors. By relaxing all the equality constraints on the means in H_0, H_1 is derived and is supported by the chi-square test. In addition, the RMSEA value for the model under H_1 is perfect. Because lower values of AIC, CAIC, and SBC values indicate better model fit (with the model complexity taken into account), these indices in Output 26.21.3 and Output 26.21.4 support that the model under H_1 is better than H_0.

However, in getting a superior model fit, H_1 might have relaxed more constraints than absolutely necessary for an optimal fit. That is, it might be possible to impose equality constraints on only some (but not all, as in H_1) of the means to reach the same or even better model fit (by the RMSEA, AIC, CAIC, or SBC criterion) than the model under H_1. But how can you determine this set of constrained means?

To answer this question, you conduct an exploratory analysis of the data by using some model modification techniques. Models established from exploratory analysis should be validated by external data in the future. However, this example demonstrates the exploratory part only.

Beginning with the model under H_0, you can manually take away some particular constraints on the means and explore whether the revised model improves the fit. If the revised model fits better, you can repeat the process until you cannot improve more. Ultimately, you might be able to find the "best" model between the models specified under H_0 and H_1. Such an exploratory analysis is laborious, considering the vast possibilities of constraints on the nine variable means in three independent groups that you could attempt to release. Fortunately, PROC CALIS provides some model modification statistics, called the LM (Lagrange multiplier) statistics, to assist this kind of exploratory analysis.

The following statements specify the model under H_0, but now with the MODIFICATION option added to the PROC CALIS statement:

```
proc calis covpattern=eqcovmat meanpattern=eqmeanvec modification;
   var x1-x9;
   group 1 / data=g1;
   group 2 / data=g2;
   group 3 / data=g3;
   fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
   run;
```

The MODIFICATION option requests the so-called LM (Lagrange multiplier) statistics for releasing the parameter constraints. These constraints include the cross-group or within-group constraints and the fixed values in the model. For the model under H_0, the covariances and the means are all constrained across groups. These are the equality constraints that you would like to release to obtain a better model fit. Output 26.21.5 shows the results of the LM statistics for releasing these equality constraints in variances, covariances, and means.

Output 26.21.5 Lagrange Multiplier Statistics for Releasing the Equality Constraints

Lagrange Multiplier Statistics for Releasing Equality Constraints

Parm	---Released Parameter-- Model	Type	Var1	Var2	LM Stat	Pr > ChiSq	------Changes----- Original Parm	Released Parm
_cov_1_1	1	COV	x1	x1	0.01137	0.9151	0.0178	-0.0355
	2	COV	x1	x1	1.00150	0.3169	0.1729	-0.3212
	3	COV	x1	x1	1.28632	0.2567	-0.1818	0.3923
_cov_2_1	1	COV	x2	x1	2.19353	0.1386	0.2038	-0.4076
	2	COV	x2	x1	0.77014	0.3802	-0.1253	0.2327
	3	COV	x2	x1	0.36128	0.5478	-0.0796	0.1718
_cov_2_2	1	COV	x2	x2	3.12065	0.0773	-0.4344	0.8687
	2	COV	x2	x2	0.05704	0.8112	-0.0609	0.1132
	3	COV	x2	x2	4.14151	0.0418	0.4817	-1.0395
_cov_3_1	1	COV	x3	x1	0.00672	0.9347	0.00888	-0.0178
	2	COV	x3	x1	2.23758	0.1347	-0.1681	0.3122
	3	COV	x3	x1	2.10455	0.1469	0.1512	-0.3264
_cov_3_2	1	COV	x3	x2	2.18538	0.1393	-0.1940	0.3881
	2	COV	x3	x2	3.14532	0.0761	0.2416	-0.4487
	3	COV	x3	x2	0.10264	0.7487	-0.0405	0.0874
_cov_3_3	1	COV	x3	x3	1.56813	0.2105	0.1815	-0.3630
	2	COV	x3	x3	0.66118	0.4161	0.1223	-0.2272
	3	COV	x3	x3	4.42160	0.0355	-0.2934	0.6332
_cov_4_1	1	COV	x4	x1	0.31691	0.5735	-0.0667	0.1333
	2	COV	x4	x1	0.32615	0.5679	0.0702	-0.1304
	3	COV	x4	x1	0.0002277	0.9880	-0.00172	0.00371
_cov_4_2	1	COV	x4	x2	0.73377	0.3917	0.1242	-0.2484
	2	COV	x4	x2	0.53196	0.4658	-0.1097	0.2038
	3	COV	x4	x2	0.01445	0.9043	-0.0168	0.0362
_cov_4_3	1	COV	x4	x3	0.0000258	0.9959	0.000547	-0.00109
	2	COV	x4	x3	0.24892	0.6178	-0.0558	0.1036
	3	COV	x4	x3	0.25646	0.6126	0.0525	-0.1134
_cov_4_4	1	COV	x4	x4	0.04412	0.8336	0.0361	-0.0722
	2	COV	x4	x4	0.52198	0.4700	0.1288	-0.2392
	3	COV	x4	x4	0.90948	0.3403	-0.1577	0.3403
_cov_5_1	1	COV	x5	x1	0.0008607	0.9766	-0.00477	0.00953
	2	COV	x5	x1	0.01238	0.9114	0.0188	-0.0348
	3	COV	x5	x1	0.00712	0.9328	-0.0132	0.0285
_cov_5_2	1	COV	x5	x2	0.10637	0.7443	-0.0649	0.1297
	2	COV	x5	x2	0.00631	0.9367	-0.0164	0.0304
	3	COV	x5	x2	0.16971	0.6804	0.0789	-0.1702
_cov_5_3	1	COV	x5	x3	0.06645	0.7966	-0.0385	0.0771
	2	COV	x5	x3	0.0008275	0.9771	0.00446	-0.00829
	3	COV	x5	x3	0.05370	0.8167	0.0334	-0.0720
_cov_5_4	1	COV	x5	x4	0.24212	0.6227	0.0809	-0.1617
	2	COV	x5	x4	0.04459	0.8328	-0.0360	0.0669
	3	COV	x5	x4	0.07959	0.7779	-0.0446	0.0963
_cov_5_5	1	COV	x5	x5	0.01778	0.8939	-0.0431	0.0862
	2	COV	x5	x5	0.08223	0.7743	-0.0962	0.1787
	3	COV	x5	x5	0.18417	0.6678	0.1336	-0.2883

Output 26.21.5 *continued*

Lagrange Multiplier Statistics for Releasing Equality Constraints

Parm	---Released Parameter-- Model	Type	Var1	Var2	LM Stat	Pr > ChiSq	------Changes----- Original Parm	Released Parm
_cov_6_1	1	COV	x6	x1	0.29558	0.5867	-0.0721	0.1442
	2	COV	x6	x1	0.26589	0.6061	-0.0710	0.1318
	3	COV	x6	x1	1.16570	0.2803	0.1378	-0.2974
_cov_6_2	1	COV	x6	x2	0.00228	0.9619	-0.00780	0.0156
	2	COV	x6	x2	1.00319	0.3165	0.1697	-0.3152
	3	COV	x6	x2	0.95767	0.3278	-0.1538	0.3320
_cov_6_3	1	COV	x6	x3	1.39116	0.2382	0.1513	-0.3027
	2	COV	x6	x3	0.08741	0.7675	-0.0394	0.0731
	3	COV	x6	x3	0.79586	0.3723	-0.1102	0.2378
_cov_6_4	1	COV	x6	x4	0.46031	0.4975	-0.0947	0.1894
	2	COV	x6	x4	0.04254	0.8366	0.0299	-0.0555
	3	COV	x6	x4	0.22665	0.6340	0.0640	-0.1381
_cov_6_5	1	COV	x6	x5	0.14991	0.6986	-0.0700	0.1399
	2	COV	x6	x5	0.04723	0.8280	0.0408	-0.0757
	3	COV	x6	x5	0.02874	0.8654	0.0295	-0.0636
_cov_6_6	1	COV	x6	x6	0.22550	0.6349	0.1079	-0.2158
	2	COV	x6	x6	0.04390	0.8340	0.0494	-0.0918
	3	COV	x6	x6	0.48451	0.4864	-0.1523	0.3286
_cov_7_1	1	COV	x7	x1	0.50774	0.4761	0.1203	-0.2406
	2	COV	x7	x1	0.01246	0.9111	-0.0196	0.0363
	3	COV	x7	x1	0.36926	0.5434	-0.0988	0.2131
_cov_7_2	1	COV	x7	x2	0.01235	0.9115	0.0228	-0.0455
	2	COV	x7	x2	0.16400	0.6855	-0.0861	0.1598
	3	COV	x7	x2	0.09159	0.7622	0.0597	-0.1288
_cov_7_3	1	COV	x7	x3	0.16844	0.6815	-0.0644	0.1288
	2	COV	x7	x3	0.15095	0.6976	0.0633	-0.1175
	3	COV	x7	x3	0.0003079	0.9860	0.00265	-0.00572
_cov_7_4	1	COV	x7	x4	0.22542	0.6349	-0.0776	0.1551
	2	COV	x7	x4	0.00754	0.9308	0.0147	-0.0273
	3	COV	x7	x4	0.15376	0.6950	0.0617	-0.1331
_cov_7_5	1	COV	x7	x5	0.07831	0.7796	-0.0631	0.1262
	2	COV	x7	x5	0.07552	0.7835	0.0643	-0.1195
	3	COV	x7	x5	3.293E-6	0.9986	0.000394	-0.00085
_cov_7_6	1	COV	x7	x6	0.13810	0.7102	0.0726	-0.1452
	2	COV	x7	x6	0.0001086	0.9917	0.00211	-0.00392
	3	COV	x7	x6	0.14999	0.6985	-0.0729	0.1572
_cov_7_7	1	COV	x7	x7	0.09334	0.7600	0.1051	-0.2101
	2	COV	x7	x7	0.00128	0.9714	0.0128	-0.0237
	3	COV	x7	x7	0.11994	0.7291	-0.1147	0.2474
_cov_8_1	1	COV	x8	x1	0.04800	0.8266	0.0353	-0.0706
	2	COV	x8	x1	0.19725	0.6569	0.0743	-0.1379
	3	COV	x8	x1	0.45888	0.4981	-0.1051	0.2268
_cov_8_2	1	COV	x8	x2	0.13689	0.7114	0.0727	-0.1453
	2	COV	x8	x2	0.31671	0.5736	-0.1147	0.2130
	3	COV	x8	x2	0.04084	0.8398	0.0382	-0.0825

Output 26.21.5 *continued*

Lagrange Multiplier Statistics for Releasing Equality Constraints

	---Released Parameter--						------Changes-----	
Parm	Model	Type	Var1	Var2	LM Stat	Pr > ChiSq	Original Parm	Released Parm
_cov_8_3	1	COV	x8	x3	0.37615	0.5397	-0.0904	0.1808
	2	COV	x8	x3	0.00452	0.9464	-0.0103	0.0191
	3	COV	x8	x3	0.47678	0.4899	0.0980	-0.2114
_cov_8_4	1	COV	x8	x4	0.00989	0.9208	0.0150	-0.0300
	2	COV	x8	x4	0.01001	0.9203	0.0157	-0.0291
	3	COV	x8	x4	0.04138	0.8388	-0.0296	0.0638
_cov_8_5	1	COV	x8	x5	0.01378	0.9066	-0.0267	0.0533
	2	COV	x8	x5	0.03154	0.8590	-0.0419	0.0778
	3	COV	x8	x5	0.09063	0.7634	0.0659	-0.1421
_cov_8_6	1	COV	x8	x6	0.0007193	0.9786	0.00510	-0.0102
	2	COV	x8	x6	0.01293	0.9095	0.0224	-0.0417
	3	COV	x8	x6	0.02067	0.8857	-0.0263	0.0568
_cov_8_7	1	COV	x8	x7	0.16543	0.6842	0.0952	-0.1904
	2	COV	x8	x7	0.29902	0.5845	-0.1328	0.2467
	3	COV	x8	x7	0.02206	0.8819	0.0335	-0.0722
_cov_8_8	1	COV	x8	x8	0.00581	0.9392	0.0244	-0.0487
	2	COV	x8	x8	0.00694	0.9336	-0.0276	0.0513
	3	COV	x8	x8	0.0000660	0.9935	0.00250	-0.00539
_cov_9_1	1	COV	x9	x1	0.19272	0.6607	-0.0532	0.1063
	2	COV	x9	x1	0.01910	0.8901	-0.0174	0.0323
	3	COV	x9	x1	0.34408	0.5575	0.0684	-0.1476
_cov_9_2	1	COV	x9	x2	0.09017	0.7640	-0.0446	0.0892
	2	COV	x9	x2	0.26496	0.6067	0.0794	-0.1474
	3	COV	x9	x2	0.04994	0.8232	-0.0320	0.0690
_cov_9_3	1	COV	x9	x3	0.44236	0.5060	0.0758	-0.1516
	2	COV	x9	x3	0.12761	0.7209	-0.0422	0.0784
	3	COV	x9	x3	0.09470	0.7583	-0.0338	0.0728
_cov_9_4	1	COV	x9	x4	0.04619	0.8298	0.0260	-0.0520
	2	COV	x9	x4	0.22996	0.6316	-0.0602	0.1117
	3	COV	x9	x4	0.07502	0.7842	0.0319	-0.0688
_cov_9_5	1	COV	x9	x5	0.02807	0.8669	0.0279	-0.0557
	2	COV	x9	x5	0.0006585	0.9795	-0.00443	0.00823
	3	COV	x9	x5	0.02058	0.8859	-0.0230	0.0496
_cov_9_6	1	COV	x9	x6	0.03989	0.8417	-0.0282	0.0563
	2	COV	x9	x6	0.15069	0.6979	-0.0568	0.1055
	3	COV	x9	x6	0.36051	0.5482	0.0815	-0.1759
_cov_9_7	1	COV	x9	x7	0.03398	0.8537	-0.0284	0.0567
	2	COV	x9	x7	0.05802	0.8097	0.0385	-0.0714
	3	COV	x9	x7	0.00362	0.9520	-0.00891	0.0192
_cov_9_8	1	COV	x9	x8	0.06050	0.8057	-0.0391	0.0781
	2	COV	x9	x8	0.56151	0.4537	0.1235	-0.2294
	3	COV	x9	x8	0.26945	0.6037	-0.0794	0.1713
_cov_9_9	1	COV	x9	x9	0.13296	0.7154	-0.0655	0.1310
	2	COV	x9	x9	0.00130	0.9712	-0.00673	0.0125
	3	COV	x9	x9	0.16526	0.6844	0.0703	-0.1517

Output 26.21.5 *continued*

Lagrange Multiplier Statistics for Releasing Equality Constraints

	---Released Parameter--						------Changes-----	
Parm	Model	Type	Var1	Var2	LM Stat	Pr > ChiSq	Original Parm	Released Parm
_mean_1	1	MEAN	x1		11.09173	0.0009	0.3453	-0.6906
	2	MEAN	x1		1.21196	0.2709	-0.1184	0.2200
	3	MEAN	x1		5.04550	0.0247	-0.2242	0.4838
_mean_2	1	MEAN	x2		21.46921	<.0001	-0.5837	1.1675
	2	MEAN	x2		15.27776	<.0001	0.5110	-0.9490
	3	MEAN	x2		0.47301	0.4916	0.0834	-0.1800
_mean_3	1	MEAN	x3		4.41967	0.0355	-0.2034	0.4067
	2	MEAN	x3		6.37770	0.0116	-0.2535	0.4708
	3	MEAN	x3		22.27732	<.0001	0.4395	-0.9485
_mean_4	1	MEAN	x4		3.26860	0.0706	0.1904	-0.3807
	2	MEAN	x4		0.03260	0.8567	0.0197	-0.0366
	3	MEAN	x4		4.06935	0.0437	-0.2045	0.4413
_mean_5	1	MEAN	x5		0.22210	0.6374	-0.0681	0.1362
	2	MEAN	x5		1.50172	0.2204	0.1837	-0.3412
	3	MEAN	x5		0.60673	0.4360	-0.1083	0.2338
_mean_6	1	MEAN	x6		1.61486	0.2038	0.1539	-0.3078
	2	MEAN	x6		6.72912	0.0095	-0.3260	0.6055
	3	MEAN	x6		1.88248	0.1701	0.1600	-0.3452
_mean_7	1	MEAN	x7		0.14035	0.7079	-0.0558	0.1116
	2	MEAN	x7		0.11034	0.7398	0.0514	-0.0954
	3	MEAN	x7		0.00153	0.9688	0.00560	-0.0121
_mean_8	1	MEAN	x8		0.12603	0.7226	-0.0510	0.1019
	2	MEAN	x8		1.96607	0.1609	0.2089	-0.3880
	3	MEAN	x8		1.16200	0.2811	-0.1490	0.3215
_mean_9	1	MEAN	x9		0.05301	0.8179	0.0248	-0.0496
	2	MEAN	x9		0.97965	0.3223	-0.1106	0.2054
	3	MEAN	x9		0.61083	0.4345	0.0810	-0.1748

To use the results of this table, you look for parameters that have large LM statistics (in the LM Stat column). Equivalently, you can look for parameters that have small p-values (in the Pr > ChiSq column). Loosely speaking, an LM statistic estimates the reduction of model fit chi-square statistic if you release the constraint on the corresponding parameter. The p-value indicates whether the improvement would be significant. Therefore, releasing those parameters with a high LM statistic and small p-value would be the key to model improvements. Bear in mind that the LM statistics are linear approximations and they might not be very accurate as estimates of the actual model improvement, which could only be accessed when you refit the model with the particular constraint released. Nonetheless, the LM statistics could still be very useful because they show which constraints could potentially improve the model the most.

Output 26.21.5 shows the results from releasing the constraints on the variances and covariances first. Each constrained element of the covariance matrix has three rows, respectively, for the three models (or groups). For example, the first parameter is _cov_1_1, which is the same variance parameter for x1 in the three models. The first row shows that if you release the variance of x1 in Model 1 from the constraint (while keeping the variances of x1 being constrained between Models 2 and 3), the LM statistic is 0.01127, and the corresponding p-value is 0.9155. This means that the model fit improvement would be very small and so you do not expect a significant model fit improvement by releasing this constraint. The columns entitled

"Changes" show the estimated parameter changes in the original parameters (that is, _cov_1_1 for Models 2 and 3) and in the released parameter (that is, the new parameter for the variance of x1 in Model 1) if you release the corresponding equality constraint. These two "Changes" columns are not very useful for the present purpose.

Looking through the results for the variance and covariance constraints, you can see that almost all the associated p-values are large (that is, as compared with the conventional 0.05 level for significance). Therefore, all these constraints on variances and covariances would not improve the model fit significantly. In contrast, the constraints on the means show that several of them could be released for a sizable model fit improvement. The largest LM statistic in the table is the one for _mean_3 in Model 3. The LM statistic is 22.27678 and its corresponding p-value is less than 0.0001. This means that if the mean of x3 in Model 3 were not constrained with the means of x3 in Models 1 and 2, you would have expected a reduction in the model fit chi-square statistic that is estimated at 22.27678. Other notable LM statistics are those for _mean_1 in Model 1, _mean_2 in Model 1 or 2, and _mean_6 in Model 2.

Two important points are noted about the use of the LM statistics. First, the LM statistics are not additive. You cannot expect that the total reduction in model fit chi-square for releasing a particular set of parameter constraints is the sum of the corresponding LM statistics. Second, once you release a particular constraint and refit the model, the LM statistics in the revised model might not follow the same pattern as those LM statistics in the original model. Basically, these are due to the nonlinearity of the fit function and the dependence of the parameter estimates. Therefore, in order to find the best model for the data, it would be more sensible to adopt a one-at-a-time approach to release the constraints. That is, you release one constraint at a time and refit the model to see if you can release more constraints to improve the model fit.

According to the results of LM statistics in Output 26.21.5, you first release the constraint on the _mean_3 parameter, which is for the mean of x3 in Model 3. The following statements fit such a model:

```
proc calis modification;
   var x1-x9;
   group 1 / data=g1;
   group 2 / data=g2;
   group 3 / data=g3;
   model 1 / group = 1;
      mstruct;
      matrix _cov_  = cov01-cov45;
      matrix _mean_ = mean1-mean9;
   model 2 / group = 2;
      refmodel 1;
   model 3 / group = 3;
      refmodel 1;
      renameparm mean3=mean3_mdl3;
   fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
   run;
```

Because the revised model is no longer a supported built-in MSTRUCT model, you cannot use the MEANPATTERN= or the COVPATTERN= options any more. Instead, you now use the MSTRUCT modeling language to specify the covariance and mean patterns. Model 1, which fits to Group 1, is an MSTRUCT model with variance and covariance parameters cov01–cov45 and mean parameters mean1–mean9. Model 2, which fits to Group 2, refers to the specifications of Model 1, as indicated in a REFMODEL statement. Hence, Model 1 and Model 2 are completely constrained in variances, covariances, and means. Model 3, which fits to Group 3, also refers to the specifications of Model 1, as indicated in another REFMODEL statement. However, the RENAMEPARM statement renames the parameter mean3 in the reference model (that is, Model 1) to a new name mean3_mdl3. As a results, all variance, covariance, and mean parameters except one in Model 3 are constrained to be the same as those in Model 1. The mean of x3 in Model 3 is the only parameter that is not constrained with any other parameters. This forms the first revised model from H_0. The MODIFICATION option is specified again to determine whether a further model fit improvement is possible.

Output 26.21.6 shows the modeling information of the first revised model. It shows that Models 2 and 3 make references to Model 1. Therefore, parameters between models are constrained by referencing.

Output 26.21.6 Modeling Information for The First Revised Model

Modeling Information

Group	Data Set	N Obs	Model	Type	Base Model	Analysis
1	WORK.G1	21	Model 1	MSTRUCT		Means and Covariances
2	WORK.G2	22	Model 2	MSTRUCT	Model 1	Means and Covariances
3	WORK.G3	20	Model 3	MSTRUCT	Model 1	Means and Covariances

Output 26.21.7 shows the initial specifications of the means, variances, and covariances in Model 1.

Output 26.21.7 Initial Mean Vector and Covariance Matrix for Model 1 in the First Revised Model

Model 1. Initial MSTRUCT _MEAN_ Vector

Variable	Parameter	Estimate
x1	mean1	.
x2	mean2	.
x3	mean3	.
x4	mean4	.
x5	mean5	.
x6	mean6	.
x7	mean7	.
x8	mean8	.
x9	mean9	.

Output 26.21.7 *continued*

Model 1. Initial MSTRUCT _COV_ Matrix

	x1	x2	x3	x4	x5
x1	. [cov01]	. [cov02]	. [cov04]	. [cov07]	. [cov11]
x2	. [cov02]	. [cov03]	. [cov05]	. [cov08]	. [cov12]
x3	. [cov04]	. [cov05]	. [cov06]	. [cov09]	. [cov13]
x4	. [cov07]	. [cov08]	. [cov09]	. [cov10]	. [cov14]
x5	. [cov11]	. [cov12]	. [cov13]	. [cov14]	. [cov15]
x6	. [cov16]	. [cov17]	. [cov18]	. [cov19]	. [cov20]
x7	. [cov22]	. [cov23]	. [cov24]	. [cov25]	. [cov26]
x8	. [cov29]	. [cov30]	. [cov31]	. [cov32]	. [cov33]
x9	. [cov37]	. [cov38]	. [cov39]	. [cov40]	. [cov41]

Model 1. Initial MSTRUCT _COV_ Matrix

	x6	x7	x8	x9
x1	. [cov16]	. [cov22]	. [cov29]	. [cov37]
x2	. [cov17]	. [cov23]	. [cov30]	. [cov38]
x3	. [cov18]	. [cov24]	. [cov31]	. [cov39]
x4	. [cov19]	. [cov25]	. [cov32]	. [cov40]
x5	. [cov20]	. [cov26]	. [cov33]	. [cov41]
x6	. [cov21]	. [cov27]	. [cov34]	. [cov42]
x7	. [cov27]	. [cov28]	. [cov35]	. [cov43]
x8	. [cov34]	. [cov35]	. [cov36]	. [cov44]
x9	. [cov42]	. [cov43]	. [cov44]	. [cov45]

Output 26.21.8 shows the initial specifications of the means in Model 2. The mean parameters in Model 2 are exactly the same as those in Model 1, as shown in Output 26.21.7. The variance and covariance parameters in Model 2 are also exactly the same as those in Model 1, but are not shown here to conserve space.

Output 26.21.8 Initial Mean Vector for Model 2 in the First Revised Model

Model 2. Initial MSTRUCT _MEAN_ Vector

Variable	Parameter	Estimate
x1	mean1	.
x2	mean2	.
x3	mean3	.
x4	mean4	.
x5	mean5	.
x6	mean6	.
x7	mean7	.
x8	mean8	.
x9	mean9	.

Output 26.21.9 shows the initial specifications of the means in Model 3. All but one mean parameter in Model 3 are exactly the same as those in Models 1 and 2, as shown in Output 26.21.7 and Output 26.21.8, respectively. The mean for x3 in Model 3 is mean3_mdl3, which is now a distinct parameter, and therefore it is not constrained with any other parameters in the first or the second models for Groups 1 or 2. However, the variance and covariance parameters in Model 3 are exactly the same as those in Model 1. They are not shown here to conserve space.

Output 26.21.9 Initial Mean Vector for Model 3 in the First Revised Model

Model 3. Initial MSTRUCT _MEAN_ Vector

Variable	Parameter	Estimate
x1	mean1	.
x2	mean2	.
x3	mean3_mdl3	.
x4	mean4	.
x5	mean5	.
x6	mean6	.
x7	mean7	.
x8	mean8	.
x9	mean9	.

Output 26.21.10 shows the fit summary of the first revised model. The model fit chi-square is 148.8865, which drops quite a bit from the original model under H_0. The p-value of the model fit chi-square is 0.0046, which is statistically significant. The RMSEA value is 0.1399, which is also a sizable improvement. All the AIC, CAIC, and SBC values are reduced, indicating better model fit than the model under H_0.

Output 26.21.10 Fit Summary for the First Revised Model

Fit Summary

Chi-Square	148.8865
Chi-Square DF	107
Pr > Chi-Square	0.0046
RMSEA Estimate	0.1399
Akaike Information Criterion	258.8865
Bozdogan CAIC	431.7589
Schwarz Bayesian Criterion	376.7589

Output 26.21.11 shows the LM statistics for releasing the equality constraints in the first revised model. Almost all of the results for the variance and covariance constraints are omitted because their LM statistics are not significant. However, Output 26.21.11 shows all the LM statistics for releasing the constraints in means. The mean of x2 in Model 2 has the largest LM statistic at 26.25044.

Output 26.21.11 LM Statistics for Releasing the Equality Constraints in the First Revised Model

Lagrange Multiplier Statistics for Releasing Equality Constraints

	---Released Parameter--						------Changes-----	
Parm	Model	Type	Var1	Var2	LM Stat	Pr > ChiSq	Original Parm	Released Parm
cov01	1	COV	x1	x1	0.64995	0.4201	0.1050	-0.2100
	2	COV	x1	x1	0.41761	0.5181	0.0874	-0.1622
	3	COV	x1	x1	2.18920	0.1390	-0.1855	0.4004
					.			
					.			
					.			
mean1	1	MEAN	x1		9.26683	0.0023	0.2872	-0.5745
	2	MEAN	x1		3.00586	0.0830	-0.1702	0.3160
	3	MEAN	x1		2.13803	0.1437	-0.1481	0.3196
mean2	1	MEAN	x2		26.25110	<.0001	-0.6568	1.3135
	2	MEAN	x2		12.34633	0.0004	0.4674	-0.8680
	3	MEAN	x2		2.52684	0.1119	0.1962	-0.4234
mean3	1	MEAN	x3		0.58886	0.4429	-0.0787	0.0828
	2	MEAN	x3		0.58886	0.4429	0.0828	-0.0787
mean4	1	MEAN	x4		6.59009	0.0103	0.2746	-0.5493
	2	MEAN	x4		0.51348	0.4736	0.0796	-0.1478
	3	MEAN	x4		11.61626	0.0007	-0.3586	0.7739
mean5	1	MEAN	x5		0.52966	0.4668	-0.1042	0.2084
	2	MEAN	x5		0.22296	0.6368	0.0702	-0.1304
	3	MEAN	x5		0.06887	0.7930	0.0374	-0.0807
mean6	1	MEAN	x6		1.16656	0.2801	0.1270	-0.2540
	2	MEAN	x6		5.29612	0.0214	-0.2810	0.5219
	3	MEAN	x6		1.69419	0.1930	0.1518	-0.3275
mean7	1	MEAN	x7		0.03791	0.8456	-0.0291	0.0582
	2	MEAN	x7		0.44509	0.5047	0.1036	-0.1923
	3	MEAN	x7		0.23803	0.6256	-0.0704	0.1520
mean8	1	MEAN	x8		0.39418	0.5301	-0.0883	0.1765
	2	MEAN	x8		0.24234	0.6225	0.0719	-0.1335
	3	MEAN	x8		0.01950	0.8890	0.0200	-0.0431
mean9	1	MEAN	x9		0.00156	0.9685	0.00423	-0.00846
	2	MEAN	x9		1.06869	0.3012	-0.1150	0.2136
	3	MEAN	x9		1.05212	0.3050	0.1065	-0.2297

You now modify the preceding statements to specify the second revised model, as shown in the following statements:

```
proc calis modification;
   var x1-x9;
   group 1 / data=g1;
   group 2 / data=g2;
   group 3 / data=g3;
   model 1 / group = 1;
      mstruct;
      matrix _cov_  = cov01-cov45;
      matrix _mean_ = mean1-mean9;
   model 2 / group = 2;
      refmodel 1;
      renameparm mean2=mean2_new;     /* constraint a */
   model 3 / group = 3;
      refmodel 1;
      renameparm mean2=mean2_new,     /* constraint a */
                 mean3=mean3_mdl3;
   fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
   run;
```

This second revised model must not constrain the mean of x2 in Model 1 with any parameters. A straightforward way to do this is to rename the mean2 parameter to a unique name in Model 1. However, for the current specification it is more convenient to rename the mean2 parameter in Models 2 and 3 to another name. In the specification of the second revised model, Models 2 and 3 still make references to Model 1. However, in the respective RENAMEPARM statements, both Model 2 and 3 rename the mean2 parameter that is referenced from Model 1 to the new name mean2_new. This way the mean for x2 in Model 1 is not constrained with the means of x2 in Models 2 and 3. But the means for x2 in Models 2 and 3 are still constrained to be equal by the same parameter mean2_new. Output 26.21.12 shows the fit summary of the second revised model.

Output 26.21.12 Fit Summary for the Second Revised Model

Fit Summary

Chi-Square	86.3927
Chi-Square DF	106
Pr > Chi-Square	0.9183
RMSEA Estimate	0.0000
Akaike Information Criterion	198.3927
Bozdogan CAIC	374.4083
Schwarz Bayesian Criterion	318.4083

Again, a sizable improvement over the first revised model is shown in the second revised model. The model fit chi-square statistic is no longer significant (p=0.9183), and the RMSEA value is perfect at 0. Large drops in the AIC, CAIC, and SBC values are also observed.

Output 26.21.13 suggests that the mean of x6 in Model 2 (which has the largest LM statistic at 11.41243) could be released from the equality constraints to achieve the largest model improvement over the current model.

Output 26.21.13 LM Statistics for Releasing the Equality Constraints in the Second Revised Model

Lagrange Multiplier Statistics for Releasing Equality Constraints

Parm	---Released Parameter--				LM Stat	Pr > ChiSq	------Changes----- Original Parm	Released Parm
	Model	Type	Var1	Var2				
cov01	1	COV	x1	x1	2.77017	0.0960	0.1384	-0.2769
	2	COV	x1	x1	0.28728	0.5920	0.0463	-0.0859
	3	COV	x1	x1	5.00080	0.0253	-0.1791	0.3864
					.			
					.			
					.			
mean1	1	MEAN	x1		2.75497	0.0970	0.1646	-0.3293
	2	MEAN	x1		3.21108	0.0731	-0.1511	0.2806
	3	MEAN	x1		0.24911	0.6177	0.0424	-0.0915
mean3	1	MEAN	x3		0.74340	0.3886	-0.0877	0.0934
	2	MEAN	x3		0.74340	0.3886	0.0934	-0.0877
mean4	1	MEAN	x4		6.17507	0.0130	0.2672	-0.5343
	2	MEAN	x4		0.02088	0.8851	-0.0146	0.0272
	3	MEAN	x4		4.71373	0.0299	-0.2072	0.4470
mean5	1	MEAN	x5		1.65520	0.1983	-0.1853	0.3706
	2	MEAN	x5		1.16123	0.2812	0.1606	-0.2982
	3	MEAN	x5		0.04040	0.8407	0.0287	-0.0618
mean6	1	MEAN	x6		5.03837	0.0248	0.2712	-0.5423
	2	MEAN	x6		11.41259	0.0007	-0.4217	0.7831
	3	MEAN	x6		1.51178	0.2189	0.1460	-0.3150
mean7	1	MEAN	x7		0.32382	0.5693	-0.0853	0.1706
	2	MEAN	x7		0.82183	0.3646	0.1410	-0.2619
	3	MEAN	x7		0.12512	0.7235	-0.0512	0.1104
mean8	1	MEAN	x8		2.39206	0.1220	-0.2210	0.4420
	2	MEAN	x8		1.58297	0.2083	0.1867	-0.3467
	3	MEAN	x8		0.08639	0.7688	0.0427	-0.0922
mean9	1	MEAN	x9		0.00682	0.9342	0.00886	-0.0177
	2	MEAN	x9		1.20949	0.2714	-0.1225	0.2274
	3	MEAN	x9		1.10018	0.2942	0.1089	-0.2349
mean2_new	2	MEAN	x2		4.47808	0.0343	0.2983	-0.2661
	3	MEAN	x2		4.47808	0.0343	-0.2661	0.2983

The process of model refitting should now become familiar. You modify the previous model to release the constraint on the mean of x6 in Model 2. As a result, the third revised model is specified by the following statements:

```
proc calis modification;
   var x1-x9;
   group 1 / data=g1;
   group 2 / data=g2;
   group 3 / data=g3;
   model 1 / group = 1;
      mstruct;
      matrix _cov_  = cov01-cov45;
      matrix _mean_ = mean1-mean9;
   model 2 / group = 2;
      refmodel 1;
      renameparm mean2=mean2_new,      /* constraint a */
                 mean6=mean6_mdl2;
   model 3 / group = 3;
      refmodel 1;
      renameparm mean2=mean2_new,      /* constraint a */
                 mean3=mean3_mdl3;
   fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
   run;
```

The only modification from the previous specification is to rename mean6 to mean6_mdl2 in the RENAMEPARM statement of Model 2. Output 26.21.14 shows the model fit summary of the third revised model.

Output 26.21.14 Fit Summary for the Third Revised Model

Fit Summary	
Chi-Square	68.7869
Chi-Square DF	105
Pr > Chi-Square	0.9976
RMSEA Estimate	0.0000
Akaike Information Criterion	182.7869
Bozdogan CAIC	361.9456
Schwarz Bayesian Criterion	304.9456

The model improvement over the second revised model is still notable in the third revised model. The chi-square value drops about 20 points in the third revised model. The AIC, CAIC, and the SBC values are reduced notably, though not as impressively as with the previous improvements.

Output 26.21.15 suggests that the mean of x4 in Model 1 (which has the largest LM statistic at 7.01946) could be released from the equality constraint to improve model fit further.

Output 26.21.15 LM Statistics for Releasing the Equality Constraints in the Third Revised Model

Lagrange Multiplier Statistics for Releasing Equality Constraints

Parm	---Released Parameter--				LM Stat	Pr > ChiSq	------Changes----- Original Parm	Released Parm
	Model	Type	Var1	Var2				
cov01	1	COV	x1	x1	2.43365	0.1188	0.1342	-0.2684
	2	COV	x1	x1	0.19040	0.6626	0.0389	-0.0724
	3	COV	x1	x1	4.11405	0.0425	-0.1680	0.3624
					.			
					.			
					.			
mean1	1	MEAN	x1		6.15753	0.0131	0.2550	-0.5101
	2	MEAN	x1		6.05749	0.0138	-0.2109	0.3917
	3	MEAN	x1		0.29286	0.5884	0.0463	-0.0999
mean3	1	MEAN	x3		2.89778	0.0887	-0.1796	0.1889
	2	MEAN	x3		2.89778	0.0887	0.1889	-0.1796
mean4	1	MEAN	x4		7.01943	0.0081	0.2850	-0.5701
	2	MEAN	x4		0.04915	0.8245	-0.0226	0.0419
	3	MEAN	x4		5.05137	0.0246	-0.2148	0.4635
mean5	1	MEAN	x5		0.21229	0.6450	-0.0672	0.1345
	2	MEAN	x5		0.07499	0.7842	-0.0443	0.0822
	3	MEAN	x5		0.55019	0.4582	0.1059	-0.2285
mean6	1	MEAN	x6		0.07011	0.7912	0.0503	-0.0486
	3	MEAN	x6		0.07011	0.7912	-0.0486	0.0503
mean7	1	MEAN	x7		0.98902	0.3200	-0.1513	0.3025
	2	MEAN	x7		2.42349	0.1195	0.2463	-0.4575
	3	MEAN	x7		0.34228	0.5585	-0.0857	0.1850
mean8	1	MEAN	x8		1.58469	0.2081	-0.1786	0.3572
	2	MEAN	x8		0.81644	0.3662	0.1347	-0.2502
	3	MEAN	x8		0.14494	0.7034	0.0549	-0.1184
mean9	1	MEAN	x9		0.13501	0.7133	0.0398	-0.0797
	2	MEAN	x9		2.54369	0.1107	-0.1796	0.3335
	3	MEAN	x9		1.61691	0.2035	0.1337	-0.2886
mean2_new	2	MEAN	x2		3.21187	0.0731	0.2484	-0.2280
	3	MEAN	x2		3.21187	0.0731	-0.2280	0.2484

To make the mean parameter for x4 in Model 1 unique, the mean parameters for x4 in Models 2 and 3 are renamed from mean4 to mean4_new, as shown in the following statements:

```
proc calis modification;
   var x1-x9;
   group 1 / data=g1;
   group 2 / data=g2;
   group 3 / data=g3;
   model 1 / group = 1;
      mstruct;
      matrix _cov_  = cov01-cov45;
      matrix _mean_ = mean1-mean9;
   model 2 / group = 2;
      refmodel 1;
      renameparm mean2=mean2_new,      /* constraint a */
                 mean4=mean4_new,      /* constraint b */
                 mean6=mean6_mdl2;
   model 3 / group = 3;
      refmodel 1;
      renameparm mean2=mean2_new,      /* constraint a */
                 mean3=mean3_mdl3,
                 mean4=mean4_new;      /* constraint b */
   fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
   run;
```

This forms the fourth revised model. Output 26.21.16 shows the fit summary of this revised model. Again, the chi-square, AIC, CAIC, and SBC values all show improvements, as compared with the third revised model. However, the improvements do seem to slow down. For example, the CAIC value drops from 361.95 to the current value at 358.43—a mere 3 points reduction. The SBC value drops from 304.95 to the current value at 300.43—a mere 4 points reduction. These small reductions indicate that you might soon reach the point that no more model fit improvement would be possible with additional release of parameter constraints.

Output 26.21.16 Fit Summary for the Fourth Revised Model

Fit Summary	
Chi-Square	60.1265
Chi-Square DF	104
Pr > Chi-Square	0.9998
RMSEA Estimate	0.0000
Akaike Information Criterion	176.1265
Bozdogan CAIC	358.4283
Schwarz Bayesian Criterion	300.4283

Output 26.21.17 suggests that the mean of x1 in Model 1 (which has the largest LM statistic at 6.45785) could be released from the equality constraint to achieve the largest model improvement over the current model.

Output 26.21.17 LM Statistics for Releasing the Equality Constraints in the Fourth Revised Model

Lagrange Multiplier Statistics for Releasing Equality Constraints

Parm	---Released Parameter--				LM Stat	Pr > ChiSq	------Changes-----	
	Model	Type	Var1	Var2			Original Parm	Released Parm
cov01	1	COV	x1	x1	2.60531	0.1065	0.1375	-0.2751
	2	COV	x1	x1	0.28124	0.5959	0.0469	-0.0871
	3	COV	x1	x1	4.75010	0.0293	-0.1788	0.3859
					.			
					.			
					.			
mean1	1	MEAN	x1		6.45759	0.0110	0.2616	-0.5232
	2	MEAN	x1		5.00966	0.0252	-0.1921	0.3568
	3	MEAN	x1		0.05927	0.8076	0.0209	-0.0451
mean3	1	MEAN	x3		1.53288	0.2157	-0.1298	0.1406
	2	MEAN	x3		1.53288	0.2157	0.1406	-0.1298
mean5	1	MEAN	x5		0.09750	0.7549	-0.0457	0.0913
	2	MEAN	x5		0.19683	0.6573	-0.0716	0.1330
	3	MEAN	x5		0.56575	0.4520	0.1070	-0.2310
mean6	1	MEAN	x6		0.35797	0.5496	0.1141	-0.1113
	3	MEAN	x6		0.35797	0.5496	-0.1113	0.1141
mean7	1	MEAN	x7		4.55702E-6	0.9983	0.000351	-0.00070
	2	MEAN	x7		0.96359	0.3263	0.1572	-0.2920
	3	MEAN	x7		1.00887	0.3152	-0.1486	0.3208
mean8	1	MEAN	x8		0.20289	0.6524	-0.0676	0.1352
	2	MEAN	x8		0.12448	0.7242	0.0525	-0.0974
	3	MEAN	x8		0.00590	0.9388	0.0110	-0.0237
mean9	1	MEAN	x9		0.05894	0.8082	-0.0271	0.0542
	2	MEAN	x9		1.63722	0.2007	-0.1448	0.2689
	3	MEAN	x9		2.44244	0.1181	0.1652	-0.3565
mean2_new	2	MEAN	x2		3.05066	0.0807	0.2396	-0.2246
	3	MEAN	x2		3.05066	0.0807	-0.2246	0.2396
mean4_new	2	MEAN	x4		1.81990	0.1773	0.2306	-0.2003
	3	MEAN	x4		1.81990	0.1773	-0.2003	0.2306

To make the mean parameter for x1 in Model 1 unique, the mean parameters for x1 in Models 2 and 3 are renamed from mean1 to mean1_new, as shown in the following statements:

```
proc calis modification;
   var x1-x9;
   group 1 / data=g1;
   group 2 / data=g2;
   group 3 / data=g3;
   model 1 / group = 1;
      mstruct;
      matrix _cov_  = cov01-cov45;
      matrix _mean_ = mean1-mean9;
   model 2 / group = 2;
      refmodel 1;
      renameparm mean1=mean1_new,     /* constraint c */
                 mean2=mean2_new,     /* constraint a */
                 mean4=mean4_new,     /* constraint b */
                 mean6=mean6_mdl2;
   model 3 / group = 3;
      refmodel 1;
      renameparm mean1=mean1_new,    /* constraint c */
                 mean2=mean2_new,    /* constraint a */
                 mean3=mean3_mdl3,
                 mean4=mean4_new;    /* constraint b */
   fitindex NoIndexType On(only)=[chisq df probchi rmsea aic caic sbc];
   run;
```

This forms the fifth revised model. Output 26.21.18 shows the fit summary of the fifth revised model. Again, the chi-square, AIC, CAIC, and SBC values all show improvements, as compared with the fourth revised model. However, the improvements slow down even more. For example, the CAIC value drops from 358.43 to the current value at 356.32. The SBC value drops from 300.43 to the current value at 297.32. Because the model fit does not improve much, this is the point where you would cease to release more equality constraints for improving the model fit.

Output 26.21.18 Fit Summary for the Fifth Revised Model

Fit Summary

Chi-Square	52.8821
Chi-Square DF	103
Pr > Chi-Square	1.0000
RMSEA Estimate	0.0000
Akaike Information Criterion	170.8821
Bozdogan CAIC	356.3270
Schwarz Bayesian Criterion	297.3270

Output 26.21.19 does not suggest the release of any equality constraints on the means, because all the p-values for the LM statistics are not significant (that is, all are greater than 0.05). Therefore, the same suggestion from examining the model fit improvements of the fifth revised model echoes here: this is the point that the "best" model for the data is found.

Output 26.21.19 LM Statistics for Releasing the Equality Constraints in the Fifth Revised Model

Lagrange Multiplier Statistics for Releasing Equality Constraints

	---Released Parameter--						------Changes-----	
Parm	Model	Type	Var1	Var2	LM Stat	Pr > ChiSq	Original Parm	Released Parm
cov01	1	COV	x1	x1	4.06279	0.0438	0.1590	-0.3180
	2	COV	x1	x1	0.48735	0.4851	0.0571	-0.1061
	3	COV	x1	x1	7.60892	0.0058	-0.2095	0.4520
					.			
					.			
					.			
mean3	1	MEAN	x3		0.08362	0.7725	-0.0312	0.0382
	2	MEAN	x3		0.08362	0.7725	0.0382	-0.0312
mean5	1	MEAN	x5		0.02394	0.8771	0.0229	-0.0458
	2	MEAN	x5		0.47064	0.4927	-0.1113	0.2067
	3	MEAN	x5		0.26010	0.6101	0.0728	-0.1570
mean6	1	MEAN	x6		0.97515	0.3234	0.1893	-0.1893
	3	MEAN	x6		0.97515	0.3234	-0.1893	0.1893
mean7	1	MEAN	x7		0.03746	0.8465	-0.0319	0.0638
	2	MEAN	x7		1.10425	0.2933	0.1683	-0.3126
	3	MEAN	x7		0.79472	0.3727	-0.1321	0.2851
mean8	1	MEAN	x8		0.86794	0.3515	-0.1426	0.2852
	2	MEAN	x8		0.47498	0.4907	0.1038	-0.1928
	3	MEAN	x8		0.03721	0.8470	0.0276	-0.0595
mean9	1	MEAN	x9		0.12190	0.7270	0.0401	-0.0801
	2	MEAN	x9		2.66768	0.1024	-0.1869	0.3472
	3	MEAN	x9		1.78113	0.1820	0.1417	-0.3058
mean1_new	2	MEAN	x1		1.28032	0.2578	-0.1794	0.1359
	3	MEAN	x1		1.28032	0.2578	0.1359	-0.1794
mean2_new	2	MEAN	x2		2.53142	0.1116	0.2117	-0.2112
	3	MEAN	x2		2.53142	0.1116	-0.2112	0.2117
mean4_new	2	MEAN	x4		2.25834	0.1329	0.2558	-0.2253
	3	MEAN	x4		2.25834	0.1329	-0.2253	0.2558

To see where the fifth revised model (equality in the covariance matrix and partial equality in the means) stands between the models under H_0 (equality in the covariance and mean matrices) and H_1 (equality in the covariance matrix only), the following table shows the fit statistics of these three models:

	H_0	**“Fifth”**	H_1
Chi-square	203.2605	52.8821	26.7897
Chi-square DF	108	103	90
Pr > chi-square	<0.0001	1.0000	1.0000
RMSEA estimate	0.2100	0.0000	0.0000
Akaike information criterion	311.2605	170.8821	170.7897
Bozdogan CAIC	480.9898	356.3270	397.0954
Schwarz Bayesian criterion	426.9898	297.3270	325.0954

The fifth revised model is labeled “Fifth” in the table. Compared with the model under H_0, the fifth revised model is clearly superior. It uses only five more parameters (or five fewer degrees of freedom), but the improvement in the model fit chi-square and the RMSEA value are huge. The AIC, CAIC, and SBC are also much better.

Compared with the model under H_1, the fifth revised model appears to be inferior in only the chi-square model fit statistic, although both models already have the highest possible p-value at 1.000 and smallest possible RMSEA value at 0. However, the model under H_1 uses 13 more parameters (or it has 13 fewer degrees of freedom), and hence it is more complex. In fact, because the model fit chi-square value does not take model complexity into account, it is often criticized as the basis for choosing competing models for the data. In contrast, the AIC, CAIC, and SBC measures take model complexity into account, and they are more reasonable as the basis for choosing competing models. Although the AIC values for the fifth revised model and the model under H_1 are very close, the CAIC and SBC values clearly favor the fifth revised model. Therefore, according to the CAIC and SBC criteria, the fifth revised model, which is a model with partial equality constraints on the means, is actually better than the model with all the means being unconstrained (that is, under H_1) for the current data with three independent groups.

Example 26.22: Illustrating Various General Modeling Languages

In PROC CALIS, you can use many different modeling languages to specify the same model. The choice of modeling language depends on personal preferences and the purposes of the analysis. See the section "Which Modeling Language?" on page 1012 for guidance. In this example, the data and the model in Example 26.16 are used to illustrate how a particular model can be specified by various general modeling languages.

RAM Model Specification

In Example 26.16, you use the PATH modeling language to specify the model because of its close resemblance to the path diagram. In this example, you consider another modeling language of PROC CALIS that is also closely related to the path diagram representation of structural equation models. The so-called RAM model language has syntax that represents the single- and double-headed paths (or arrows) in the path diagram. However, unlike the PATH modeling language, the RAM modeling language is matrix-based. The following statements show how you can specify the same path model with the RAM model specification for the data in Example 26.16:

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

In the RAM model for covariance structure analysis, you have two important matrices to specify. The first one is the _A_ matrix, which is for the specification of the single-headed paths (arrows) in the path diagram. The second one is the _P_ matrix, which is for the specification of the double-headed paths (arrows) in the path diagram. Hence, to specify the RAM model is much like mapping the path diagram arrows into the parameter of the RAM model matrices.

In the RAM statement, you can specify the variables in the model in the VAR= option. The VAR= list contains all observed and latent variables in your path diagram (without the use of error terms). Although you can specify the variables in the VAR= list in any order you like, the variable order in the list is also the order of variables in the RAM model matrices. In VAR= list of the RAM statement, you put comments to note the order of the variables.

After you specify the variable list, you can specify the model parameter locations in the RAM statement entries. In the first nine entries, you specify the single-headed paths by mapping them into the elements of the _A_ matrix of the RAM model. For example, the first entry represents the single-headed path of variable 1 (Anomie67) from variable 7 (Alien67). The corresponding path effect or coefficient is fixed at 1, which is also the value for _A_[1,7]. Another example is the ninth path entry. You specify a single-headed path of variable 8 (Alien71) from variable 7 (Alien67). The corresponding path effect or coefficient is a free parameter named beta, which is also the parameter for _A_[8,7]. Hence, you can specify all single-headed paths in the path diagram as elements in the _A_ matrix of the RAM model.

To facilitate the comparisons between the RAN and PATH modeling languages, the PATH model specification in Example 26.16 for the same data is reproduced in the following:

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

It is clear that each of the path entries specified in the PATH statement corresponds to an matrix element entry of the _A_ matrix in the RAM statement. How about the specifications of the double-headed arrows in the path diagram? Do the RAM and PATH model specifications correspond to each other?

The answer is yes. In the PATH modeling language, you specify all double-headed arrows in the path diagram as entries either in the PVAR or PCOV statement. In the RAM modeling language, you specify the corresponding entries as matrix element entries of the _P_ matrix in the RAM statement. For example, the error variance of Anomie67 is a parameter called _Variabletheta1 in the PVAR statement of the PATH model. You specify the same parameter for the _P_[1,1] element in an entry of the RAM statement. Another example is the error covariance between Powerless67 and Powerless71. You specify this a parameter called theta5 in the last entry of the PCOV statement in the PATH model. You specify the same parameter for the _P_[2,4] element in the last entry of the RAM statement. Therefore, it is not difficult to find that the specifications in the PATH and the RAM model have some kind of one-to-one correspondence.

Output 26.22.1 shows the RAM model estimates for the Wheaton data. These RAM model estimates match the set of estimates using the PATH model specification, as shown in Output 26.16.10.

Output 26.22.1 RAM Model Estimates

RAM Pattern and Estimates

Matrix	-----Row-----		---Column----		Parameter	Estimate	Standard Error	t Value
_A_ (1)	Anomie67	1	Alien67	7		1.00000		
	Powerless67	2	Alien67	7		0.83300		
	Anomie71	3	Alien71	8		1.00000		
	Powerless71	4	Alien71	8		0.83300		
	Education	5	SES	9		1.00000		
	SEI	6	SES	9	lambda	5.36883	0.43371	12.37880
	Alien67	7	SES	9	gamma1	-0.62994	0.05634	-11.18092
	Alien71	8	SES	9	gamma2	-0.24086	0.05489	-4.38836
	Alien71	8	Alien67	7	beta	0.59312	0.04678	12.67884
_P_ (2)	Anomie67	1	Anomie67	1	theta1	3.60796	0.20092	17.95717
	Powerless67	2	Powerless67	2	theta2	3.59488	0.16448	21.85563
	Anomie71	3	Anomie71	3	theta1	3.60796	0.20092	17.95717
	Powerless71	4	Powerless71	4	theta2	3.59488	0.16448	21.85563
	Education	5	Education	5	theta3	2.99366	0.49861	6.00398
	SEI	6	SEI	6	theta4	259.57639	18.31151	14.17559
	Alien67	7	Alien67	7	psi1	5.67046	0.42301	13.40500
	Alien71	8	Alien71	8	psi2	4.51479	0.33532	13.46394
	SES	9	SES	9	phi	6.61634	0.63914	10.35190
	Anomie67	1	Anomie71	3	theta5	0.90580	0.12167	7.44472
	Powerless67	2	Powerless71	4	theta5	0.90580	0.12167	7.44472

LINEQS Model Specification

Another way to specify the model in Example 26.16 is to use the LINEQS modeling language, which is shown in the following:

```
proc calis nobs=932 data=Wheaton;
   lineqs
      Anomie67     = 1.0    * f_Alien67 + e1,
      Powerless67  = 0.833  * f_Alien67 + e2,
      Anomie71     = 1.0    * f_Alien71 + e3,
      Powerless71  = 0.833  * f_Alien71 + e4,
      Education    = 1.0    * f_SES     + e5,
      SEI          = lambda * f_SES     + e6,
      f_Alien67    = gamma1 * f_SES     + d1,
      f_Alien71    = gamma2 * f_SES     + beta * f_Alien67 + d2;
   variance
      E1           = theta1,
      E2           = theta2,
      E3           = theta1,
      E4           = theta2,
      E5           = theta3,
      E6           = theta4,
      D1           = psi1,
      D2           = psi2,
      f_SES        = phi;
   cov
      E1  E3       = theta5,
      E2  E4       = theta5;
run;
```

As compared with the PATH and RAM modeling languages, the most distinct feature of the LINEQS modeling language is the explicit use of error terms in equation specifications. In the LINEQS statement, you specify exactly one equation for each endogenous variable. In each equation, you list an endogenous variable on the left-hand-side of the equation and all its predictors on the right-hand-side of the equation. You must also include an error term in each equation. Because each endogenous variable in the LINEQS statement can only be specified in exactly one equation, the number of equations in the LINEQS model and the number of paths in the corresponding path diagram do not match necessarily. In this example, there are eight equations in the LINEQS statement, but there are nine paths in the corresponding path diagram.

In addition, in the LINEQS model, you need to follow a convention of naming latent variables. For latent variables that are neither errors nor disturbances, you must use either the 'F' or 'f' prefix. For error terms, you must use either the 'E' or 'e' prefix. For disturbances, you must use either the 'D' or 'd' prefix. However, in the PATH or RAM model specification, no such convention is imposed. For example, f_Alien67, f_Alien71, and f_SES are latent factors in the LINEQS model. They are not error terms, and so they must start with the 'f' prefix. However, this prefix is not needed in the PATH or RAM model. Furthermore, there are no explicit error terms that need to be specified in the PATH or RAM model, let alone specific prefixes for the error terms.

The PVAR statement in the PATH model is replaced with the VARIANCE statement in the LINEQS model, and the PCOV statement with the COV statement. The PVAR and PCOV statements in the PATH model are for the partial variance and partial covariance specifications. The partial variance or covariance concepts are used in the PATH or RAM model specification because error terms are not named explicitly. Specification of error variances in the PATH and RAM model is conceptualized as the specification of the partial variances of the corresponding variables. But in the LINEQS model, because errors or disturbances are named explicitly as *exogenous* variables, the partial variance or covariance concepts are no longer necessary. Instead, you specify the variances of the error terms directly, which reflects the conceptualization behind the VARIANCE statement of the LINEQS modeling language. Similarly, you use the COV, but not PCOV, statement in the LINEQS modeling language because you can specify the covariances among variables or error terms without using the partial covariance conceptualization.

In this example, the variances of the errors ("E"-variables) and disturbances ("D"-variables) specified in the VARIANCE statement of the LINEQS model correspond to the partial variances of the endogenous variables specified in the PVAR statement of the PATH model. Similarly, covariances of errors specified in the COV statement of the LINEQS model correspond to the partial covariances of endogenous variables specified in the PCOV statement of the PATH model. The estimation results of the LINEQS model are shown in Output 26.22.2. Again, they are essentially the same estimates obtained from the PATH model specified in Example 26.16, as shown in Output 26.16.10.

Output 26.22.2 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 Type	Variable	Parameter	Estimate	Standard 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

Var1	Var2	Parameter	Estimate	Standard 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 1212 for details.

To use the LISMOD modeling language, you must recognize four types of variables in the 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 $\mathbf{y}$-variables are manifest variables that are indicators of the η-variables, and the $\mathbf{x}$-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 $\mathbf{y}$-variables. Manifest indicators for SES include Education and SEI, which are the $\mathbf{x}$-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 26.16:

```
proc calis nobs=932 data=Wheaton;
   lismod
      yvar   = Anomie67 Powerless67 Anomie71 Powerless71,
      xvar   = Education SEI,
      etavar = Alien67  Alien71,
      xivar  = SES;
   matrix _LAMBDAY_
      [1,1]  = 1,
      [2,1]  = 0.833,
      [3,2]  = 1,
      [4,2]  = 0.833;
   matrix _LAMBDAX_
      [1,1]  = 1,
      [2,1]  = lambda;
   matrix _GAMMA_
      [1,1]  = gamma1,
      [2,1]  = gamma2;
   matrix _BETA_
      [2,1]  = beta;
   matrix _THETAY_
      [1,1]  = theta1-theta2 theta1-theta2,
      [3,1]  = theta5,
      [4,2]  = theta5;
   matrix _THETAX_
      [1,1]  = theta3-theta4;
   matrix _PSI_
      [1,1]  = psi1-psi2;
   matrix _PHI_
      [1,1]  = phi;
run;
```

In the LISMOD statement, you specify the four lists of variables in the model. The orders of the variables in these lists define the order of the row and column variables in the model matrices, of which the parameter locations are specified in the MATRIX statements.

The estimated model is divided into three conceptual parts. The first part is the measurement model that relates the η-variables with the **y**-variables, as shown in Output 26.22.3:

Output 26.22.3 LISMOD Model Measurement Model for the η-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 0.2009 17.9572 [theta1]	0	0.9058 0.1217 7.4447 [theta5]	0
Powerless67	0	3.5949 0.1645 21.8556 [theta2]	0	0.9058 0.1217 7.4447 [theta5]
Anomie71	0.9058 0.1217 7.4447 [theta5]	0	3.6080 0.2009 17.9572 [theta1]	0
Powerless71	0	0.9058 0.1217 7.4447 [theta5]	0	3.5949 0.1645 21.8556 [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 26.22.4:

Output 26.22.4 LISMOD Model Measurement Model for the ξ-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 26.22.5:

Output 26.22.5 LISMOD Structural Model for the Latent Variables

BETA Matrix: Estimate/StdErr/t-value

	Alien67	Alien71
Alien67	0	0
Alien71	0.5931 0.0468 12.6788 [beta]	0

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 0.4230 13.4050 [psi1]	0
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 the current example, there is only one such effect. The effect of Alien67 on Alien71 is represented by the free parameter beta. The _GAMMA_ matrix contains effects of the ξ-variable, which is SES in this example, on the η-variables Alien67 on Alien71. These effects are represented by the free parameters gamma1 and gamma2. The _PSI_ matrix contains the error variances and covariances in the structural model. In this example, psi1 and psi2 are two free parameters for the error variances. Finally, the _PHI_ matrix is the covariance matrix for the 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 26.23: Testing Competing Path Models for the Career Aspiration Data

This example uses some well-known data from Haller and Butterworth (1960). The section “A Combined Measurement-Structural Model” on page 330 analyzes some models for these data. Inspired by the examples given in Loehlin (1987), this example shows additional applications to the same data set, but with a focus on testing nested models. By manipulating the OUTMODEL= data set, this example shows how you can specify new models in an efficient way. Various models and analyses of these data are also given by Duncan, Haller, and Portes (1968), Jöreskog and Sörbom (1988), and Loehlin (1987).

The study is concerned with the career aspirations of high school students and how these aspirations are affected by close friends. The data are collected from 442 seventeen-year-old boys in Michigan. There are 329 boys in the sample who named another boy in the sample as a best friend. The data from these 329 boys paired with the data from their best friends are analyzed.

Because of the dependency of the data, the effective sample size assumed in the example is 329, which you can specify in the NOBS= option in the PROC CALIS statements. See the section “A Combined Measurement-Structural Model” on page 330 for the justification of the use of this effective sample size.

The correlation matrix, taken from Jöreskog and Sörbom (1988), is shown in the following DATA step:

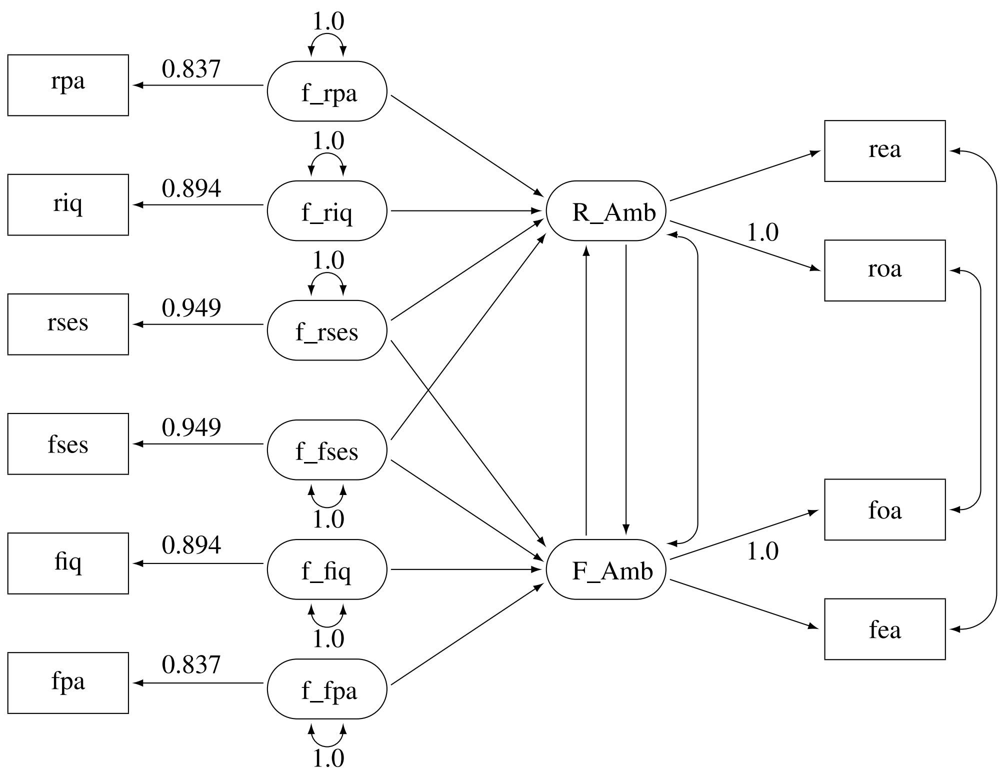
```
title 'Peer Influences on Aspiration: Haller & Butterworth (1960)';
data aspire(type=corr);
   _type_='corr';
   input _name_ $ riq rpa rses roa rea fiq fpa fses foa fea;
   label riq='Respondent: Intelligence'
         rpa='Respondent: Parental Aspiration'
         rses='Respondent: Family SES'
         roa='Respondent: Occupational Aspiration'
         rea='Respondent: Educational Aspiration'
         fiq='Friend: Intelligence'
         fpa='Friend: Parental Aspiration'
         fses='Friend: Family SES'
         foa='Friend: Occupational Aspiration'
         fea='Friend: Educational Aspiration';
   datalines;
riq   1.      .      .      .      .      .       .      .      .      .
rpa   .1839  1.      .      .      .      .       .      .      .      .
rses  .2220  .0489  1.      .      .      .       .      .      .      .
roa   .4105  .2137  .3240  1.      .      .       .      .      .      .
rea   .4043  .2742  .4047  .6247  1.      .       .      .      .      .
fiq   .3355  .0782  .2302  .2995  .2863  1.       .      .      .      .
fpa   .1021  .1147  .0931  .0760  .0702  .2087   1.      .      .      .
fses  .1861  .0186  .2707  .2930  .2407  .2950  -.0438  1.      .      .
foa   .2598  .0839  .2786  .4216  .3275  .5007   .1988  .3607  1.      .
fea   .2903  .1124  .3054  .3269  .3669  .5191   .2784  .4105  .6404  1.
;
```

For illustration purposes, this correlation matrix is treated here as if it were a covariance matrix for PROC CALIS to analyze. The reason is that the chi-square tests shown in this example are valid only with covariance structure analysis. See Example 26.26 for an illustration of covariance structure analysis on correlations.

Model 1: The Full Model

Loehlin (1987) analyzes the following path model for the data:

Figure 26.4 Path Diagram for Career Aspiration: Model 1



In Figure 26.4, the observed variables rpa, riq, rses, fses, fiq, and fpa are measured with errors. Their true scores counterparts f_rpa, f_riq, f_rses, f_fses, f_fiq, and f_fpa are latent variables in the model. Path coefficients from these latent variables to the observed variables are fixed coefficients, indicating the square roots of the theoretical reliabilities in the model. These latent variables, rather than the observed counterparts, serve as predictors of the ambition factors R_Amb and F_Amb. The error terms for these two latent factors are correlated, as indicated by a double-headed path (arrow) that connects the two factors. Correlated errors for the occupational aspiration variables (roa and foa) and the educational aspiration variables (rea and fea) are also shown in Figure 26.4. These correlated errors are also represented by two double-headed paths (arrows) in the path diagram.

Notice that the covariances among the six exogenous latent variables (f_rpa, f_riq, f_rses, f_fses, f_fiq, and f_fpa) are not represented in the path diagram for two reasons. First, there are 15 of these covariances and hence you need 15 double-headed arrows to represent them in the path diagram. Apparently, because of the space limitations, it would be difficult to put all these double-headed arrows in the path diagram without cluttering it. Second, covariances among exogenous latent variables are free parameters by default in PROC CALIS, and therefore omitting these double-headed arrows in the path diagram is compatible with the default model specification in PROC CALIS. Similarly, double-headed arrows for the error variances of the endogenous variables (rpa, riq, rses, fses, fiq, fpa, R_Amb, and F_Amb) in the path diagram are omitted because they are unconstrained free parameters and are set automatically by default in PROC CALIS .

The model represented by the path digram in Figure 26.4 is considered to be the full model for the data, in the sense that it has the largest number of parameters among the competing models considered this example. The same model is analyzed in the section "A Combined Measurement-Structural Model" on page 330 with the following specification:

```
proc calis data=aspire nobs=329;
   path
      /* measurement model for intelligence and environment */
      rpa     <---  f_rpa    = 0.837,
      riq     <---  f_riq    = 0.894,
      rses    <---  f_rses   = 0.949,
      fses    <---  f_fses   = 0.949,
      fiq     <---  f_fiq    = 0.894,
      fpa     <---  f_fpa    = 0.837,

      /* structural model of influences: 5 equality constraints */
      f_rpa   --->  R_Amb ,
      f_riq   --->  R_Amb ,
      f_rses  --->  R_Amb ,
      f_fses  --->  R_Amb ,
      f_rses  --->  F_Amb ,
      f_fses  --->  F_Amb ,
      f_fiq   --->  F_Amb ,
      f_fpa   --->  F_Amb ,
      F_Amb   --->  R_Amb ,
      R_Amb   --->  F_Amb ,

      /* measurement model for aspiration: 1 equality constraint */
      R_Amb   --->  rea  ,
      R_Amb   --->  roa      = 1.,
      F_Amb   --->  foa      = 1.,
      F_Amb   --->  fea  ;
   pvar
      f_rpa f_riq f_rses f_fpa f_fiq f_fses = 6 * 1.0;
   pcov
      R_Amb F_Amb             ,
      rea  fea                ,
      roa  foa                ;
run;
```

The PATH model specification represents each arrow (single-headed and double-headed) in the path diagram. You transcribe each arrow in Figure 26.4 into an entry in the PATH model. The PATH statement specifies all the single-headed arrows in the path diagram. The PVAR statement specifies all the double-headed arrows that point to individual variables (that is, the fixed error variances of the exogenous latent variables) in the path diagram. The PCOV statement specifies all the double-headed arrows that connect paired variables (that is, the error covariances) in the path diagram.

Output 26.23.1 shows the fit summary of Model 1.

Output 26.23.1 Career Aspiration Data: Fit Summary of Model 1

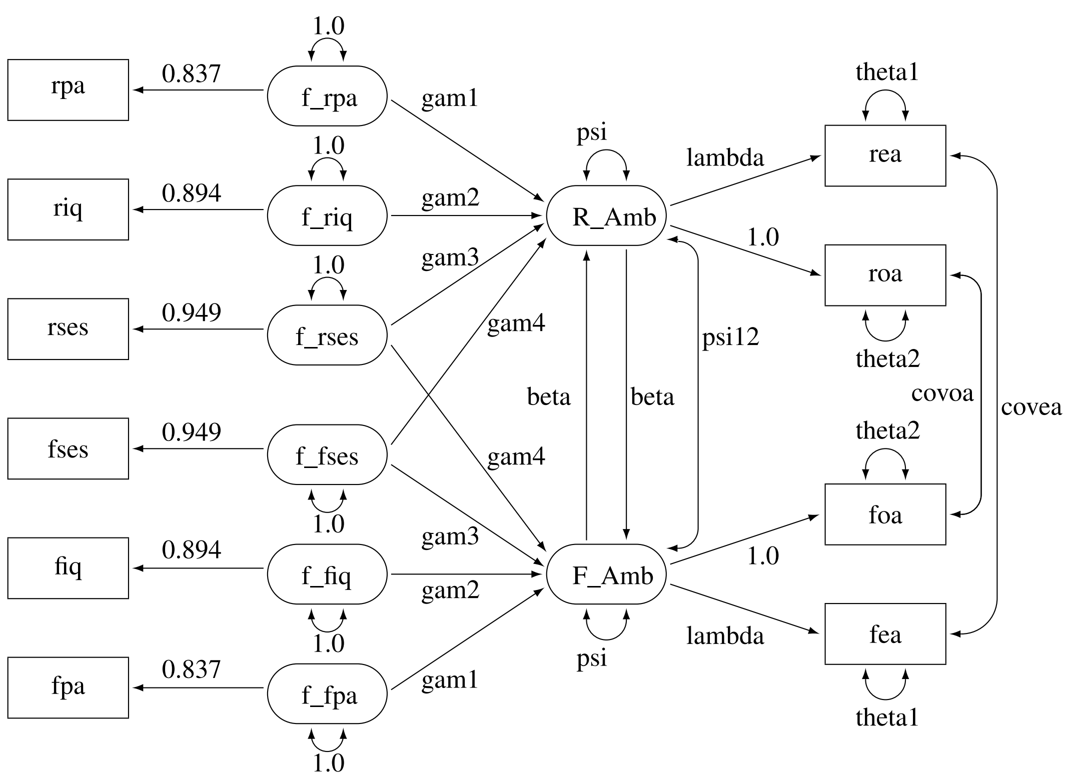
Fit Summary	
Chi-Square	12.0132
Chi-Square DF	13
Pr > Chi-Square	0.5266
Standardized RMSR (SRMSR)	0.0149
RMSEA Estimate	0.0000
Akaike Information Criterion	96.0132
Bozdogan CAIC	297.4476
Schwarz Bayesian Criterion	255.4476

Since the p-value for the chi-square test is 0.5266, this model clearly cannot be rejected. Both standardized RMSR and RMSEA are very small. All these point to an excellent model fit. Three information-theoretic fit indices are also shown: Akaike's information criterion (AIC), Bozdogan's CAIC, and Schwarz's Bayesian Criterion (SBC). These indices are useful when you need to compare competing models for the data.

Model 2: The Model with Equality Constraints

You now consider a much more restrictive model with equality constraints in the model. The path diagram for this constrained model is shown in Figure 26.5.

Figure 26.5 Path Diagram for Career Aspiration: Model 2



The main idea about setting the equality constraints in this model is that there is some symmetry in the model components that correspond to the respondent and his friend. In particular, the corresponding coefficients or parameters should be equal. For example, the path f_rpa--->R_Amb for the respondent has the same effect as that of f_fpa--->F_Amb. In the path diagram, they are both labeled by the same parameter gam1. Generalizing the same idea to other pairs of paths, Output 26.5 shows nine pairs of these equality constraints, which are all represented by the same parameter names for distinct (single-headed or double-headed) paths.

However, because of the space limitation, there are six more equality constraints that are not shown in the path diagram. These six constraints concern the covariance structures of the exogenous latent factors f_rpa, f_riq, f_rses, f_fses, f_fiq, and f_fpa. The first three factors are for the respondent, and the last three are for his friend. Using the same symmetry argument, the covariance structures imposed on these exogenous latent factors are shown in the following:

```
         f_rpa  f_riq  f_rses   f_fpa f_fiq f_fses
f_rpa      1.
f_riq      c1     1.
f_rses     c2     c3     1.

f_fpa      c4     c5     c6       1.
f_fiq      c5     c7     c8       c1    1.
f_fses     c6     c8     c9       c2    c3    1.
```

In this pattern of covariance structures, the covariance matrix (upper left portion) for the latent factors of the respondent is the same as that (lower right portion) for the latent factors of his friend. The cross-covariances among the factors between the friends (lower left portion) also display a symmetry pattern. There are six pairs of equality constraints in the covariance structures. Imposing these six pairs of equality constraints and the nine pairs of equality constraints in the path diagram lead to Loehlin's (1987) Model 2.

You can specify the current constrained model by the following PATH modeling language of PROC CALIS:

```
proc calis data=aspire nobs=329 outmodel=model2;
   path
      /* measurement model for intelligence and environment */
      rpa     <---  f_rpa    = 0.837,
      riq     <---  f_riq    = 0.894,
      rses    <---  f_rses   = 0.949,
      fses    <---  f_fses   = 0.949,
      fiq     <---  f_fiq    = 0.894,
      fpa     <---  f_fpa    = 0.837,

      /* structural model of influences: 5 equality constraints */
      f_rpa   --->  R_Amb    = gam1,
      f_riq   --->  R_Amb    = gam2,
      f_rses  --->  R_Amb    = gam3,
      f_fses  --->  R_Amb    = gam4,
      f_rses  --->  F_Amb    = gam4,
      f_fses  --->  F_Amb    = gam3,
      f_fiq   --->  F_Amb    = gam2,
      f_fpa   --->  F_Amb    = gam1,
      F_Amb   --->  R_Amb    = beta,
      R_Amb   --->  F_Amb    = beta,

      /* measurement model for aspiration: 1 equality constraint */
      R_Amb   --->  rea      = lambda,
      R_Amb   --->  roa      = 1.,
      F_Amb   --->  foa      = 1.,
      F_Amb   --->  fea      = lambda;
   pvar
      f_rpa f_riq f_rses f_fpa f_fiq f_fses = 6 * 1.0,
      R_Amb F_Amb             = 2 * psi,        /* 1 ec */
      rea fea                 = 2 * theta1,     /* 1 ec */
      roa foa                 = 2 * theta2;     /* 1 ec */
   pcov
      R_Amb F_Amb             = psi12,
      rea  fea                = covea,
      roa  foa                = covoa,
      f_rpa f_riq f_rses      = cov1-cov3,       /* 3 ec */
      f_fpa f_fiq f_fses      = cov1-cov3,
      f_rpa f_riq f_rses * f_fpa f_fiq f_fses =  /* 3 ec */
          cov4 cov5 cov6  cov5 cov7 cov8  cov6 cov8 cov9;
run;
```

In the current PATH model specification, you specify the same set of paths as in Model 1. In addition, to set the required constraints in this path model, you use parameter names to label the related paths, variances, or covariances. Same parameter names mean equality constraints. The 15 equality constraints are labeled with comments in the specification. In the PROC CALIS statement, you use the OUTMODEL= option to output the model estimation results into the output data set model2, which is used for subsequent hypotheses tests.

Output 26.23.2 shows the fit summary of Model 2.

Output 26.23.2 Career Aspiration Data: Fit Summary of Loehlin (1987) Model 2

Fit Summary

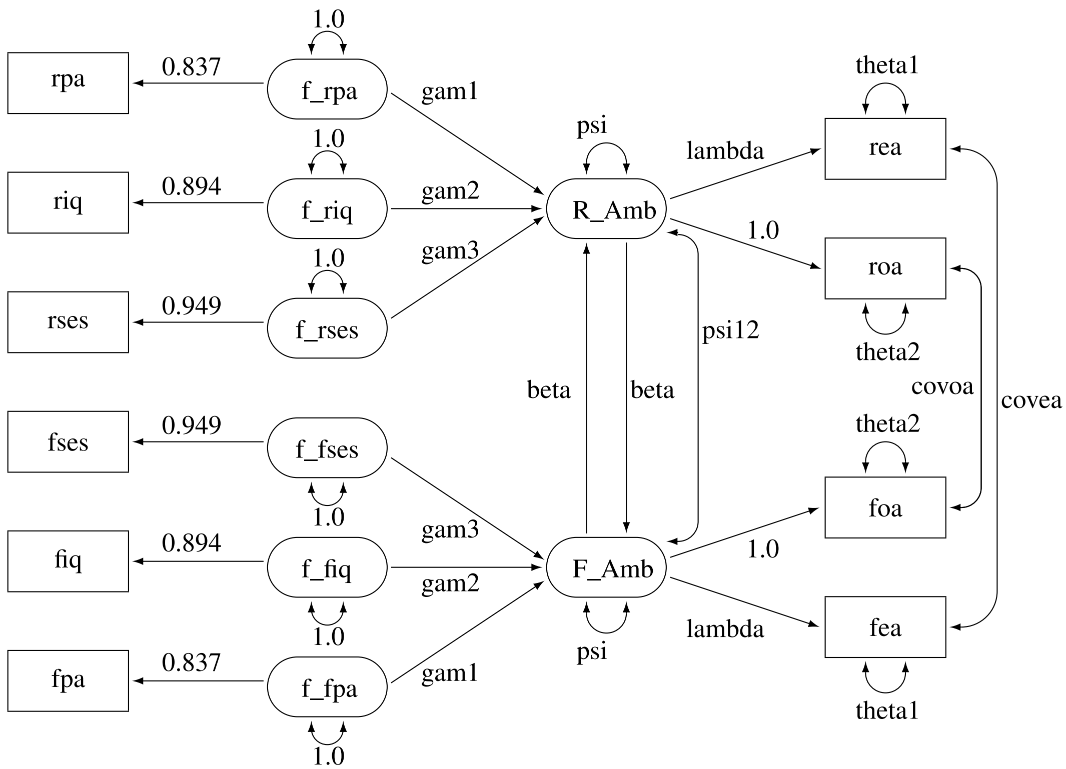
Chi-Square	19.0697
Chi-Square DF	28
Pr > Chi-Square	0.8960
Standardized RMSR (SRMSR)	0.0276
RMSEA Estimate	0.0000
Akaike Information Criterion	73.0697
Bozdogan CAIC	202.5632
Schwarz Bayesian Criterion	175.5632

The test of Loehlin's (1987) Model 2 against Model 1 yields a chi-square of $19.0697 - 12.0132 = 7.0565$ with 15 degrees of freedom, which is clearly not significant. This indicates that the restricted Model 2 fits at least as well as Model 1. Schwarz's Bayesian criterion (SBC) is also much lower for Model 2 (175.5623) than for Model 1 (255.4476). Hence, Model 2 seems preferable on both substantive and statistical grounds.

Model 3: No SES Paths

A question of substantive interest is whether the friend's socioeconomic status (SES) has a significant direct influence on a boy's ambition. This can be addressed by omitting the paths from f_fses to R_Amb and from f_rses to F_Amb designated by the parameter name gam4, yielding Loehlin's (1987) Model 3. The corresponding path diagram is shown in Figure 26.6.

Figure 26.6 Path Diagram for Career Aspiration: Model 3



In Figure 26.6, you drop the paths f_rses--->F_Amb and f_fses--->R_Amb from the previous model. Using the path diagram in Figure 26.6, you can specify the current model the same way you do for Model 2. However, because you have the estimation results from Model 2 in the SAS data set model2, you can modify this SAS data set to reflect the current model specification and then input the modified SAS data set as an INMODEL= file for PROC CALIS to analyze.

First, you create a new SAS data set model3 by the following DATA step:

```
data model3(type=calismdl);
   set model2;
   if _name_='gam4' then
      do;
         _name_=' ';
         _estim_=0;
      end;
run;
```

Essentially, by blanking out the parameter name for the target paths, you are stating that these paths are no longer associated with the free parameter gam4 in the new model. Instead, you put a fixed zero to these paths. This way you eliminate the paths f_rses--->F_Amb and f_fses--->R_Amb for Model 3, of which the model specification is now saved in the model3 data set.

Next, you input model3 as the INMODEL= data set for PROC CALIS to analyze, as shown in the following statements:

```
proc calis data=aspire nobs=329 inmodel=model3;
run;
```

PROC CALIS can now use the previous estimation results for fitting the required model. Output 26.23.3 shows the fit summary of Model 3.

Output 26.23.3 Career Aspiration Data: Fit Summary of Loehlin (1987) Model 3

Fit Summary	
Chi-Square	23.0365
Chi-Square DF	29
Pr > Chi-Square	0.7749
Standardized RMSR (SRMSR)	0.0304
RMSEA Estimate	0.0000
Akaike Information Criterion	75.0365
Bozdogan CAIC	199.7340
Schwarz Bayesian Criterion	173.7340

The chi-square value for testing Model 3 versus Model 2 is $23.0365 - 19.0697 = 3.9668$ with one degree of freedom and a p-value of 0.0464. The chi-square test shows a marginal significance, which means that the paths might be needed in the model. However, the SBC (173.7340) indicates that Model 3 is slightly preferable to Model 2, which has an SBC value of 175.5632.

Model 4: No Reciprocal Influence between the Ambition Factors

Another important question is whether the reciprocal influences between the respondent's and friend's ambitions are needed in the model. To test whether these paths are zero, you can set the parameter beta for the paths linking R_Amb and F_Amb to zero to obtain Loehlin's (1987) Model 4.

Similar to Model 3, you can modify the model2 data set to form the new model data set model4 for PROC CALIS to analyze, as shown in the following statements:

```
data model4(type=calismdl);
   set model2;
   if _name_='beta' then
      do;
         _name_=' ';
         _estim_=0;
      end;
run;

proc calis data=aspire nobs=329 inmodel=model4;
run;
```

Output 26.23.4 shows the fit summary of Model 4.

Output 26.23.4 Career Aspiration Data: Fit Summary of Loehlin (1987) Model 4

Fit Summary

Chi-Square	20.9981
Chi-Square DF	29
Pr > Chi-Square	0.8592
Standardized RMSR (SRMSR)	0.0304
RMSEA Estimate	0.0000
Akaike Information Criterion	72.9981
Bozdogan CAIC	197.6956
Schwarz Bayesian Criterion	171.6956

The chi-square value for testing Model 4 versus Model 2 is $20.9981 - 19.0697 = 1.9284$ with one degree of freedom and a p-value of 0.1649. Hence, there is little evidence of reciprocal influence.

Model 5: No Disturbance Correlation between the Ambition Factors

Loehlin's (1987) Model 2 has the direct paths connecting the latent ambition factors R_Amb and F_Amb and a covariance between the disturbance or error terms (that is, a double-headed arrow connecting the two factors in the path diagram shown in Figure 26.5). The presence of this disturbance correlation serves as a "wastebasket" that enables other omitted variables to have joint influences on the respondent's and his friend's ambition factors. To test the hypothesis that this disturbance correlation is zero, you use the following statements to set the parameter psi12 to zero in the model5 data set and fit the new model by PROC CALIS:

```
data model5(type=calismdl);
   set model2;
   if _name_='psi12' then
      do;
         _name_=' ';
         _estim_=0;
      end;
run;

proc calis data=aspire nobs=329 inmodel=model5;
run;
```

Output 26.23.5 displays the fit summary of Model 5.

Output 26.23.5 Career Aspiration Data: Fit Summary of Loehlin (1987) Model 5

Fit Summary

Chi-Square	19.0745
Chi-Square DF	29
Pr > Chi-Square	0.9194
Standardized RMSR (SRMSR)	0.0276
RMSEA Estimate	0.0000
Akaike Information Criterion	71.0745
Bozdogan CAIC	195.7721
Schwarz Bayesian Criterion	169.7721

The chi-square value for testing Model 5 versus Model 2 is $19.0745 - 19.0697 = 0.0048$ with one degree of freedom. This test statistic is insignificant. Therefore, omitting the covariance between the disturbance terms causes hardly any deterioration in the fit of the model.

Model 7: No Reciprocal Influence and No Disturbance Correlation between the Ambition Factors

The test in Model 4 fails to provide evidence of a direct reciprocal influence between the respondent's and friend's ambitions, and the test in Model 5 fails to provide evidence of a covariance or correlation between the disturbance terms for the ambition factors. Because you consider these two tests separately, you cannot establish evidence to eliminate the reciprocal influence and the disturbance correlation jointly. Instead, to make such a joint inference, it is important to test both hypotheses together by setting both beta and psi12 to zero as in Loehlin's (1987) Model 7. The following statements show how you can do that by modifying the model2 data set to form a new INMODEL= data set model7 for PROC CALIS to analyze:

```
data model7(type=calismdl);
   set model2;
   if _name_='psi12'|_name_='beta' then
      do;
         _name_=' ';
         _estim_=0;
      end;
run;

proc calis data=aspire nobs=329 inmodel=model7;
run;
```

Output 26.23.6 shows the fit summary of Model 7.

Output 26.23.6 Career Aspiration Data: Fit Summary of Loehlin (1987) Model 7

Fit Summary	
Chi-Square	25.3466
Chi-Square DF	30
Pr > Chi-Square	0.7080
Standardized RMSR (SRMSR)	0.0363
RMSEA Estimate	0.0000
Akaike Information Criterion	75.3466
Bozdogan CAIC	195.2480
Schwarz Bayesian Criterion	170.2480

When Model 7 is tested against Models 2, 4, and 5, the p-values are respectively 0.0433, 0.0370, and 0.0123, indicating that the combined effect of the reciprocal influence and the covariance of the disturbance terms is statistically significant. Thus, the hypothesis tests indicate that it is acceptable to omit either the reciprocal influences or the covariance of the disturbances, but not both.

Model 6: No Error Correlations between the Friend's Educational and Occupational Aspiration

It is also of interest to test the covariances (covea and covoa) between the error terms for educational aspiration (that is, between rea and fea) and occupational aspiration (that is, between roa and foa), because these terms are omitted from Jöreskog and Sörbom's (1988) models. Constraining covea and covoa to zero produces Loehlin's (1987) Model 6. You can use the following statements to fit this model:

```
data model6(type=calismdl);
   set model2;
   if _name_='covea'|_name_='covoa' then
      do;
         _name_=' ';
         _estim_=0;
      end;
run;

proc calis data=aspire nobs=329 inmodel=model6;
run;
```

Output 26.23.7 shows the fit summary of Model 6.

Output 26.23.7 Career Aspiration Data: Loehlin (1987) Model 6

Fit Summary

Chi-Square	33.4475
Chi-Square DF	30
Pr > Chi-Square	0.3035
Standardized RMSR (SRMSR)	0.0306
RMSEA Estimate	0.0187
Akaike Information Criterion	83.4475
Bozdogan CAIC	203.3489
Schwarz Bayesian Criterion	178.3489

The chi-square value for testing Model 6 versus Model 2 is $33.4476 - 19.0697 = 14.3779$ with two degrees of freedom and a p-value of 0.0008, indicating that there is considerable evidence of correlation between the error terms.

Summary of Competing Models

The following table summarizes the results from Loehlin's (1987) seven models.

Model	χ^2	df	p-value	SBC
1. Full model	12.0132	13	0.5266	255.4476
2. Equality constraints	19.0697	28	0.8960	175.5632
3. No SES path	23.0365	29	0.7749	173.7340
4. No reciprocal influence	20.9981	29	0.8592	171.6956
5. No disturbance correlation	19.0745	29	0.9194	169.7721
6. No error correlation	33.4475	30	0.3035	178.3489
7. Constraints from both 4 and 5	25.3466	30	0.7080	170.2480

For comparing models, you can use a DATA step to compute the differences of the chi-square statistics and p-values, as shown in the following statements:

```
data _null_;
   array achisq[7] _temporary_
      (12.0132 19.0697 23.0365 20.9981 19.0745 33.4475 25.3466);
   array adf[7] _temporary_
      (13 28 29 29 29 30 30);
   retain indent 16;
   file print;
   input ho ha @@;
   chisq = achisq[ho] - achisq[ha];
   df = adf[ho] - adf[ha];
   p = 1 - probchi( chisq, df);
   if _n_ = 1 then put
      / +indent 'model comparison   chi**2   df   p-value'
      / +indent '---------------------------------------';
   put +indent +3 ho ' versus ' ha @18 +indent chisq 8.4 df 5. p 9.4;
datalines;
2 1    3 2    4 2    5 2    7 2    7 4    7 5    6 2
;
```

The DATA step displays the table in Output 26.23.8.

Output 26.23.8 Career Aspiration Data: Model Comparisons

```
                Peer Influences on Aspiration: Haller & Butterworth (1960)
                                              '

                  model comparison   chi**2   df   p-value
                  ---------------------------------------
                     2  versus 1     7.0565   15    0.9561
                     3  versus 2     3.9668    1    0.0464
                     4  versus 2     1.9284    1    0.1649
                     5  versus 2     0.0048    1    0.9448
                     7  versus 2     6.2769    2    0.0433
                     7  versus 4     4.3485    1    0.0370
                     7  versus 5     6.2721    1    0.0123
                     6  versus 2    14.3778    2    0.0008
```

Although none of the seven models can be rejected when tested against the alternative of an unrestricted covariance matrix, the model comparisons make it clear that there are important differences among the models. Schwarz's Bayesian criterion indicates Model 5 as the model of choice. The constraints added to Model 5 in Model 7 can be rejected (p=0.0123), while Model 5 cannot be rejected when tested against the less constrained Model 2 (p=0.9448). Hence, among the small number of models considered, Model 5 has strong statistical support. However, as Loehlin (1987, p. 106) points out, many other models for these data could be constructed. Further analysis should consider, in addition to simple modifications of the models, the possibility that more than one friend could influence a boy's aspirations, and that a boy's ambition might have some effect on his choice of friends. Pursuing such theories would be statistically challenging.

Example 26.24: 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 CALIS to fit this latent growth curve model, the random intercept and effect are treated as if they were covarying latent factors. To make them stand out more as latent variables, the random intercept and slope are renamed as f_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 calis method=ml data=growth nostand noparmname;
   lineqs
      y1 = 0. * Intercept + f_alpha                 + e1,
      y2 = 0. * Intercept + f_alpha  +  1 * f_beta + e2,
      y3 = 0. * Intercept + f_alpha  +  2 * f_beta + e3,
      y4 = 0. * Intercept + f_alpha  +  3 * f_beta + e4,
      y5 = 0. * Intercept + f_alpha  +  4 * f_beta + e5;
   variance
      f_alpha f_beta,
      e1-e5 = 5 * evar;
   mean
      f_alpha f_beta;
   cov
      f_alpha f_beta;
   fitindex on(only)=[chisq df probchi];
run;
```

In the LINEQS model specification, f_alpha and f_beta are treated as latent factors representing the random intercept and random slope, respectively. The f_ prefix for latent factors is required as a convention in the LINEQS modeling language. See the sections "Naming Variables in the LINEQS Model" on page 1206 and "Naming Variables and Parameters" on page 1238 for details.

Notice that you need to set the ordinary (non-random) intercepts for endogenous variables to zero by the **0.*Intercept** specification because non-random intercepts for observed endogenous variables are default parameters in the LINEQS model. Because you have already used f_alpha as the random intercept, you must turn off the default non-random intercept term for the observed endogenous variables y1–y5. Otherwise, your latent growth curve model might be over-parameterized.

At $T_1 = 0$, y_1 represents the initial confidence measurement so that it is not subject to the random effect f_beta. The next four measurements y_2, y_3, y_4, and y_5 are measured at time points T_2, T_3, T_4, and T_5, respectively. These are fixed time points with constant values 1, 2, 3, and 4, respectively, in the equations of the LINEQS statement.

The means, variances and covariances of f_alpha and f_beta are parameters in the model. The variances of these two latent variables are specified in the VARIANCE statement, while their covariance is specified in the COV statement. The means of f_alpha and f_beta are specified in the MEAN statement. Unlike the specification for the variances of e1–e5. All these parameters for the latent factors are unnamed because you do not need to constrain them by references.

The error variances for e1–e5 are also specified in the VARIANCE statement. Using the shorthand notation 5 * evar, the parameter name evar is repeated five times for the five error variances. This constrains the error variances for e1–e5 to be equal.

You also use some special printing options in this example. In the PROC CALIS statement, the NOSTAND option is specified because standardized solution is not of interest. The reason is that $y1$-$y5$ were already measured on comparable scales, making standardization unnecessary for interpretations. Another printing option specified is the NOPARMNAME option in the PROC CALIS statement. This option suppresses the printing of parameter names in the output for estimation. This makes the output look more concise when you do not need to make references to the parameter names. Still another printing option used is the ON(ONLY)= option of the FITINDEX statement. This option trims down the display of fit indices to include only those listed in the option. See the FITINDEX statement on page 1082 for details.

Output 26.24.1 shows the fit summary table.

Output 26.24.1 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 26.24.1, 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 26.24.2 Estimation of Random Intercepts and Effects with Constrained Error Variances

Estimates for Variances of Exogenous Variables

Variable Type	Variable	Estimate	Standard 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

Var1	Var2	Estimate	Standard Error	t Value
f_alpha	f_beta	-0.35281	1.13815	-0.30998

Mean Parameters

Variable Type	Variable	Estimate	Standard Error	t Value
Latent	f_alpha	14.15875	1.02906	13.75890
	f_beta	4.04813	0.27563	14.68665

In Output 26.24.2, 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. They are both significantly different from zero.

Given that the model does not fit that well, perhaps you should not take the interpretations of these estimates so seriously. Knowing that the distribution of the errors might have been time-dependent, you now try to improve the fit of the model by relaxing the constraint about common error variances. You can use the following specifications:

```
proc calis method=ml data=growth nostand noparmname;
   lineqs
      y1 = 0. * Intercept + f_alpha                + e1,
      y2 = 0. * Intercept + f_alpha  +  1 * f_beta + e2,
      y3 = 0. * Intercept + f_alpha  +  2 * f_beta + e3,
      y4 = 0. * Intercept + f_alpha  +  3 * f_beta + e4,
      y5 = 0. * Intercept + f_alpha  +  4 * f_beta + e5;
   variance
      f_alpha f_beta,
      e1-e5;
   mean
      f_alpha f_beta;
   cov
      f_alpha f_beta;
   fitindex on(only)=[chisq df probchi];
run;
```

In this new specification, there is only one change in the VARIANCE statement from the previous specification. That is, you now specify only the error variables without putting parameter names for them. This makes the variances of e1–e5 free (unconstrained) parameters in the model.

Output 26.24.3 shows the model fit summary.

Output 26.24.3 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

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.

Output 26.24.4 shows the estimation results.

Output 26.24.4 Estimation of Random Intercepts and Effects with Unconstrained Error Variances

Estimates for Variances of Exogenous Variables

Variable Type	Variable	Estimate	Standard 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

Var1	Var2	Estimate	Standard Error	t Value
f_alpha	f_beta	0.35291	0.90366	0.39054

Mean Parameters

Variable Type	Variable	Estimate	Standard 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 you need to collect more data and fit the models again to examine the significance of the covariance between the random intercept and slope.

Example 26.25: Higher-Order and Hierarchical Factor Models

In this example, confirmatory higher-order and hierarchical factor models are fitted by PROC CALIS.

In higher-order factor models, factors are at different levels. The higher-order factors explain the relationships among factors at the next lower level, in the same way that the first-order factors explain the relationships among manifest variables. For example, in a two-level higher oder 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 the bifactor solution. In a bifactor solution, there are two classes of factors—group factors and a general factor. For example, in the preceding hierarchical factor pattern F1–F3 are group factors for different abilities and F4 is a general factor such as "intelligence" (see, for example, Holzinger and Swineford 1937). See Mulaik and Quartetti (1997) for more examples and distinctions among various types of hierarchical factor models. Certainly, if substantiated by your theory, hierarchical factor models with more than two layers are possible.

In this example, you use PROC CALIS to fit these two types of confirmatory factor models. First, you fit a second-order factor model to a real data set. Then you fit a bifactor model to the same data set. In the final section of this example, an informal account of the relationship between the higher-order and hierarchical factor models is attempted. Techniques for constraining parameters using PROC CALIS are also shown. This final section might be too technical in the first reading. Interested readers are referred to articles by Mulaik and Quartetti (1997), Schmid and Leiman (1957), and Yung, Thissen, and McLeod (1999) for more details.

A Second-Order Factor Analysis Model

In this section, a second-order confirmatory factor analysis model is applied to a correlation matrix of Thurstone reported by McDonald (1985). The correlation matrix is read into a SAS data set in the following statements:

```
data Thurst(type=corr);
title "Example of THURSTONE resp. McDONALD (1985, p.57, p.105)";
   _type_ = 'corr'; input _name_ $ V1-V9;
   label V1='Sentences' V2='Vocabulary' V3='Sentence Completion'
         V4='First Letters' V5='Four-letter Words' V6='Suffices'
         V7='Letter series' V8='Pedigrees' V9='Letter Grouping';
   datalines;
V1  1.       .      .      .      .      .      .      .      .
V2   .828   1.      .      .      .      .      .      .      .
V3   .776   .779   1.      .      .      .      .      .      .
V4   .439   .493    .460  1.      .      .      .      .      .
V5   .432   .464    .425   .674  1.      .      .      .      .
V6   .447   .489    .443   .590   .541  1.      .      .      .
V7   .447   .432    .401   .381   .402   .288  1.      .      .
V8   .541   .537    .534   .350   .367   .320   .555  1.      .
V9   .380   .358    .359   .424   .446   .325   .598   .452  1.
;
```

Variables in this data set are measures of cognitive abilities. Three factors are assumed for these nine variable V1–V9. These three factors are the first-order factors in the analysis. A second-order factor is also assumed to explain the correlations among the three first-order factors.

The following statements define a second-order factor model by using the LINEQS modeling language.

```
proc calis corr data=Thurst method=max nobs=213 nose nostand;
   lineqs
      V1      = X11 * Factor1                                + E1,
      V2      = X21 * Factor1                                + E2,
      V3      = X31 * Factor1                                + E3,
      V4      =             X42 * Factor2                    + E4,
      V5      =             X52 * Factor2                    + E5,
      V6      =             X62 * Factor2                    + E6,
      V7      =                         X73 * Factor3        + E7,
      V8      =                         X83 * Factor3        + E8,
      V9      =                         X93 * Factor3        + E9,
      Factor1 =                                   L1g * FactorG + E10,
      Factor2 =                                   L2g * FactorG + E11,
      Factor3 =                                   L3g * FactorG + E12;
   variance
      FactorG   = 1. ,
      E1-E12    = U1-U9 W1-W3;
   bounds
      0. <= U1-U9;
   fitindex ON(ONLY)=[chisq df probchi];
   /* SAS Programming Statements: Dependent parameter definitions */
      W1  = 1. - L1g * L1g;
      W2  = 1. - L2g * L2g;
      W3  = 1. - L3g * L3g;
run;
```

In the first nine equations of the LINEQS statement, variables V1–V3 are manifest indicators of latent factor Factor1, variables V4–V6 are manifest indicators of latent factor Factor2, and variables V7–V9 are manifest indicators of latent factor Factor3. In the last three equations of the LINEQS statement, the three first-order factors Factor1–Factor3 are explained by a common source: FactorG. Hence, Factor1–Factor3 are correlated due to the common source FactorG in the model.

An error term is added to each equation in the LINEQS statement. These error terms E1–E12 are needed because the factors are not assumed to be perfect predictors of the corresponding outcome variables.

In the VARIANCE statement, you specify variance parameters for all independent or exogenous variables in the model: FactorG, and E1–E12. The variance of FactorG is fixed at one for identification. Variances for E1–E9 are given parameter names U1–U9, respectively. Variances for E10–E12 are given parameter names W1–W3, respectively. Note that for model identification purposes, W1–W3 are defined as dependent parameters in the SAS programming statements. That is,

$$W_i = 1. - L_{ig}^2 \quad (i = 1, 2, 3)$$

These dependent parameter definitions ensure that the variances for Factor1–Factor3 are fixed at ones for identification.

In the BOUNDS statement, you specify that variance parameters U1–U9 must be positive in the solution.

In addition to the statements for model specification, you specify some output control options in the PROC CALIS statement. You use the NOSE and NOSTAND options suppress the display of standard errors and standardized results. In the FITINDEX statement, the ON(ONLY)= option requests only the model fit chi-square and its associated degrees of freedom and p-value be shown in the fit summary table. Using printing options in PROC CALIS to reduce the amount the of printout is a good practice. It makes your output more focused, as you output only what you need in a particular situation.

In Output 26.25.1, parameters and their initial values, gradients, and bounds are shown.

Output 26.25.1 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 the independent parameters.

Output 26.25.2 shows the information about optimization—iteration history and the convergence status.

Output 26.25.2 Optimization

Parameter Estimates	21
Functions (Observations)	45
Lower Bounds	9
Upper Bounds	0

Optimization Start

Active Constraints	0	Objective Function	0.5693888709
Max Abs Gradient Element	0.5749163348	Radius	1.8533033852

Iter	Restarts	Func Calls	Act Con	Objective Function	Obj Fun Change	Max Abs Gradient Element	Lambda	Actual Over Pred 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.

Output 26.25.3 shows the fit summary table. The chi-square model fit value is 38.196, with df=24, and p=0.033. This indicates a satisfactory model fit.

Output 26.25.3 Fit Summary

```
                    Fit Summary

          Chi-Square                38.1963
          Chi-Square DF                  24
          Pr > Chi-Square            0.0331
```

Output 26.25.4 shows the fitted equations with final estimates. Interpretations of these loadings are not done here. The last table in this output shows various variance estimates. These estimates are classified by whether they are for the latent variables, error variables, or disturbance variables.

Output 26.25.4 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
```

Output 26.25.4 *continued*

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 indeed verify the required equality, as shown in the following:

$$1.0000 = (0.8221)^2 + 0.32420$$

A Bifactor Model

A bifactor model (or a hierarchical factor model with two layers) for the same data set is now considered. In this model, the same set of factors as in the preceding higher-order factor model are used. The most notable difference is that the second-order factor FactorG in the higher-order factor model is no longer a factor of the first-order factors Factor1–Factor3. Instead, FactorG, like Factor1–Factor3, now also serves as a factor of the observed variable V1–V9. Unlike Factor1–Factor3, FactorG is considered to be a *general* factor in the sense that *all* observed variables have direct functional relationships with it. In contrast, Factor1–Factor3 are *group* factors in the sense that each of them has a direct functional relationship with only one group of *observed* variables. Because of the coexistence of a general factor and group factors at the same factor level, such a hierarchical model is also called a bifactor model.

The bifactor model is specified in the following statements:

```
proc calis corr data=Thurst method=max nobs=213 nose nostand;
   lineqs
      V1 = X11 * Factor1                        + X1g * FactorG + E1,
      V2 = X21 * Factor1                        + X2g * FactorG + E2,
      V3 = X31 * Factor1                        + X3g * FactorG + E3,
      V4 =           X42 * Factor2              + X4g * FactorG + E4,
      V5 =           X52 * Factor2              + X5g * FactorG + E5,
      V6 =           X62 * Factor2              + X6g * FactorG + E6,
      V7 =                     X73 * Factor3 + X7g * FactorG + E7,
      V8 =                     X83 * Factor3 + X8g * FactorG + E8,
      V9 =                     X93 * Factor3 + X9g * FactorG + E9;
   variance
      Factor1-Factor3 = 3 * 1.,
      FactorG         = 1. ,
      E1-E9           = U1-U9;
   cov
      Factor1-Factor3 FactorG = 6 * 0.;
   bounds
      0. <= U1-U9;
   fitindex ON(ONLY)=[chisq df probchi];
run;
```

In the LINEQS statement, there are only nine equations for the manifest variables in the model. Unlike the second-order factor model fitted previously, Factor1–Factor3 are no longer functionally related to FactorG and therefore there are no equations with Factor1–Factor3 as outcome variables.

The factor variances are all fixed at 1 in the VARIANCE statement. The variance parameters for E1–E9 are named U1–U9, respectively. The BOUNDS statement, again, is specified so that only positive estimates are accepted for error variance estimates.

All factors in the bifactor model are uncorrelated. In the COV statement, you specify that the six covariances among Factor1–Factor3 and FactorG are all zero. This specification is necessary because by default exogenous variables (excluding error terms) in the LINEQS model are correlated.

Like the previous PROC CALIS run, options are specified in the PROC CALIS and the FITINDEX statements to reduce the amount of default output.

There are more parameters in this model than in the preceding higher-order factor model, as shown in Output 26.25.5, which shows the optimization information.

Output 26.25.5 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

Output 26.25.5 *continued*

Iter	Restarts	Func Calls	Act Con	Objective Function	Obj Fun Change	Max Abs Gradient Element	Lambda	Actual Over Pred 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 26.25.6

Output 26.25.6 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 26.25.7. Estimates are left uninterpreted because they are not the main interest of this example.

Output 26.25.7 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

               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 26.25.8 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 CALIS.

For some practical researchers, the technical details involved in the next section might not be of interest and therefore could be skipped.

A Constrained Bifactor Model and Its Equivalence to the Second-Order Factor Model

To demonstrate that the second-order factor model is indeed nested within the bifactor model, a constrained bifactor model is fitted in this section. This constrained bifactor model is essentially the same as the preceding bifactor model, but with additional constraints on the factor loadings. Hence, the constrained bifactor model is nested within the unconstrained bifactor model.

Furthermore, if it can be shown that the constrained bifactor model is equivalent to the previous second-order factor, then the second-order factor model must also be nested within the unconstrained bifactor model. As a result, it justifies the chi-square difference test conducted in the previous section.

The construction of such a constrained bifactor model is based on Yung, Thissen, and McLeod (1999). In the following statements, a constrained bifactor model is specified.

```
proc calis corr data=Thurst method=max nobs=213 nose nostand;
   lineqs
      V1 = X11 * Factor1                          + X1g * FactorG + E1,
      V2 = X21 * Factor1                          + X2g * FactorG + E2,
      V3 = X31 * Factor1                          + X3g * FactorG + E3,
      V4 =             X42 * Factor2              + X4g * FactorG + E4,
      V5 =             X52 * Factor2              + X5g * FactorG + E5,
      V6 =             X62 * Factor2              + X6g * FactorG + E6,
      V7 =                         X73 * Factor3 + X7g * FactorG + E7,
      V8 =                         X83 * Factor3 + X8g * FactorG + E8,
      V9 =                         X93 * Factor3 + X9g * FactorG + E9;
   variance
      Factor1-Factor3 = 3 * 1.,
      FactorG         = 1. ,
      E1-E9           = U1-U9;
   cov
      Factor1-Factor3 FactorG = 6 * 0.;
   bounds
      0. <= U1-U9;
   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, you add a PARAMETERS statement and nine SAS programming statements to the previous bifactor model. In the PARAMETERS statement, three new independent parameters are added: p1, p2, and p3. These parameters represent the proportions that constrain the factor loadings of the observed variables on the group factors Factor1–Factor3 and the general factor FactorG. They are all free parameters and have initial values at 0.5. The next nine SAS programming statements represent the proportionality constraints imposed. For example, X1g–X3g are now dependent parameters expressed as functions of p1, X11, X21, and X31. Adding three new parameters (in the PARAMETERS statement) and redefining nine original parameters as dependent (in the SAS programming statements) is equivalent to adding six ($= 9 - 3$) constraints to the original bifactor model. Mathematically, the additional statements in specifying the constrained bifactor model realizes the following six constraints:

$$\frac{\text{X1g}}{\text{X11}} = \frac{\text{X2g}}{\text{X21}} = \frac{\text{X3g}}{\text{X31}}$$

$$\frac{\text{X4g}}{\text{X42}} = \frac{\text{X5g}}{\text{X52}} = \frac{\text{X6g}}{\text{X62}}$$

$$\frac{\text{X7g}}{\text{X73}} = \frac{\text{X8g}}{\text{X83}} = \frac{\text{X9g}}{\text{X93}}$$

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 26.25.10. But first, it is also useful to consider the optimization information of the constrained bifactor model, which is shown in Output 26.25.9.

Output 26.25.9 Optimization

Parameter Estimates	21
Functions (Observations)	45
Lower Bounds	9
Upper Bounds	0

Optimization Start

Active Constraints	0	Objective Function	6.8290849536
Max Abs Gradient Element	8.9903038299	Radius	46.912034073

Iter	Restarts	Func Calls	Act Con	Objective Function	Obj Fun Change	Max Abs Gradient Element	Lambda	Actual Over Pred Change
1*	0	4	0	4.42090	2.4082	1.5979	0.195	0.121
2*	0	6	0	2.42951	1.9914	2.4449	0.195	1.700
3*	0	8	0	1.52546	0.9040	2.1675	0.781	1.657
4*	0	10	0	0.86547	0.6600	1.2570	0.781	1.552
5*	0	12	0	0.50425	0.3612	0.4075	0.781	1.351
6*	0	14	0	0.41864	0.0856	1.1698	0.195	0.216
7*	0	16	0	0.28298	0.1357	0.2355	0.0488	0.983
8	0	19	0	0.27010	0.0129	0.9602	0.00098	0.327
9	0	21	0	0.21798	0.0521	0.2433	0	0.751
10*	0	23	0	0.20781	0.0102	0.0373	0.0488	0.943
11*	0	26	0	0.20251	0.00530	0.0273	0.0918	1.064
12	0	28	0	0.19176	0.0108	0.0718	0.0515	1.008
13	0	30	0	0.18575	0.00601	0.1407	0.00393	0.562
14	0	32	0	0.18024	0.00551	0.0224	0	0.921
15	0	34	0	0.18017	0.000064	0.00161	0	1.067
16	0	36	0	0.18017	2.357E-6	0.000538	0	1.344
17	0	38	0	0.18017	3.389E-7	0.000216	0	1.380
18	0	40	0	0.18017	5.146E-8	0.000075	0	1.389
19	0	42	0	0.18017	7.886E-9	0.000031	0	1.391
20	0	44	0	0.18017	1.211E-9	0.000012	0	1.392

Optimization Results

Iterations	20	Function Calls	47
Jacobian Calls	22	Active Constraints	0
Objective Function	0.1801712147	Max Abs Gradient Element	0.0000117927
Lambda	0	Actual Over Pred Change	1.3915929449
Radius	0.0002502182		

Convergence criterion (GCONV=1E-8) satisfied.

As shown Output 26.25.9, there are 21 independent parameters in the constrained bifactor model for the 45 “Functions (Observations).” These numbers match those of the second-order factor model exactly. The optimization shows some problems in initial iterations. The iteration numbers with asterisks indicate that the Hessian matrix is not positive definite in those iterations. But as long as the final converged iteration is not marked with an asterisk, the problems exhibited in early iterations do not raise any concern, as in the current case. Next, the fit summary is shown in Output 26.25.10.

Output 26.25.10 Model Fit

Fit Summary

Chi-Square	38.1963
Chi-Square DF	24
Pr > Chi-Square	0.0331

In Output 26.25.10, 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 26.25.11, estimation results of the constrained bifactor model are shown.

Output 26.25.11 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
```

Output 26.25.11 *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	0.18150
	E2	U2	0.16493
	E3	U3	0.26713
	E4	U4	0.30150
	E5	U5	0.36450
	E6	U6	0.50641
	E7	U7	0.39034
	E8	U8	0.48136
	E9	U9	0.50511

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 26.25.4 for the second-order factor model match those of the constrained bifactor model exactly in Output 26.25.11. 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 26.25.4 for the second-order factor model are now transformed to yield the loading estimates in the constrained bifactor model.

Denote $\mathbf{P}_1$ as the first-order factor loading matrix, $\mathbf{P}_2$ as the second-order factor loading matrix, and $\mathbf{U}_1^2$ be the matrix of variances for disturbances. That is, for the second-order factor model,

$$\mathbf{P}_1 = \begin{pmatrix} 0.9047 & 0 & 0 \\ 0.9138 & 0 & 0 \\ 0.8561 & 0 & 0 \\ 0 & 0.8358 & 0 \\ 0 & 0.7972 & 0 \\ 0 & 0.7026 & 0 \\ 0 & 0 & 0.7808 \\ 0 & 0 & 0.7202 \\ 0 & 0 & 0.7035 \end{pmatrix}$$

$$\mathbf{P}_2 = \begin{pmatrix} 0.8221 \\ 0.7818 \\ 0.8150 \end{pmatrix}$$

$$\mathbf{U}_1^2 = \begin{pmatrix} 0.3242 & 0 & 0 \\ 0 & 0.3888 & 0 \\ 0 & 0 & 0.3358 \end{pmatrix}$$

According to Yung, Thissen, and McLeod (1999), the transformation to obtain the estimates in the equivalent constrained bifactor model is:

$$\begin{aligned}\mathbf{L}_1 &= \mathbf{P}_1\mathbf{U}_1 \\ \mathbf{L}_2 &= \mathbf{P}_1\mathbf{P}_2\end{aligned}$$

where $\mathbf{L}_1$ is the matrix of the first-layer factor loadings (that is, loadings on group factors Factor1–Factor3), and $\mathbf{L}_2$ 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 $\mathbf{L}_1$ and $\mathbf{L}_2$ shows that:

$$\mathbf{L}_1 = \begin{pmatrix} 0.5151 & 0 & 0 \\ 0.5203 & 0 & 0 \\ 0.4875 & 0 & 0 \\ 0 & 0.5212 & 0 \\ 0 & 0.4971 & 0 \\ 0 & 0.4381 & 0 \\ 0 & 0 & 0.4525 \\ 0 & 0 & 0.4173 \\ 0 & 0 & 0.4077 \end{pmatrix}$$

$$\mathbf{L}_2 = \begin{pmatrix} 0.7438 \\ 0.7512 \\ 0.7038 \\ 0.6534 \\ 0.6232 \\ 0.5493 \\ 0.6364 \\ 0.5870 \\ 0.5734 \end{pmatrix}$$

With very minor numerical differences and ignorable sign changes, these transformation results match the estimated loadings observed in Output 26.25.11 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 26.26: Linear Relations among Factor Loadings

In this example, you use the FACTOR modeling language of PROC CALIS to specify a confirmatory factor analysis model with linear constraints on loadings. You use SAS programming statements to set the constraints. This example also discusses the differences between fitting covariance structures and correlation structures in the current modeling context.

The correlation matrix of six variables from Kinzer and Kinzer (N=326) is used by Guttman (1957) as an example that yields an approximate simplex. McDonald (1980) uses this data set as an example of factor analysis where he assumes that the loadings on the second factor are linear functions of the loadings on the first factor. Let **B** be the factor loading matrix containing the two factors and six variables so that:

$$\mathbf{B} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \\ b_{51} & b_{52} \\ b_{61} & b_{62} \end{pmatrix}$$

and

$$b_{j2} = \alpha + \beta b_{j1}, \quad j = 1, \ldots, 6$$

The correlation structures are represented by:

$$\mathbf{P} = \mathbf{BB}' + \mathbf{\Psi}$$

where $\mathbf{\Psi} = \mathrm{diag}(\psi_{11}, \psi_{22}, \psi_{33}, \psi_{44}, \psi_{55}, \psi_{66})$ 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 calis data=kinzer nobs=326 nose;
   factor
      factor1 ---> var1-var6   = b11 b21 b31 b41 b51 b61 (6 *.6),
      factor2 ---> var1-var6   = b12 b22 b32 b42 b52 b62;
   pvar
      factor1-factor2 = 2 * 1.,
      var1-var6       = psi1-psi6 (6 *.3);
   cov
      factor1 factor2 = 0.;
   parameters alpha (1.);
   /* SAS Programming Statements to define dependent parameters */
   b12 = alpha - b11;
   b22 = alpha - b21;
   b32 = alpha - b31;
   b42 = alpha - b41;
   b52 = alpha - b51;
   b62 = alpha - b61;
   fitindex on(only)=[chisq df probchi];
run;
```

In the FACTOR statement, you specify two factors, named factor1 and factor2, for the variables. In this model, all manifest variables have nonzero loadings on the two factors. These loading parameters are specified after the equal signs and are named with the prefix 'b.' You specify the initial estimates in the parentheses for the parameters in the first entry of the FACTOR statement. The loadings in the first entry are all free parameters with initial estimates of .6. In the second entry of the FACTOR statement, you specify the Loadings of var1–var6 on factor2. However, these parameters are dependent, as shown in the SAS programming statements. Initial values for these dependent parameters are thus unnecessary.

In the PVAR statement, the factor variances are fixed at ones, while the error variances of the variables are free parameters named psi1–psi6. Again, you provide initial estimates for these error variance parameters. All have the initial value of 0.3.

An additional parameter alpha is specified in the PARAMETERS statement with an initial value of 1. Then, you use six SAS programming statements to define the loadings on the second factor as functions of the loadings on the first factor. Lastly, the FITINDEX statement is used to trim the results in the fit summary table.

In the specification, there are twelve loadings in the FACTOR statement and six error variances in the PVAR statement. Adding the parameter alpha in the list, there are 19 parameters in total. However, the loading parameters are not all independent of each other. As defined in the SAS programming statements, six loadings are dependent. This reduces the number of free parameters to 13. Hence the degrees of freedom for the model is $8 = 21 - 13$. Notice that the factor variances are fixed at 1, as specified in the PVAR statement, and covariance among the two factors is fixed at zero, as specified in the COV statement.

Output 26.26.1 shows a concise fit summary table. The chi-square test statistic of model fit is 10.337 with df=8 (p=0.242). This indicates a good model fit.

Output 26.26.1 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 26.26.2, and the estimated error variances and the estimate for alpha are presented in Output 26.26.3.

Output 26.26.2 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 26.26.3 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 those reported in Browne (1982). Notice that there are no standard error estimates in the output, as requested by the NOSE option in the PROC CALIS statement. Standard error estimates are not of interest in this example.

In fitting the preceding factor model, wrong covariance structures rather than the intended correlation structures have been specified. As pointed out by Browne (1982), fitting such covariance structures directly is not entirely appropriate for analyzing correlations. For example, when fitting the correlation structures, the diagonal elements of **P** must always be fixed ones. This fact has never been enforced in the preceding specification. A simple check of the estimates will illustrate the problem. In Output 26.26.2, the loading estimates of VAR1 on the two factors are 0.3609 and 0.6174, respectively. In Output 26.26.3, the error variance estimate for VAR1 is 0.53036. The fitted variance of VAR1 can therefore be computed by the following equation:

$$\text{fitted variance} = 0.3609^2 + 0.6174^2 + 0.53036 = 1.0418$$

This fitted value is quite a bit off from 1.00, as required for the standardized variance of VAR1.

Fortunately, even though the wrong covariance structure model has been analyzed, the preceding analysis is not completely useless. For the current confirmatory factor model, according to Browne (1982) the estimates obtained from fitting the wrong covariance structure model are still consistent (as if they were estimating the population parameters in the correlation structures). However, the chi-square test statistic as reported previously is not correct.

Note that using the CORR option in the PROC CALIS statement will not solve the problem. By specifying the CORR option you merely request PROC CALIS to use the correlation matrix directly as a covariance matrix in the objective function for model fitting. It still would not constrain the fitting of the diagonal elements to 1 during estimation.

In the next section, a solution to the correlation analysis problem is suggested. It is not claimed that this is the only solution or the best solution. Alternative treatments of the problem are possible.

Fitting the Correct Correlation Structures

This main idea of this solution is to embed the intended correlation structures (with correct constraints on the diagonal elements of the correlation matrix) into a covariance structure model so that the estimation methods of PROC CALIS can be applied legitimately to the specially constructed covariance structures.

First, the issue of the fixed ones on the diagonal of the correlation structure model is addressed. That is, the diagonal elements of the correlation structures represented by $(\mathbf{BB}' + \mathbf{\Psi})$ 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:

$$\mathbf{\Psi}_{jj} = 1. - b_{j1}^2 - b_{j2}^2, \quad j = 1, \ldots, 6$$

Other constraints might also serve the purpose, but the proposed constraints here are the most convenient and intuitive.

Now, due to the fact that discrepancy functions used in PROC CALIS are derived for covariance matrices rather than correlation matrices, PROC CALIS is essentially set up for analyzing covariance structures (with or without mean structures), but not correlation structures. Hence, the statistical theory behind PROC CALIS applies to covariance structure analysis, but it might not generalize to correlation structure analysis in all situations. Despite that, with some manipulations PROC CALIS can fit the correct correlation structures to the current data without compromising the statistical theory. These manipulations are now discussed. Recall that the correlation structures are represented by:

$$\mathbf{P} = \mathbf{B}\mathbf{B}' + \boldsymbol{\Psi}$$

As before, in the $\mathbf{B}$ matrix, there are six linear constraints on the factor loadings. In addition, the diagonal elements of $(\mathbf{B}\mathbf{B}' + \boldsymbol{\Psi})$ are constrained to ones, as done by defining the error variances as dependent parameters of the loadings in the preceding equation. To analyze the correlation structures by using PROC CALIS, a covariance structure model with such correlation structures embedded is now specified. That is, the covariance structure to be fitted by PROC CALIS is as follows:

$$\boldsymbol{\Sigma} = \mathbf{D}\mathbf{P}\mathbf{D}' = \mathbf{D}(\mathbf{B}\mathbf{B}' + \boldsymbol{\Psi})\mathbf{D}'$$

where $\mathbf{D}$ is a 6 x 6 diagonal matrix containing the population standard deviations for the manifest variables. Theoretically, it is legitimate that you analyze this covariance structure model for studying the embedded correlation structures. In addition, it does not matter whether your input matrix is a correlation or covariance matrix, or any rescaled covariance matrix (by multiplying any variables by any positive constants). You would get correct results if you could somehow specify these covariance structures correctly in PROC CALIS. However, there seems to be nowhere in PROC CALIS that you can specify the diagonal matrix $\mathbf{D}$ 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 $\mathbf{T} = \mathbf{D}\mathbf{B}$ and $\mathbf{K} = \mathbf{D}\boldsymbol{\Psi}\mathbf{D}'$. The covariance structure model of interest can now be rewritten as:

$$\boldsymbol{\Sigma} = \mathbf{T}\mathbf{T}' + \mathbf{K}$$

This form of covariance structures implies a confirmatory factor model with factor loading matrix $\mathbf{T}$ and error covariance matrix $\mathbf{K}$. 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 $\mathbf{B}$ and $\boldsymbol{\Psi}$ of the embedded correlation structures, you would define the model parameters as functions of this basic set of parameters of interest. This can be accomplished by using the PARAMETERS and the SAS programming statements.

All in all, you can use the following statements to set up such a confirmatory factor model with the desired correlation structures embedded.

```
proc calis 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;
   cov
      factor1 factor2 = 0.;
   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:

$$\mathbf{\Psi}_{jj} = 1. - b_{j1}^2 - b_{j2}^2, \quad j = 1, \ldots, 6$$

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 $\mathbf{K} = \mathbf{D\Psi D}'$, but in non-matrix forms:

$$t_{ji} = d_j b_{ji} \quad (j = 1, \ldots, 6; \quad i = 1, 2)$$

$$k_{jj} = d_j d_j \mathbf{\Psi}_{jj} \quad (j = 1, \ldots, 6)$$

where d_j denotes the j-th diagonal element of **D**.

The fit summary is presented in Output 26.26.4. 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 26.26.4 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 26.26.5. 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 26.26.5 Estimates of Loadings and Error Variances

Factor Loading Matrix

	factor1	factor2
var1	0.3448 [t11]	0.6367 [t12]
var2	0.3200 [t21]	0.6512 [t22]
var3	0.4873 [t31]	0.4778 [t32]
var4	0.5703 [t41]	0.3948 [t42]
var5	0.7741 [t51]	0.1964 [t52]
var6	0.6778 [t61]	0.3126 [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 26.26.6 as "additional" parameters.

Output 26.26.6 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 26.27: Multiple-Group Model for Purchasing Behavior

In this example, data were collected from customers who made purchases from a retail company during years 2002 and 2003. A two-group structural equation model is fitted to the data.

The variables are:

Spend02:	total purchase amount in 2002
Spend03:	total purchase amount in 2003
Courtesy:	rating of the courtesy of the customer service
Responsive:	rating of the responsiveness of the customer service
Helpful:	rating of the helpfulness of the customer service
Delivery:	rating of the timeliness of the delivery
Pricing:	rating of the product pricing
Availability:	rating of the product availability
Quality:	rating of the product quality

For the ratings scales, nine-point scales were used. Customers could respond 1 to 9 on these scales, with 1 representing "extremely unsatisfied" and 9 representing "extremely satisfied." Data were collected from two different regions, which are labeled as Region 1 ($N = 378$) and Region 2 ($N = 423$), respectively. The ratings were collected at the end of year 2002 so that they represent customers' purchasing experience in year 2002.

The central questions of the study are:

- How does the overall customer service affect the current purchases and predict future purchases?
- How does the overall product quality affect the current purchases and predict future purchases?
- Do current purchases predict future purchases?
- Do the two regions have different structural models for predicting the purchases?

In stating these research questions, you use several constructs that might or might not correspond to objective measurements. Current and future purchases are certainly measurable directly by the spending of the customers. That is, because customer service and product satisfaction and quality were surveyed between 2002 and 2003, Spend02 represents current purchases and Spend03 represents future purchases in the study. Both variables Spend02 and Spend03 are objective measurements without measurement errors. All you need to do is to extract the information from the transaction records. But how about hypothetical constructs such as customer service quality and product quality? How would you measure them in the model?

In measuring these hypothetical constructs, you might ask customers' perception about the service or product quality directly in a single question. A simple survey with two questions about the customer service and product qualities could then be what you need. These questions are called indicators (or indicator variables) of the underlying constructs. However, using just one indicator (question) for each of these hypothetical constructs would be quite unreliable—that is, measurement errors might dominate in the data collection process. Therefore, multiple indicators are usually recommended for measuring such hypothetical constructs.

There are two main advantages of using multiple indicators for hypothetical constructs. The first one is conceptual and the other is statistical and mathematical.

First, hypothetical constructs might conceptually be multifaceted themselves. Measuring a hypothetical construct by a single indicator does not capture the full meaning of the construct. For example, the product quality might refer to the durability of the product, the outlook of the product, the pricing of the product, and the availability of product, among others. The customer service quality might refer to the politeness of the customer service, the timeliness of the delivery, and the responsiveness of customer services, among others. Therefore, multiple indicators for a single hypothetical construct might be necessary if you want to cover the multifaceted aspects of a given hypothetical construct.

Second, from a statistical point of view, the reliability would be higher if you combine correlated indicators for a construct than if you use a single indicator only. Therefore, combining correlated indicators would lead to more accurate and reliable results.

One way to combine correlated indicators is to use a simple sum of them to represent the underlying hypothetical construct. However, a better way is to use the structural equation modeling technique that represents each indicator (variable) as a function of the underlying hypothetical construct plus an error term. In structural equation modeling, hypothetical constructs are constructed as latent factors, which are unobserved systematic (that is, non-error) variables. Theoretically, latent factors are free from measurement errors, and so the estimation through the structural equation modeling technique is more accurate than if you just use simple sums of indicators to represent hypothetical constructs. Therefore, a structural equation modeling approach is the method of the choice in the current analysis.

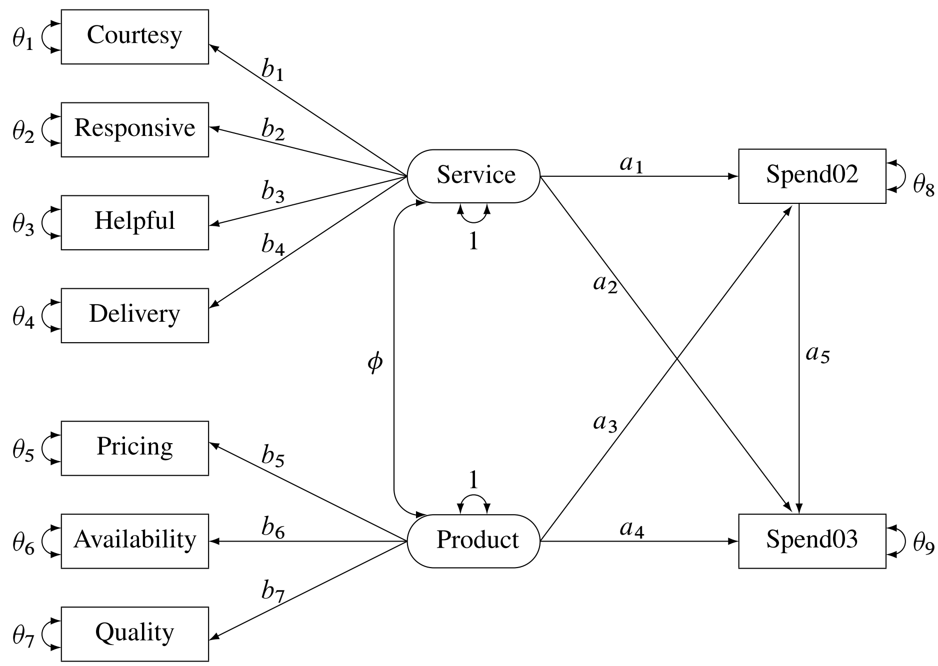
In practice, you must also make sure that there are enough indicators for the identification of the underlying latent factor, and hence the identification of the entire model. Necessary and sufficient rules for identification are very complicated to describe and are out of the scope of the current discussion (however, see Bollen 1989b for discussions of identification rules for various classes of structural equation models). Some simple rules of thumb might be useful as a guidance. For example, for unconstrained situations, you should at least have three indicators (variables) measured for a latent factor. Unfortunately, these rules of thumb do not guarantee identification in every case.

In this example, Service and Product are latent factors in the structural equation model which represent service and product qualities, respectively. In the model, these two latent factors are reflected by the ratings of the customers. Ratings on the Courtesy, Responsive, Helpful, and Delivery scales are indicators of Service. Ratings on the Pricing, Availability, and Quality scales are indicators of Product (that is, product quality).

A Path Diagram

A path diagram shown in Figure 26.7 represents the structural equation model for the purchase behavior. Observed or manifest variables are represented by rectangles, and latent variables are represented by ovals. As mentioned, two latent variables (factors), Service and Product, are created as overall measures of customer service and product qualities, respectively.

Figure 26.7 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 (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 that are attached to 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 that are attached to Service and Product represent the variances of these two latent variables. In the current model, both of these variances are fixed at one.

When the double-headed arrows point to two variables, they represent covariances in the path diagram. For example, in Figure 26.7 the covariance between Service and Product is represented by the parameter ϕ.

The Basic Path Model Specification

For the moment, it is hypothesized that both Region 1 and Region 2 data are fitted by the same model as shown in Figure 26.7. Once the path diagram is drawn, it is readily translated into the PATH modeling language. See the PATH statement on page 1137 for details about how to use the PATH modeling language to specify structural equation models.

To represent all the features in the path diagram in the PATH model language, you can use the following specification:

```
path
   Service ---> Spend02      = a1,
   Service ---> Spend03      = a1,
   Product ---> Spend02      = a3,
   Product ---> Spend03      = a4,
   Spend02 ---> Spend03      = a5,
   Service ---> Courtesy     = b1,
   Service ---> Responsive   = b2,
   Service ---> Helpful      = b3,
   Service ---> Delivery     = b4,
   Product ---> Pricing      = b5,
   Product ---> Availability = b6,
   Product ---> Quality      = b7;
pvar
   Courtesy Responsive Helpful
   Delivery Pricing
   Availability Quality = theta01-theta07,
   Spend02 = theta08,
   Spend03 = theta09,
   Service Product = 2 * 1.;
pcov
   Service Product = phi;
```

The PATH statement captures all the path coefficient specifications and the direction of the paths (single-headed arrows) in the path diagram. The first five paths define how Spend02 and Spend03 are predicted from the latent variables Service, Product, and Spend02. The next seven paths define the measurement model, which shows how the latent variables in the model relate to the observed indicator variables.

The PVAR statement captures the specification of the error variances and the variances of exogenous variables (that is, the double-headed arrows in the path diagram). The PCOV statement captures the specification of covariance between the two latent variables in the model (which is represented by the double-headed arrow that connects Service and Product in the path diagram).

You can also use the following simpler version of the PATH model specification for the path diagram:

```
path
   Service ---> Spend02  Spend03      ,
   Product ---> Spend02  Spend03      ,
   Spend02 ---> Spend03               ,
   Service ---> Courtesy Responsive
                Helpful Delivery      ,
   Product ---> Pricing  Availability
                Quality               ;
pvar
   Courtesy Responsive Helpful Delivery Pricing
   Availability Quality Spend02 Spend03,
   Service Product = 2 * 1.;
pcov
   Service Product;
```

There are two simplifications in this PATH model specification. First, you do not need to specify the parameter names if they are unconstrained in the model. For example, parameter a1 in the model is unique to the path effect from Service to Spend02. You do not need to name this effect because it is not constrained to be the same as any other parameter in the model. Similar, all the path coefficients (effects), error variances, and covariances in the path diagram are not constrained. Therefore, you can omit all the corresponding parameter name specifications in the PATH model specification. The only exceptions are the variances of Service and Product. Both are fixed constants 1 in the path diagram, and so you must specify them explicitly in the PVAR statement.

Second, you use a condensed way to specify the paths. In the first three path entries of the PATH statement, you specify how Spend02 and Spend03 are predicted from the latent variables Service, Product, and Spend02. Notice that in each path entry, you can define more than one path (single-headed arrow relationship). For example, in the first path entry, you specify two paths: one is Service--->Spend02 and the other is Service--->Spend03. In the last two path entries of the PATH statement, you define the relationships between the two latent constructs Spend and Service and their measured indicators. Each of these path entries specifies multiple paths (single-headed arrow relationships).

You use this simplified PATH specifications in the subsequent analysis.

A Restrictive Model with Invariant Mean and Covariance Structures

In this section, you fit a mean and covariance structure model to the data from two regions, as shows in the following DATA steps:

```
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
;
```

To include the analysis of the mean structures, you need to introduce the mean and intercept parameters in the model. Although various researchers propose some representation schemes that include the mean parameters in the path diagram, the mean parameters are not depicted in Figure 26.7. The reason is that representing the mean and intercept parameters in the path diagram would usually obscure the "causal" paths, which are of primary interest. In addition, it is a simple matter to specify the mean and intercept parameters in the MEAN statement without the help of a path diagram when you follow these principles:

- Each variable in the path diagram has a mean parameter that can be specified in the MEAN statement. For an exogenous variable, the mean parameter refers to the variable mean. For an endogenous variable, the mean parameter refers to the intercept of the variable.
- The means of exogenous observed variables are free parameters by default. The means of exogenous latent variables are fixed zeros by default.

- The intercepts of endogenous observed variables are free parameters by default. The intercepts of endogenous latent variables are fixed zeros by default.
- The total number of mean parameters should not exceed the number of observed variables.

Because all nine observed variables are endogenous (each has at least one single-headed arrow pointing to it) in the path diagram, you can specify these nine intercepts in the MEAN statement, as shown in the following specification:

```
mean
   Courtesy Responsive Helpful Delivery Pricing
   Availability Quality Spend02 Spend03;
```

However, the intercepts of endogenous observed variables are already free parameters by default and this MEAN statement specification is not necessary for the current situation. For the means of the latent variables Service and Product, you do not have any theoretical reasons to set them other than the default fixed zero. Hence, you do not need to set these mean parameters explicitly either. Consequently, to include the analysis of the mean structures with these default mean parameters, all you need to specify the MEANSTR option in the PROC CALIS statement, as shown in the following specification of the fitting of a constrained two-group model for the purchase data:

```
proc calis meanstr;
   group 1 / data=region1 label="Region 1" nobs=378;
   group 2 / data=region2 label="Region 2" nobs=423;
   model 1 / group=1,2;
      path
         Service ---> Spend02  Spend03      ,
         Product ---> Spend02  Spend03      ,
         Spend02 ---> Spend03               ,
         Service ---> Courtesy Responsive
                      Helpful Delivery      ,
         Product ---> Pricing  Availability
                      Quality               ;
      pvar
         Courtesy Responsive Helpful Delivery Pricing
         Availability Quality Spend02 Spend03,
         Service Product = 2 * 1.;
      pcov
         Service Product;
   run;
```

You use the GROUP statements to specify the data for the two regions. Using the DATA= options in the GROUP statements, you assign the Region 1 data to Group 1 and the Region 2 data to Group 2. You label the two groups by the LABEL= options. Because the number of observations is not defined in the data sets, you use the NOBS= options in the GROUP statements to provide this information.

In the MODEL statement, you specify in the GROUP= option that both Groups 1 and 2 are fitted by the same model—model 1. Next, the path model is specified. As discussed before, you do not need to specify the default mean parameters by using the MEAN statement because the MEANSTR option in the PROC CALIS statement already indicates the analysis of mean structures.

Output 26.27.1 presents a summary of modeling information. Each group is listed with its associated data set, number of observations, and its corresponding model and the model type. In the current analysis, the same model is fitted to both groups. Next, a table for the types of variables is presented. As intended, all nine observed (manifest) variables are endogenous, and all latent variables are exogenous in the model.

Output 26.27.1 Modeling Information and Variables

```
                              Modeling Information

Group  Label     Data Set       N Obs  Model    Type     Analysis

    1  Region 1  WORK.REGION1     378  Model 1  PATH     Means and Covariances
    2  Region 2  WORK.REGION2     423  Model 1  PATH     Means and Covariances

                        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. The fit summary table is presented in Output 26.27.2.

Output 26.27.2 Fit Summary

Fit Summary

Modeling Info	N Observations	801
	N Variables	9
	N Moments	108
	N Parameters	31
	N Active Constraints	0
	Baseline Model Function Value	0.5003
	Baseline Model Chi-Square	399.7468
	Baseline Model Chi-Square DF	72
	Pr > Baseline Model Chi-Square	<.0001
Absolute Index	Fit Function	3.5297
	Chi-Square	2820.2504
	Chi-Square DF	77
	Pr > Chi-Square	<.0001
	Z-Test of Wilson & Hilferty	43.2575
	Hoelter Critical N	29
	Root Mean Square Residual (RMSR)	28.2208
	Standardized RMSR (SRMSR)	2.1367
	Goodness of Fit Index (GFI)	0.9996
Parsimony Index	Adjusted GFI (AGFI)	0.9995
	Parsimonious GFI	1.0690
	RMSEA Estimate	0.2986
	RMSEA Lower 90% Confidence Limit	0.2892
	RMSEA Upper 90% Confidence Limit	0.3081
	Probability of Close Fit	<.0001
	Akaike Information Criterion	2882.2504
	Bozdogan CAIC	3058.5121
	Schwarz Bayesian Criterion	3027.5121
	McDonald Centrality	0.1804
Incremental Index	Bentler Comparative Fit Index	0.0000
	Bentler-Bonett NFI	-6.0551
	Bentler-Bonett Non-normed Index	-6.8265
	Bollen Normed Index Rho1	-5.5970
	Bollen Non-normed Index Delta2	-7.4997
	James et al. Parsimonious NFI	-6.4756

The model chi-square statistic is 2820.25. With df=77 and $p < .0001$, the null hypothesis for the mean and covariance structures is rejected. All incremental fit indices are negative. These negative indices indicate a bad model fit, as compared with the independence model. The same fact can be deduced by comparing the chi-square values of the baseline model and the fitted model. The baseline model has five degrees of freedom less (five parameters more) than the structural model but the chi-square value is only 399.747, much less than the model fit chi-square value of 2820.25. Because variables in social and behavioral sciences are almost always expected to correlate with each other, a structural model that explains relationships even worse than the baseline model is deemed inappropriate for the data. The RMSEA for the structural model is 0.2986, which also indicates a bad model fit. However, the GFI, AGFI, and parsimonious GFI indicate good model fit, which is a little surprising given the fact that all other indices indicate the opposite and the overall model is pretty restrictive in the first place.

There are some warnings in the output:

```
WARNING: Model 1. The estimated error variance for variable Spend02 is
         negative.
WARNING: Model 1. Although all predicted variances for the observed and
         latent variables are positive, the corresponding predicted
         covariance matrix is not positive definite. It has one negative
         eigenvalue.
```

PROC CALIS routinely checks the properties of the estimated variances and the predicted covariance matrix. It issues warnings when there are problems. In this case, the error variance estimate of Spend02 is negative, and the predicted covariance matrix for the observed and latent variables is not positive definite and has one negative eigenvalue. You can inspect Output 26.27.3, which shows the variance parameter estimates of the variables.

Output 26.27.3 Variance Estimates

Model 1. Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	Courtesy	_Parm13	2.59181	0.13600	19.05743
	Responsive	_Parm14	2.92423	0.15325	19.08205
	Helpful	_Parm15	2.44625	0.12320	19.85656
	Delivery	_Parm16	3.53408	0.18169	19.45095
	Pricing	_Parm17	2.52948	0.12784	19.78694
	Availability	_Parm18	1.57410	0.16884	9.32296
	Quality	_Parm19	2.41658	0.13230	18.26611
	Spend02	_Parm20	-14.40124	16.92863	-0.85070
	Spend03	_Parm21	22.79309	5.75120	3.96319
Exogenous	Service		1.00000		
	Product		1.00000		

The error variance estimate for Spend02 is -14.40, which is negative and might have led to the negative eigenvalue problem in the predicted covariance matrix for the observed and latent variables.

A Model with Unconstrained Parameters for the Two Regions

With all the bad model fit indications and the problematic predicted covariance matrix for the latent variables, you might conclude that an overly restricted model has been fit. Region 1 and Region 2 might not share exactly the same set of parameters. How about fitting a model at the other extreme with all parameters unconstrained for the two groups (regions)? Such a model can be easily specified, as shown in the following statements:

```
proc calis meanstr;
   group 1 / data=region1 label="Region 1" nobs=378;
   group 2 / data=region2 label="Region 2" nobs=423;
   model 1 / group=1;
      path
         Service ---> Spend02  Spend03      ,
         Product ---> Spend02  Spend03      ,
         Spend02 ---> Spend03               ,
         Service ---> Courtesy Responsive Helpful Delivery  ,
         Product ---> Pricing  Availability Quality ;
      pvar
         Courtesy Responsive Helpful Delivery Pricing
         Availability Quality Spend02 Spend03,
         Service Product = 2 * 1.;
      pcov
         Service Product;
   model 2 / group=2;
      refmodel 1/ allnewparms;
run;
```

Unlike the previous specification, in the current specification Group 2 is now fitted by a new model labeled as Model 2. This model is based on Model 1, as specified in REFMODEL statement. The ALLNEWPARMS option in the REFMODEL statement request that all parameters specified in Model 1 be renamed so that they become new parameters in Model 2. As a result, this specification gives different sets of estimates for Model 1 and Model 2, although both models have the same path structures and a comparable set of parameters.

The optimization converges without problems. The fit summary table is displayed in Output 26.27.4. The chi-square statistic is 29.613 (df= 46, $p = .97$). The theoretical model is not rejected. Many other measures of fit also indicate very good model fit. For example, the GFI, AGFI, Bentler CFI, Bentler-Bonett NFI, and Bollen nonnormed index delta2 are all close to one, and the RMSEA is close to zero.

Output 26.27.4 Fit Summary

Fit Summary

Modeling Info	N Observations	801
	N Variables	9
	N Moments	108
	N Parameters	62
	N Active Constraints	0
	Baseline Model Function Value	0.5003
	Baseline Model Chi-Square	399.7468
	Baseline Model Chi-Square DF	72
	Pr > Baseline Model Chi-Square	<.0001
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	0.0000
	RMSEA Upper 90% Confidence Limit	0.0000
	Probability of Close Fit	1.0000
	Akaike Information Criterion	153.6131
	Bozdogan CAIC	506.1365
	Schwarz Bayesian Criterion	444.1365
	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 gives you the same overall chi-square as in Output 26.27.4.

PROC CALIS also provides a table for comparing the relative model fit of the groups. In Output 26.27.5, basic modeling information and some measures of fit for the two groups are shown along with the corresponding overall measures.

Output 26.27.5 Fit Comparison among Groups

Fit Comparison Among Groups

		Overall	Region 1
Modeling Info	N Observations	801	378
	N Variables	9	9
	N Moments	108	54
	N Parameters	62	31
	N Active Constraints	0	0
	Baseline Model Function Value	0.5003	0.4601
	Baseline Model Chi-Square	399.7468	173.4482
	Baseline Model Chi-Square DF	72	36
Fit Index	Fit Function	0.0371	0.0023
	Percent Contribution to Chi-Square	100	3
	Root Mean Square Residual (RMSR)	0.0670	0.0172
	Standardized RMSR (SRMSR)	0.0220	0.0057
	Goodness of Fit Index (GFI)	1.0000	1.0000
	Bentler-Bonett NFI	0.9259	0.9950

Fit Comparison Among Groups

		Region 2
Modeling Info	N Observations	423
	N Variables	9
	N Moments	54
	N Parameters	31
	N Active Constraints	0
	Baseline Model Function Value	0.5363
	Baseline Model Chi-Square	226.2986
	Baseline Model Chi-Square DF	36
Fit Index	Fit Function	0.0681
	Percent Contribution to Chi-Square	97
	Root Mean Square Residual (RMSR)	0.0907
	Standardized RMSR (SRMSR)	0.0298
	Goodness of Fit Index (GFI)	1.0000
	Bentler-Bonett NFI	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 and f_1 and f_2 are the discrepancy functions for the groups. Even though T is chi-square distributed under the null hypothesis, in general the individual terms $(N_1-1) f_1$ and $(N_2 - 1) f_2$ are not chi-square distributed under the same null hypothesis. So when you compare the group fits by using the statistics in Output 26.27.5, you should treat those as descriptive measures only.

The current model is a special case where f_1 and f_2 are actually independent of each other. The reason is that there are no constrained parameters for the models fitted to the two groups. This would imply that the individual terms $(N_1 - 1) f_1$ and $(N_2 - 1) f_2$ are chi-square distributed under the null hypothesis. Nonetheless, this fact is not important to the group comparison of the descriptive statistics in Output 26.27.5. The values of f_1 and f_2 are shown in the row labeled "Fit Function." Group 1 (Region 1) is fitted better by its model ($f_1 = 0.0023$) than is Group 2 (Region 2) by its model ($f_2 = 0.0681$). Next, the percentage contributions to the overall chi-square statistic for the two groups are shown. Group 1 contributes only 3% ($= (N_1 - 1) f_1 / T \times 100\%$) while Group 2 contributes 97%. Other measures like 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. The next section considers a partially constrained model for the two groups of data.

A Model with Partially Constrained Parameters for the Two Regions

For multiple-group analysis, cross-group constraints are of primary interest and should be explored whenever appropriate. The first fitting with all model parameters constrained for the groups has been shown to be too restrictive, while the current model with no cross-group constraints fits very well—so well that it might have overfit unnecessarily. A multiple-group model between these extremes is now explored. The following statements specify such a partially constrained model:

```
proc calis meanstr modification;
   group 1 / data=region1 label="Region 1" nobs=378;
   group 2 / data=region2 label="Region 2" nobs=423;
   model 3 / label="Model for References Only";
      path
         Service ---> Spend02  Spend03      ,
         Product ---> Spend02  Spend03      ,
         Spend02 ---> Spend03               ,
         Service ---> Courtesy Responsive
                      Helpful Delivery      ,
         Product ---> Pricing  Availability
                      Quality               ;
      pvar
         Courtesy Responsive Helpful Delivery Pricing
         Availability Quality Spend02 Spend03,
         Service Product = 2 * 1.;
      pcov
         Service Product;
   model 1 / groups=1;
      refmodel 3;
      mean
         Spend02 Spend03 = G1_InterSpend02 G1_InterSpend03,
         Courtesy Responsive Helpful
         Delivery Pricing Availability
         Quality = G1_intercept01-G1_intercept07;
   model 2 / groups=2;
      refmodel 3;
      mean
         Spend02 Spend03 = G2_InterSpend02 G2_InterSpend03,
         Courtesy Responsive Helpful
         Delivery Pricing Availability
         Quality = G2_intercept01-G2_intercept07;
      simtests
         SpendDiff       = (Spend02Diff Spend03Diff)
         MeasurementDiff = (CourtesyDiff ResponsiveDiff
                            HelpfulDiff DeliveryDiff
                            PricingDiff AvailabilityDiff
                            QualityDiff);
      Spend02Diff      = G2_InterSpend02 - G1_InterSpend02;
      Spend03Diff      = G2_InterSpend03 - G1_InterSpend03;
      CourtesyDiff     = G2_intercept01  - G1_intercept01;
      ResponsiveDiff   = G2_intercept02  - G1_intercept02;
      HelpfulDiff      = G2_intercept03  - G1_intercept03;
      DeliveryDiff     = G2_intercept04  - G1_intercept04;
      PricingDiff      = G2_intercept05  - G1_intercept05;
      AvailabilityDiff = G2_intercept06  - G1_intercept06;
      QualityDiff      = G2_intercept07  - G1_intercept07;
run;
```

In this specification, you use a special model definition. Model 3 serves as a reference model. You are not going to fit this model directly to any data set, but the specifications of other two models makes reference to it. Model 3 is no different from the basic path model specification used in preceding examples. The PATH model specification reflects the path diagram in Figure 26.7.

Region 1 is fitted by Model 1, which makes reference to Model 3 by using the REFMODEL statement. In addition, you add the MEAN statement specification. You now specify the intercept parameters explicitly by using the parameter names G1_intercept01–G1_intercept07, G1_InterSpend02, and G1_InterSpend03. In previous examples, these intercept parameters are set by default by PROC CALIS. This explicit parameter naming serves the purpose of distinguishing these parameters from those for Model 2.

Region 2 is fitted by Model 2, which also refers to Model 3 by using the REFMODEL statement. You also specify a MEAN statement for this model with explicit specifications of the intercept parameters. You name these intercepts G2_intercept01–G2_intercept07, G2_InterSpend02, and G2_InterSpend03. The G2 prefix distinguishes these parameters from the corresponding intercept parameters in the parent model. All in all, this means that both Models 1 and 2 refers to Model 3, except that Model 2 uses a different set of intercept parameters. In other words, in this multiple-group model the covariance structures for the two regions are constrained to be the same, while the means structures are allowed to be unconstrained.

You request additional statistics or tests in the current PROC CALIS analysis. The MODIFICATION option in the PROC CALIS statement requests that the Lagrange multiplier tests and Wald tests be conducted. The Lagrange multiplier tests provide information about which constrained or fixed parameters could be freed or added so as to improve the overall model fit. The Wald tests provide information about which existing parameters could be fixed at zeros (eliminated) without significantly affecting the overall model fit. These tests are discussed in more detail when the results are presented.

In the SIMTESTS statement, two simultaneous tests are requested. The first simultaneous test is named SpendDiff, which includes two parametric functions Spend02Diff and Spend03Diff. The second simultaneous test is named MeasurementDiff, which includes seven parametric functions: CourtesyDiff, ResponsiveDiff, HelpfulDiff, DeliveryDiff, PricingDiff, AvailabilityDiff, and QualityDiff. The null hypothesis of these simultaneous tests is of the form

$$H_0 : t_i = 0 \quad (i = 1 \dots k)$$

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 26.27.6.

Output 26.27.6 Fit Summary

Fit Summary

Modeling Info	N Observations	801
	N Variables	9
	N Moments	108
	N Parameters	40
	N Active Constraints	0
	Baseline Model Function Value	0.5003
	Baseline Model Chi-Square	399.7468
	Baseline Model Chi-Square DF	72
	Pr > Baseline Model Chi-Square	<.0001
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	187.5461
	Bozdogan CAIC	414.9806
	Schwarz Bayesian Criterion	374.9806
	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 an α-level at 0.01 or larger is chosen. However, in practical structural equation modeling, the chi-square test is not the only criterion, or even an important criterion, for evaluating model fit. The RMSEA estimate for the current model is 0.0382, which indicates a good fit. The probability level of close fit is 0.9275, indicating that a good population fit hypothesis (that is, population RMSEA < 0.05) cannot be rejected. The GFI, AGFI, and parsimonious GFI all indicate good fit. However, the incremental indices show only a respectable model fit.

Comparison of the model fit to the groups is shown in Output 26.27.7.

Output 26.27.7 Fit Comparison among Groups

Fit Comparison Among Groups

		Overall	Region 1
Modeling Info	N Observations	801	378
	N Variables	9	9
	N Moments	108	54
	N Parameters	40	31
	N Active Constraints	0	0
	Baseline Model Function Value	0.5003	0.4601
	Baseline Model Chi-Square	399.7468	173.4482
	Baseline Model Chi-Square DF	72	36
Fit Index	Fit Function	0.1346	0.1261
	Percent Contribution to Chi-Square	100	44
	Root Mean Square Residual (RMSR)	0.1577	0.1552
	Standardized RMSR (SRMSR)	0.0678	0.0792
	Goodness of Fit Index (GFI)	1.0000	1.0000
	Bentler-Bonett NFI	0.7310	0.7260

Fit Comparison Among Groups

		Region 2
Modeling Info	N Observations	423
	N Variables	9
	N Moments	54
	N Parameters	31
	N Active Constraints	0
	Baseline Model Function Value	0.5363
	Baseline Model Chi-Square	226.2986
	Baseline Model Chi-Square DF	36
Fit Index	Fit Function	0.1422
	Percent Contribution to Chi-Square	56
	Root Mean Square Residual (RMSR)	0.1599
	Standardized RMSR (SRMSR)	0.0557
	Goodness of Fit Index (GFI)	1.0000
	Bentler-Bonett NFI	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.

The constrained parameter estimates for the two regions are shown in Output 26.27.8.

Output 26.27.8 Estimates of Path Coefficients and Other Covariance Parameters

Model 1. PATH List

Path			Parameter	Estimate	Standard Error	t Value
Service	--->	Spend02	_Parm01	0.37475	0.21318	1.75795
Service	--->	Spend03	_Parm02	0.53851	0.20840	2.58401
Product	--->	Spend02	_Parm03	0.80372	0.21939	3.66347
Product	--->	Spend03	_Parm04	0.59879	0.22144	2.70409
Spend02	--->	Spend03	_Parm05	0.08952	0.03694	2.42326
Service	--->	Courtesy	_Parm06	0.72418	0.07989	9.06482
Service	--->	Responsive	_Parm07	0.90452	0.08886	10.17972
Service	--->	Helpful	_Parm08	0.64969	0.07683	8.45574
Service	--->	Delivery	_Parm09	0.64473	0.09021	7.14677
Product	--->	Pricing	_Parm10	0.63452	0.07916	8.01600
Product	--->	Availability	_Parm11	0.76737	0.08265	9.28516
Product	--->	Quality	_Parm12	0.79716	0.08922	8.93470

Model 1. Variance Parameters

Variance Type	Variable	Parameter	Estimate	Standard Error	t Value
Error	Courtesy	_Parm13	1.98374	0.13169	15.06379
	Responsive	_Parm14	2.02152	0.16159	12.51005
	Helpful	_Parm15	1.96535	0.12263	16.02727
	Delivery	_Parm16	2.97542	0.17049	17.45184
	Pricing	_Parm17	1.93952	0.12326	15.73583
	Availability	_Parm18	1.63156	0.13067	12.48646
	Quality	_Parm19	2.08849	0.15329	13.62464
	Spend02	_Parm20	13.47066	0.71842	18.75051
	Spend03	_Parm21	13.02883	0.68682	18.96966
Exogenous	Service		1.00000		
	Product		1.00000		

Model 1. Covariances Among Exogenous Variables

Var1	Var2	Parameter	Estimate	Standard Error	t Value
Service	Product	_Parm22	0.33725	0.07061	4.77599

All parameter estimates but one are statistically significant at $\alpha = 0.05$. The parameter _Parm01, which represents the path coefficient from Service to Spend02, has a t value of 1.76. This is only marginally significant. Although all these results bear the title of Model 1, these estimates are the same for Model 2, of which the corresponding results are not shown here.

The mean and intercept parameters for the two models (regions) are shown in Output 26.27.9.

Output 26.27.9 Estimates of Means and Intercepts

Model 1. Means and Intercepts

Type	Variable	Parameter	Estimate	Standard Error	t Value
Intercept	Spend02	G1_InterSpend02	183.50000	0.19585	936.95628
	Spend03	G1_InterSpend03	285.49480	6.78127	42.10048
	Courtesy	G1_intercept01	4.31200	0.08157	52.86519
	Responsive	G1_intercept02	4.72400	0.08679	54.43096
	Helpful	G1_intercept03	3.92100	0.07958	49.27201
	Delivery	G1_intercept04	4.35700	0.09484	45.93968
	Pricing	G1_intercept05	6.14400	0.07882	77.94992
	Availability	G1_intercept06	4.99400	0.07674	65.07315
	Quality	G1_intercept07	5.97100	0.08500	70.24543

Model 2. Means and Intercepts

Type	Variable	Parameter	Estimate	Standard Error	t Value
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 26.27.10 can confirm this.

Output 26.27.10 shows two simultaneous tests, as requested in the original statements.

Output 26.27.10 Simultaneous Tests

Simultaneous Tests

Simultaneous Test	Parametric Function	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 following hypotheses:

$$H_0 : \text{G2\_InterSpend02 - G1\_InterSpend02} = 0$$
$$H_0 : \text{G2\_InterSpend03 - G1\_InterSpend03} = 0$$

The exceedingly large chi-square value $10,460$ suggests the composite null hypothesis is false. Individual tests for these hypotheses suggest that each of these hypotheses should be rejected. The chi-square values for individual tests are $10,227$ and 185.84, respectively.

Similarly, the simultaneous and individual tests of the intercepts in the measurement model suggest that the two models (groups) differ significantly in the means of the measured variables. Region 2 has significantly higher means in variables Helpful, Delivery, and Pricing, but significantly lower means in variables Courtesy, Responsive, Availability, and Quality.

Now you are ready to answer the main research questions. The overall customer service (Service) does affect the future purchase (Spend03), but not the current purchase (Spend02), because the corresponding path coefficient (_Parm01) is only marginally significant. Perhaps this is an artifact because the rating was done after the purchases in 2002. That is, purchases in 2002 had been done before the impression about customer service was fully formed. However, this argument cannot explain why overall customer service (Service) also shows a strong and significant relationship with purchases in 2002 (Spend02). Nonetheless, customer service and product quality do affect the future purchases (Spend03) in an expected way, even after partialling out the effect of the previous purchase amount (Spend02). Apart from the mean differences of the variables, the common measurement and prediction (or structural) models fit the two regions very well.

Because the current model fits well and most parts of fitting meet your expectations, you might accept this model without looking for further improvement. Nonetheless, for illustration purposes, it would be useful to consider the LM test results. In Output 26.27.11, ranked LM statistics for the path coefficients in Model 1 and Model 2 are shown.

Output 26.27.11 LM Tests for Path Coefficients

Model 1. Rank Order of the 10 Largest LM Stat for Path Relations

To	From	LM Stat	Pr > ChiSq	Parm 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

Model 2. Rank Order of the 10 Largest LM Stat for Path Relations

To	From	LM Stat	Pr > ChiSq	Parm 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 decrease) are shown first in the tables. For example, the first path coefficient that is suggested to be freed in Model 1 is the Service <--- Courtesy path. The associated p-value is 0.0008 and the estimated change of parameter value is -0.171. The second path coefficient is for the Service <--- Helpful path, but it is not significant at the 0.05 level. So, is it good to add the Service <--- Courtesy path to Model 1, based on the LM test results? The answer is that it depends on your application and the theoretical and practical implications. For example, the Service -->- Courtesy path, which is a part of the measurement model, is already specified in Model 1. Even though the LM test statistic shows a significant decrease of model fit chi-square, adding the Service <--- Courtesy path might destroy the measurement model and lead to problematic interpretations. In this case, it is wise not to add the Service <--- Courtesy path, which is suggested by the LM test results.

LM tests for the path coefficients in Model 2 are shown at the bottom of Output 26.27.11. 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 26.27.12.

Output 26.27.12 LM Tests for Error Covariances

Model 1. Rank Order of the 10 Largest LM Stat for Error Variances and Covariances

Error of	Error of	LM Stat	Pr > ChiSq	Parm 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 of	Error of	LM Stat	Pr > ChiSq	Parm 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 not shown here simply for conserving space.

PROC CALIS ranks and outputs the LM test results for some default subsets of parameters. You have seen the subsets for path coefficients and correlated errors in the two previous outputs. Some other LM test results are not shown. With this kind of default LM output, there could be a huge amount of modification indices to look at. Fortunately, you can limit the LM test results to any subsets of potential parameters that you might be interested in. With your substantive knowledge, you can define such meaningful subsets of potential parameters by using the LMTESTS statement. The LM test indices and rankings are then done for each predefined subset of potential parameters. With these customized LM results, you can limit your attention to consider only those meaningful parameters to be added. See the LMTESTS statement on page 1101 for details.

The next group of LM tests is for releasing implicit equality constraints in your model, as shown in Output 26.27.13.

Output 26.27.13 LM Tests for Equality Constraints

Lagrange Multiplier Statistics for Releasing Equality Constraints

	Released Parameter						Changes	
Parm	Model	Type	Var1	Var2	LM Stat	Pr > ChiSq	Original Parm	Released Parm
_Parm01	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
_Parm02	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
_Parm03	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
_Parm04	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
_Parm05	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
_Parm06	1	DV_IV	Courtesy	Service	19.17225	<.0001	0.2851	-0.3191
	2	DV_IV	Courtesy	Service	19.17225	<.0001	-0.3191	0.2851
_Parm07	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
_Parm08	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
_Parm09	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
_Parm10	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
_Parm11	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
_Parm12	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
_Parm13	1	COVERR	Courtesy	Courtesy	45.24725	<.0001	0.7204	-0.8064
	2	COVERR	Courtesy	Courtesy	45.24725	<.0001	-0.8064	0.7204
_Parm14	1	COVERR	Responsive	Responsive	1.73499	0.1878	-0.1555	0.1740
	2	COVERR	Responsive	Responsive	1.73499	0.1878	0.1740	-0.1555
_Parm15	1	COVERR	Helpful	Helpful	11.13266	0.0008	-0.3448	0.3860
	2	COVERR	Helpful	Helpful	11.13266	0.0008	0.3860	-0.3448
_Parm16	1	COVERR	Delivery	Delivery	4.99097	0.0255	-0.3364	0.3766
	2	COVERR	Delivery	Delivery	4.99097	0.0255	0.3766	-0.3364
_Parm17	1	COVERR	Pricing	Pricing	2.86428	0.0906	0.1729	-0.1936
	2	COVERR	Pricing	Pricing	2.86428	0.0906	-0.1936	0.1729
_Parm18	1	COVERR	Availability	Availability	2.53147	0.1116	-0.1494	0.1672
	2	COVERR	Availability	Availability	2.53147	0.1116	0.1672	-0.1494
_Parm19	1	COVERR	Quality	Quality	0.07328	0.7866	-0.0315	0.0352
	2	COVERR	Quality	Quality	0.07328	0.7866	0.0352	-0.0315
_Parm20	1	COVERR	Spend02	Spend02	0.00214	0.9631	0.0304	-0.0340
	2	COVERR	Spend02	Spend02	0.00214	0.9631	-0.0340	0.0304
_Parm21	1	COVERR	Spend03	Spend03	0.0001773	0.9894	-0.00842	0.00946
	2	COVERR	Spend03	Spend03	0.0001773	0.9894	0.00946	-0.00842
_Parm22	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 26.27.13 shows you which parameter can be unconstrained so that your overall model fit might improve. For example, if you unconstrain the first parameter _Parm01, which is for the path effect of Spend02 <--- Service, for the two models, the expected chi-square decrease (LM Stat) is about 0.0158, which is not significant ($p = .9001$). The associated parameter changes are small too. However, if you consider unconstraining parameter _Parm06, which is for the path effect of Courtesy <--- Service, the expected decrease of chi-square is 19.22 ($p < 0.0001$). 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 _Parm06 as the parameter for the Courtesy <--- Service path in Model 2) and fit the model again, the new estimate of _Parm06 is 0.2852 greater than the previous _Parm06 estimate. The estimate of "new" is 0.3196 less than the previous _Parm06 estimate. The second row for the _Parm06 parameter shows similar but reflected results. It is for renaming the parameter location in Model 2. For this example each equality constraint has exactly two locations, one for Model 1 and one for Model 2. That is the reason why you always observe reflected results for freeing the locations successively. Reflected results are not the case if you have equality constraints with more than two parameter locations.

Another example of a large expected improvement of model fit is the result of freeing the constrained variances of Courtesy among the two models. The corresponding row to look at is the row with parameter _Parm13, where the parameter type is labeled "COVERR" and the values for Var1 and Var2 are both "Courtesy." The LM statistic is 45.255, which is a significant chi-square decrease if you free either parameter location. If you choose to rename the error variance for Courtesy in Model 1, the new _Parm13 estimate is 0.8052 smaller than the original _Parm13 estimate. The new estimate of the error variance for Courtesy in Model 2 is 0.7211 greater than the previous _Parm13 estimate. Finally, the constrained parameter _Parm15, which is the error variance parameter for Helpful in both models, is also a potential constraint that can be released with a significant model fit improvement.

In addition to the LM statistics for suggesting ways to improve model fit, PROC CALIS also computes the Wald tests to show which parameters can be constrained to zero without jeopardizing the model fit significantly. The Wald test results are shown in Output 26.27.14.

Output 26.27.14 Wald Tests

Stepwise Multivariate Wald Test

	------Cumulative Statistics-----			--Univariate Increment--	
Parm	Chi-Square	DF	Pr > ChiSq	Chi-Square	Pr > ChiSq
_Parm01	3.09039	1	0.0788	3.09039	0.0788

In Output 26.27.14, you see that _Parm01, which is for the path effect of Spend02 <--- Service, is suggested to be a fixed zero parameter (eliminated from the model) by the Wald test. Fixing this parameter to zero (or dropping the Spend02 <--- Service path from the model) is expected to increase the model fit chi-square by 3.085 (p=.079), which is only marginally significant at $\alpha = 0.05$.

As is the case for the LM test statistics, you should not automatically adhere to the suggestions by the Wald statistics. Substantive and theoretical considerations should always be considered when determining whether a parameter should be added or dropped.

Example 26.28: Fitting the RAM and EQS Models by the COSAN Modeling Language

The COSAN modeling language in PROC CALIS enables you to specify the direct or implied mean and covariance structures for the data in terms of matrix formulas. It is a very general modeling language, and all other modeling languages in PROC CALIS are special cases of the COSAN modeling language. This example shows how you can apply the COSAN modeling language to situations where you might usually use the "easier" modeling languages. Therefore, the purpose of this example is not to recommend the use of the COSAN modeling specification to the specific application. Rather, through its connections with other more well-known model types, this example intends to help you understand the basics of the COSAN modeling language.

In Example 26.16, you fit a path model to the Wheaton data (Wheaton et al. 1977) by using the PATH modeling language. The mathematical basis of the PATH modeling language is the RAM model. In Example 26.22, you use the RAM and LINEQS statements to specify the same path model. In all these different types of specifications, you specify the functional relationships of the variables and the variance and covariance parameters in the model. PROC CALIS then generates the implied covariance structures for analysis internally. The COSAN modeling language is quite different. In the COSAN statement, you specify the covariance structures directly as a matrix formula. This example shows how you can do that in two different ways. One specification emulates the RAM model (McDonald 1978, 1980) covariance structures and the other emulates the EQS model (Bentler 1995) covariance structures.

Emulating the RAM model by the COSAN Modeling Language

In the RAM model, you specify all information regarding the path effects or coefficients (that is, single-headed arrows in the path diagram) in the so-called **A** (_A_) matrix. You specify all the information regarding the variances and covariances (that is, the double-headed arrows in the path diagram) in the **P** (_P_) matrix. See the section "The RAM Model" on page 1229 for more details about the mathematical model for RAM. Once you define these two matrices, the implied covariance structures for the observed variables are derived by the formula

$$\mathbf{\Sigma} = \mathbf{J} * (\mathbf{I} - \mathbf{A})^{-1} * \mathbf{P} * (\mathbf{I} - \mathbf{A})^{-1\prime} * \mathbf{J}'$$

where $\mathbf{I}$ is an identity matrix and $\mathbf{J}$ is a selection matrix that contains 0 or 1 as its elements for selecting the covariance structures elements for the observed variables.

For example, in the RAM model specification in Example 26.22, you essentially use the following RAM model specification:

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

In the RAM statement, you specify all the parameters in the _A_ and _P_ matrices, and PROC CALIS generates the corresponding covariance structures for analysis. However, with the COSAN modeling language, in addition to the parameter in the model matrices, you need to supply the matrix formula for the covariance structures, as shown in the preceding formula for $\boldsymbol{\Sigma}$.

Before discussing how you can specify the COSAN model that corresponds to this RAM model specification, it is useful to look at the initial model matrices that are generated by the preceding RAM model specification. To do this, you use the PRIMAT option in the PROC CALIS statement.

Output 26.28.1 and Output 26.28.2 show the initial _A_ and _P_ matrices, respectively, for the RAM model.

Output 26.28.1 Initial _A_ Matrix of the RAM Model

Initial RAM _A_ Matrix

	Anomie67	Powerless67	Anomie71	Powerless71	Education
Anomie67	0	0	0	0	0
Powerless67	0	0	0	0	0
Anomie71	0	0	0	0	0
Powerless71	0	0	0	0	0
Education	0	0	0	0	0
SEI	0	0	0	0	0
Alien67	0	0	0	0	0
Alien71	0	0	0	0	0
SES	0	0	0	0	0

Output 26.28.1 *continued*

Initial RAM _A_ Matrix

	SEI	Alien67	Alien71	SES
Anomie67	0	1.0000	0	0
Powerless67	0	0.8330	0	0
Anomie71	0	0	1.0000	0
Powerless71	0	0	0.8330	0
Education	0	0	0	1.0000
SEI	0	0	0	. [lambda]
Alien67	0	0	0	. [gamma1]
Alien71	0	. [beta]	0	. [gamma2]
SES	0	0	0	0

Output 26.28.2 Initial _P_ Matrix of the RAM Model

Initial RAM _P_ Matrix

	Anomie67	Powerless67	Anomie71	Powerless71	Education
Anomie67	. [theta1]	0	. [theta5]	0	0
Powerless67	0	. [theta2]	0	. [theta5]	0
Anomie71	. [theta5]	0	. [theta1]	0	0
Powerless71	0	. [theta5]	0	. [theta2]	0
Education	0	0	0	0	. [theta3]
SEI	0	0	0	0	0
Alien67	0	0	0	0	0
Alien71	0	0	0	0	0
SES	0	0	0	0	0

Output 26.28.2 *continued*

Initial RAM _P_ Matrix

	SEI	Alien67	Alien71	SES
Anomie67	0	0	0	0
Powerless67	0	0	0	0
Anomie71	0	0	0	0
Powerless71	0	0	0	0
Education	0	0	0	0
SEI	. [theta4]	0	0	0
Alien67	0	. [psi1]	0	0
Alien71	0	0	. [psi2]	0
SES	0	0	0	. [phi]

Essentially, to specify the same model by the COSAN modeling language, you need to provide the same information in these two initial model matrices and the covariance structure formula for Σ in the COSAN model specification, which is shown in the following statements:

```
proc calis data=Wheaton nobs=932 nose;
   cosan
      var= Anomie67 Powerless67 Anomie71 Powerless71 Education SEI,
      J(9, IDE) * A(9, GEN, IMI) * P(9, SYM);
   matrix  A
      [1 2 8   , 7] = 1.0  0.833  beta,
      [3 4     , 8] = 1.0  0.833 ,
      [5 6 7 8 , 9] = 1.   lambda  gamma1  gamma2;
   matrix P
      [1,1] = theta1-theta2 theta1-theta4 ,
      [7,7] = psi1 psi2 phi,
      [3,1] = theta5 ,
      [4,2] = theta5 ;
   vnames
      J = [Anomie67 Powerless67 Anomie71 Powerless71
           Education SEI Alien67 Alien71 SES],
      A = J,
      P = A;
run;
```

In the PROC CALIS statement, you provide the data set in the DATA= option and the number of observations in the NOBS= option. You use the NOSE option to turn off the computation of the standard error estimates.

In the VAR= option of the COSAN statement, you provide the list of observed variables for the analysis. You do not specify the latent variables in the VAR= option in the COSAN statement as you do in the VAR= option in the RAM statement. Then, you specify the formula for the covariance structures for the set of variables in the VAR= list. Because the covariance structure formula is symmetric, you only need to specify "half" of it. That is, the specification `J(9,IDE)*A(9,GEN,IMI)*P(9,SYM)` in the COSAN statement automatically expands to

$$\mathbf{J} * (\mathbf{I} - \mathbf{A})^{-1} * \mathbf{P} * (\mathbf{I} - \mathbf{A})^{-1\prime} * \mathbf{J}'$$

which is the required covariance structures. The arguments in the matrices represent the number of columns, the matrix type, and the transformation type (optional), respectively. For example, the notation `A(9, GEN, IMI)` means that matrix **A** has nine columns and it is a general (GEN) rectangular or square matrix. You do not specify the number of rows for matrix **A** explicitly, but PROC CALIS can deduce that because matrix **A** follows matrix **J** in the multiplication. To make matrix multiplication conformable, the number of rows for matrix **A** must be the same as the number of columns for matrix **J**, which is nine. The IMI notation means the identity-minus-inverse transformation, which results in putting $(\mathbf{I} - \mathbf{A})^{-1}$ in the expression. Matrix **P** in the covariance structure formula is a 9×9 symmetric matrix. It does not have any transformation in the formula. Matrix **J** in the covariance structure formula is a so-called generalized identity matrix (IDE), which has six rows and nine columns. Basically, you use this matrix to select the observed variables in the covariance structure formula. The exact form of this matrix will become clear when the PROC CALIS output is shown.

Next, you use two MATRIX statements to specify the parameters in the model matrices **A** and **P**, for RAM model matrices _A_ and _P_, respectively. For example, in the first entry of the MATRIX statement for the **A** matrix, you specify the elements `[1,7]`, `[2,7]`, and `[8,7]` by 1.0, 0.833, and beta, respectively. The first two elements are fixed constants, while the last one is a free parameter named beta. Similarly, you specify all the fixed or free parameters in matrix **A**, which reflects the same pattern you specify for the _A_ matrix of the RAM model, as shown in Output 26.28.1.

For the **P** matrix, you specify the parameters in the same fashion. Because **P** is defined as a symmetric matrix, you need to specify only the lower triangular elements. In the first entry of the MATRIX statement for the **P** matrix, you specify the `[1,1]` element, but the trailing parameter list has six parameters. The `[1,1]` notation here is interpreted as the starting location of the matrix. It proceeds to `[2,2]`, `[3,3]`, `[4,4]` and so on. The length of the trailing parameter list determines the number of elements being specified. Therefore, the last parameter in this entry is for $\mathbf{P}[6,6]$, which is a free parameter theta4. Similarly, you define all other parameters in the **P** matrix, which reflects the same pattern you specify for the _P_ matrix of the RAM model, as shown in Output 26.28.2.

In the VNAMES statement, you can specify the column variable names for the model matrices. You provide a set of nine variable names for the column of matrix **J** in the pairs of brackets. The first six names are those of the observed variables in the COSAN model, while the last six names are for latent factors. How about the row variable names for matrix **J**? Because matrix **J** is the first matrix in the covariance structure formula, its row names are automatically the same as the names of the observed variables in the VAR= list of the COSAN statement. Next, you specify the column variable names of matrix **A**. You equate that to matrix **J**, meaning that the column variable names in matrix **A** are the same those for matrix **J**. How about the row variable names for matrix **A**? Because matrix **A** follows matrix **J** in the covariance structure formula, its row names are automatically same as the column names for matrix **J**. Lastly, you define that the column names for matrix **P** are the same as those for matrix **A**.

Notice that column names serve only as labels. PROC CALIS does not know the identities of the row and column variables. For example, the first column of matrix **A** is Anomie67, which is also a name for an observed variable in the COSAN model. Keeping other specifications intact, you could name this column by any other name without affecting the model estimation. It is recommended that you use sensible names that help you remember the identities of the row and column variables, such as this example shows.

Output 26.28.3 shows the modeling information and the observed variables in the COSAN model. PROC CALIS analyzed the covariance structures of the six observed variables listed in Output 26.28.3.

Output 26.28.3 Modeling Information of the COSAN Model for the Wheaton Data: RAM Emulation

```
                         Modeling Information

                   Data Set              WORK.WHEATON
                   N Obs                 932
                   Model Type            COSAN
                   Analysis              Covariances

               Observed Variables (N = 6) in the Model

     Anomie67  Powerless67  Anomie71  Powerless71  Education  SEI
```

Output 26.28.4 shows the covariance structures and some properties of the model matrices. The covariance structure formula for Sigma is defined as required. You can also check the matrix properties in this output to see if they are what you intend them to be.

Output 26.28.4 The Covariance Structures and Model Matrices of the COSAN Model for the Wheaton Data: RAM Emulation

```
                        COSAN Model Structures

              Sigma =     J*inv(_I_-A)*P*(inv(_I_-A))`*J`

                      Summary of Model Matrices

           Matrix      N Row      N Col     Matrix Type

           A               9          9     GEN: Square
           J               6          9     IDE: (I || 0)
           P               9          9     SYM: Symmetric
```

Output 26.28.4 shows that $\mathbf{J}$ is a 6×9 "identity" matrix $(\mathbf{I}||\mathbf{0})$. Essentially, $\mathbf{J}$ is a selection matrix that contains either 0 or 1 as its elements. The role of matrix $\mathbf{J}$ in the covariance structure formula is to extract first six rows and columns in the inner covariance structures $(\mathbf{I} - \mathbf{A})^{-1} * \mathbf{P} * (\mathbf{I} - \mathbf{A})^{-1\prime}$ (which is 9×9) to form the covariance structures only for the observed variables (which is 6×6). But how can this identity matrix have more columns (9) than rows (6)? In common mathematical notation, an identity matrix must always be a square matrix. However, for convenience in notation, PROC CALIS generalizes it to the IDE type. An IDE matrix that has the same numbers of columns and rows is a square identity matrix. If an IDE matrix has more columns than rows, it denotes an identity matrix concatenated (to the right) by a null matrix (that is, the $(\mathbf{I}||\mathbf{0})$ notation). If an IDE matrix has more rows than columns, it denotes an identity matrix appended (to the bottom) by a null matrix (that is, the $(\mathbf{I}//\mathbf{0})$ notation). The generalized definition for the IDE matrix offers an efficient way to define selection matrix, such as the $\mathbf{J}$ matrix shown in this example.

Output 26.28.5 shows the model fit chi-square of the COSAN model. This is the same model fit as in Output 26.16.6 of Example 26.16, as expected.

Output 26.28.5 Model Fit of the COSAN Model for the Wheaton Data: RAM Emulation

Fit Summary

Chi-Square	13.4851
Chi-Square DF	9
Pr > Chi-Square	0.1419

Output 26.28.6 shows the estimates in the **A** matrix.

Output 26.28.6 Estimate of the **A** Matrix by the COSAN Model Specification

Model Matrix A
(9 x 9 General Square Matrix)

	Anomie67	Powerless67	Anomie71	Powerless71	Education
Anomie67	0	0	0	0	0
Powerless67	0	0	0	0	0
Anomie71	0	0	0	0	0
Powerless71	0	0	0	0	0
Education	0	0	0	0	0
SEI	0	0	0	0	0
Alien67	0	0	0	0	0
Alien71	0	0	0	0	0
SES	0	0	0	0	0

Output 26.28.6 *continued*

Model Matrix A
(9 x 9 General Square Matrix)

	SEI	Alien67	Alien71	SES
Anomie67	0	1.0000	0	0
Powerless67	0	0.8330	0	0
Anomie71	0	0	1.0000	0
Powerless71	0	0	0.8330	0
Education	0	0	0	1.0000
SEI	0	0	0	5.3689 [lambda]
Alien67	0	0	0	-0.6299 [gamma1]
Alien71	0	0.5931 [beta]	0	-0.2409 [gamma2]
SES	0	0	0	0

The estimates in Output 26.28.6 from the COSAN model specification are essentially the same as those from the RAM model specification, as shown in the matrix form in Output 26.28.7.

Output 26.28.7 Estimate of the $\mathbf{A}$ Matrix by the RAM Model Specification

RAM _A_ Matrix

	Anomie67	Powerless67	Anomie71	Powerless71	Education
Anomie67	0	0	0	0	0
Powerless67	0	0	0	0	0
Anomie71	0	0	0	0	0
Powerless71	0	0	0	0	0
Education	0	0	0	0	0
SEI	0	0	0	0	0
Alien67	0	0	0	0	0
Alien71	0	0	0	0	0
SES	0	0	0	0	0

Output 26.28.7 *continued*

RAM _A_ Matrix

	SEI	Alien67	Alien71	SES
Anomie67	0	1.0000	0	0
Powerless67	0	0.8330	0	0
Anomie71	0	0	1.0000	0
Powerless71	0	0	0.8330	0
Education	0	0	0	1.0000
SEI	0	0	0	5.3688 [lambda]
Alien67	0	0	0	-0.6299 [gamma1]
Alien71	0	0.5931 [beta]	0	-0.2409 [gamma2]
SES	0	0	0	0

Output 26.28.8 shows the estimates in the **P** matrix.

Output 26.28.8 Estimate of the **P** Matrix by the COSAN Model Specification

Model Matrix P
(9 x 9 Symmetric Matrix)

	Anomie67	Powerless67	Anomie71	Powerless71	Education
Anomie67	3.6078 [theta1]	0	0.9058 [theta5]	0	0
Powerless67	0	3.5950 [theta2]	0	0.9058 [theta5]	0
Anomie71	0.9058 [theta5]	0	3.6078 [theta1]	0	0
Powerless71	0	0.9058 [theta5]	0	3.5950 [theta2]	0
Education	0	0	0	0	2.9938 [theta3]
SEI	0	0	0	0	0
Alien67	0	0	0	0	0
Alien71	0	0	0	0	0
SES	0	0	0	0	0

Output 26.28.8 *continued*

Model Matrix P
(9 x 9 Symmetric Matrix)

	SEI	Alien67	Alien71	SES
Anomie67	0	0	0	0
Powerless67	0	0	0	0
Anomie71	0	0	0	0
Powerless71	0	0	0	0
Education	0	0	0	0
SEI	259.5738 [theta4]	0	0	0
Alien67	0	5.6705 [psi1]	0	0
Alien71	0	0	4.5148 [psi2]	0
SES	0	0	0	6.6162 [phi]

Again, aside from very minor numerical differences, the estimates shown in Output 26.28.8 from the COSAN model specification are essentially the same as those from the RAM model specification, as shown in the matrix form in Output 26.28.9.

Output 26.28.9 Estimate of the $\mathbf{P}$ Matrix by the RAM Model Specification

RAM _P_ Matrix

	Anomie67	Powerless67	Anomie71	Powerless71	Education
Anomie67	3.6080 [theta1]	0	0.9058 [theta5]	0	0
Powerless67	0	3.5949 [theta2]	0	0.9058 [theta5]	0
Anomie71	0.9058 [theta5]	0	3.6080 [theta1]	0	0
Powerless71	0	0.9058 [theta5]	0	3.5949 [theta2]	0
Education	0	0	0	0	2.9937 [theta3]
SEI	0	0	0	0	0
Alien67	0	0	0	0	0
Alien71	0	0	0	0	0
SES	0	0	0	0	0

Output 26.28.9 *continued*

RAM _P_ Matrix

	SEI	Alien67	Alien71	SES
Anomie67	0	0	0	0
Powerless67	0	0	0	0
Anomie71	0	0	0	0
Powerless71	0	0	0	0
Education	0	0	0	0
SEI	259.5764 [theta4]	0	0	0
Alien67	0	5.6705 [psi1]	0	0
Alien71	0	0	4.5148 [psi2]	0
SES	0	0	0	6.6163 [phi]

Emulating the EQS model by the COSAN Modeling Language

The LINEQS modeling language in PROC CALIS enables you to specify the functional relationships among variables by using the equation input, much the same way that you can do with the EQS software (Bentler 1995). The covariance structure formula for the observed variables in the EQS model is

$$\mathbf{\Sigma} = \mathbf{J} * (\mathbf{I} - \mathbf{Beta})^{-1} * \mathbf{Gamma} * \mathbf{Phi} * \mathbf{Gamma}' * (\mathbf{I} - \mathbf{Beta})^{-1\prime} * \mathbf{J}'$$

where **I** is an identity matrix, **J** is a selection matrix that contains 0 or 1 as its elements for selecting the covariance structures elements for the observed variables, **Beta** is a square matrix for specifying relationships among the endogenous variables, Gamma is a matrix for specifying relationships between the endogenous variables and the exogenous variables, and **Phi** is a matrix for specifying the variances and covariances of the exogenous variables. Notice that in the EQS model, error or disturbance variables are counted as exogenous variables in the model.

In Example 26.22, you use the following LINEQS specification for the Wheaton data:

```
proc calis nobs=932 data=Wheaton primat nose;
   lineqs
      Anomie67     = 1.0    * f_Alien67 + e1,
      Powerless67  = 0.833  * f_Alien67 + e2,
      Anomie71     = 1.0    * f_Alien71 + e3,
      Powerless71  = 0.833  * f_Alien71 + e4,
      Education    = 1.0    * f_SES     + e5,
      SEI          = lambda * f_SES     + e6,
      f_Alien67    = gamma1 * f_SES     + d1,
      f_Alien71    = gamma2 * f_SES     + beta * f_Alien67 + d2;
   variance
      E1           = theta1,
      E2           = theta2,
      E3           = theta1,
      E4           = theta2,
      E5           = theta3,
      E6           = theta4,
      D1           = psi1,
      D2           = psi2,
      f_SES        = phi;
   cov
      E1  E3       = theta5,
      E2  E4       = theta5;
run;
```

In the LINEQS statement, you specify all the functional relationships among variables. In the VARIANCE and COV statements, you specify all the variance and covariance parameters in the model. None of the parameters is specified as a matrix element in the LINEQS model. The default output by PROC CALIS does not print the EQS model matrices. To print these model matrices, you use the PRIMAT option in the PROC CALIS statement. Output 26.28.10, Output 26.28.11, and Output 26.28.12 show the initial specification of these model matrices:

Output 26.28.10 The Initial _EQSBETA_ Matrix by the LINEQS Model Specification

Initial _EQSBETA_ Matrix

	Anomie67	Anomie71	Education	Powerless67
Anomie67	0	0	0	0
Anomie71	0	0	0	0
Education	0	0	0	0
Powerless67	0	0	0	0
Powerless71	0	0	0	0
SEI	0	0	0	0
f_Alien67	0	0	0	0
f_Alien71	0	0	0	0

Initial _EQSBETA_ Matrix

	Powerless71	SEI	f_Alien67	f_Alien71
Anomie67	0	0	1.0000	0
Anomie71	0	0	0	1.0000
Education	0	0	0	0
Powerless67	0	0	0.8330	0
Powerless71	0	0	0	0.8330
SEI	0	0	0	0
f_Alien67	0	0	0	0
f_Alien71	0	0	. [beta]	0

Output 26.28.11 The Initial _EQSGAMMA_ Matrix by the LINEQS Model Specification

Initial _EQSGAMMA_ Matrix

	f_SES	e1	e3	e5	e2
Anomie67	0	1.0000	0	0	0
Anomie71	0	0	1.0000	0	0
Education	1.0000	0	0	1.0000	0
Powerless67	0	0	0	0	1.0000
Powerless71	0	0	0	0	0
SEI	. [lambda]	0	0	0	0
f_Alien67	. [gamma1]	0	0	0	0
f_Alien71	. [gamma2]	0	0	0	0

Initial _EQSGAMMA_ Matrix

	e4	e6	d1	d2
Anomie67	0	0	0	0
Anomie71	0	0	0	0
Education	0	0	0	0
Powerless67	0	0	0	0
Powerless71	1.0000	0	0	0
SEI	0	1.0000	0	0
f_Alien67	0	0	1.0000	0
f_Alien71	0	0	0	1.0000

Output 26.28.12 The Initial _EQSPHI_ Matrix by the LINEQS Model Specification

Initial _EQSPHI_ Matrix

	f_SES	e1	e3	e5	e2
f_SES	. [phi]	0	0	0	0
e1	0	. [theta1]	. [theta5]	0	0
e3	0	. [theta5]	. [theta1]	0	0
e5	0	0	0	. [theta3]	0
e2	0	0	0	0	. [theta2]
e4	0	0	0	0	. [theta5]
e6	0	0	0	0	0
d1	0	0	0	0	0
d2	0	0	0	0	0

Output 26.28.12 *continued*

Initial _EQSPHI_ Matrix

	e4	e6	d1	d2
f_SES	0	0	0	0
e1	0	0	0	0
e3	0	0	0	0
e5	0	0	0	0
e2	. [theta5]	0	0	0
e4	. [theta2]	0	0	0
e6	0	. [theta4]	0	0
d1	0	0	. [psi1]	0
d2	0	0	0	. [psi2]

In the COSAN modeling language, you need to provide the three initial model matrices and the covariance structure formula for Σ, which is shown in the following statements:

```
proc calis cov data=Wheaton nobs=932 nose;
   cosan
      var = Anomie67 Anomie71 Education Powerless67 Powerless71 SEI,
      J(8, IDE) * Beta(8, GEN, IMI) * Gamma(9, GEN) * Phi(9, SYM);
   matrix Beta
           [1 4 8   , 7] = 1.0  0.833  beta,
           [2 5     , 8] = 1.0  0.833 ;
   matrix Gamma
           [3 6 7 8 , 1] = 1.0  lambda gamma1 gamma2,
           [1,2]         = 8 *  1.0;
   matrix Phi
           [1,1] = phi 2*theta1 theta3 2*theta2 theta4 psi1 psi2,
           [3,2] = theta5 ,
           [6,5] = theta5 ;
   vnames J     = [Anomie67 Anomie71 Education Powerless67 Powerless71 SEI
                    f_Alien67 f_Alien71],
          Beta  = J,
          Gamma = [f_SES e1 e3 e5 e2 e4 e6 d1 d2],
          Phi   = Gamma;
run;
```

In the PROC CALIS statement, you provide the data set in the DATA= option and the number of observations in the NOBS= option. You use the NOSE option to turn off the computation of the standard error estimates.

In the VAR= option of the COSAN statement, you provide the list of observed variables for the analysis. You arrange the observed variables in such a way that they are in the same order as in Output 26.28.10, Output 26.28.10, and Output 26.28.12. This is useful for comparing the results from the LINEQS and COSAN model specifications. After the specification of the observed variables, you specify the covariance structure model in the COSAN statement. Again, you only need to specify "half" of it. That is, the specification `J(8,IDE)*Beta(8,GEN,IMI)*Gamma(9,GEN)*Phi(9,SYM)` in the COSAN statement automatically expands to

$$\mathbf{\Sigma} = \mathbf{J} * (\mathbf{I} - \mathbf{Beta})^{-1} * \mathbf{Gamma} * \mathbf{Phi} * \mathbf{Gamma}' * (\mathbf{I} - \mathbf{Beta})^{-1\prime} * \mathbf{J}'$$

which is the required covariance structures. Matrix properties and transformation types are defined in the arguments for the matrices.

Next, you use three matrix statements to specify the parameters in the matrix elements. The specifications here reflect exactly the initial specifications for the LINEQS model matrices as shown in Output 26.28.10, Output 26.28.10, and Output 26.28.12.

In the VNAMES statement, you specify the column variable names for the matrices. The column variable names of the **J** matrix include all the observed variable names and the names of the intended endogenous latent factors f_Alien67 and f_Alien71. The column variable names for the **Beta** matrix are the same as those for matrix **J**. The column variables for the **Gamma** matrix include the intended latent factor f_SES and error variable names e1–e6 and d1–d2, which are arranged in such a way that they match the order of the error variables in the LINEQS output shown in Output 26.28.12.

Output 26.28.13 shows the covariance structures and some properties of the model matrices. The covariance structure formula for **Sigma** is defined as required. You can also check the matrix properties in this output to see if they are what you intend them to be.

Output 26.28.13 The Covariance Structures and Model Matrices of the COSAN Model for the Wheaton Data: EQS Emulation

```
                              COSAN Model Structures

       Sigma =    J*inv(_I_-Beta)*Gamma*Phi*Gamma`*(inv(_I_-Beta))`*J`

                             Summary of Model Matrices

               Matrix      N Row      N Col     Matrix Type

               Beta            8          8     GEN: Square
               Gamma           8          9     GEN: Rectangular
               J               6          8     IDE: (I || 0)
               Phi             9          9     SYM: Symmetric
```

Output 26.28.14 shows the model fit chi-square of the current COSAN model. As expected, this is the same model fit as in Output 26.16.6 of Example 26.16 and in Output 26.28.5.

Output 26.28.14 Model Fit of the COSAN Model for the Wheaton Data: EQS Emulation

Fit Summary

Chi-Square	13.4851
Chi-Square DF	9
Pr > Chi-Square	0.1419

Output 26.28.15 shows the estimates of the **Beta** matrix by the COSAN model specification. These estimates are essentially the same as the estimates of the _EQSBETA_ matrix obtained from the LINEQS model specification, as shown in Output 26.28.16.

Output 26.28.15 Estimate of the **Beta** Matrix by the COSAN Model Specification

Model Matrix Beta
(8 x 8 General Square Matrix)

	Anomie67	Anomie71	Education	Powerless67
Anomie67	0	0	0	0
Anomie71	0	0	0	0
Education	0	0	0	0
Powerless67	0	0	0	0
Powerless71	0	0	0	0
SEI	0	0	0	0
f_Alien67	0	0	0	0
f_Alien71	0	0	0	0

Model Matrix Beta
(8 x 8 General Square Matrix)

	Powerless71	SEI	f_Alien67	f_Alien71
Anomie67	0	0	1.0000	0
Anomie71	0	0	0	1.0000
Education	0	0	0	0
Powerless67	0	0	0.8330	0
Powerless71	0	0	0	0.8330
SEI	0	0	0	0
f_Alien67	0	0	0	0
f_Alien71	0	0	0.5931 [beta]	0

Output 26.28.16 Estimate of the _EQSBETA_ Matrix by the LINEQS Model Specification

EQSBETA Matrix

	Anomie67	Anomie71	Education	Powerless67
Anomie67	0	0	0	0
Anomie71	0	0	0	0
Education	0	0	0	0
Powerless67	0	0	0	0
Powerless71	0	0	0	0
SEI	0	0	0	0
f_Alien67	0	0	0	0
f_Alien71	0	0	0	0

EQSBETA Matrix

	Powerless71	SEI	f_Alien67	f_Alien71
Anomie67	0	0	1.0000	0
Anomie71	0	0	0	1.0000
Education	0	0	0	0
Powerless67	0	0	0.8330	0
Powerless71	0	0	0	0.8330
SEI	0	0	0	0
f_Alien67	0	0	0	0
f_Alien71	0	0	0.5931 [beta]	0

Output 26.28.17 shows the estimates of the **Gamma** matrix by the COSAN model specification. Again, these estimates are essentially the same as the estimates of the _EQSGAMMA_ matrix obtained from the LINEQS model specification, as shown in Output 26.28.18.

Output 26.28.17 Estimate of the Gamma Matrix by the COSAN Model Specification

Model Matrix Gamma
(8 x 9 General Rectangular Matrix)

	f_SES	e1	e3	e5	e2
Anomie67	0	1.0000	0	0	0
Anomie71	0	0	1.0000	0	0
Education	1.0000	0	0	1.0000	0
Powerless67	0	0	0	0	1.0000
Powerless71	0	0	0	0	0
SEI	5.3689 [lambda]	0	0	0	0
f_Alien67	-0.6299 [gamma1]	0	0	0	0
f_Alien71	-0.2409 [gamma2]	0	0	0	0

Model Matrix Gamma
(8 x 9 General Rectangular Matrix)

	e4	e6	d1	d2
Anomie67	0	0	0	0
Anomie71	0	0	0	0
Education	0	0	0	0
Powerless67	0	0	0	0
Powerless71	1.0000	0	0	0
SEI	0	1.0000	0	0
f_Alien67	0	0	1.0000	0
f_Alien71	0	0	0	1.0000

Output 26.28.18 Estimate of the _EQSGAMMA_ Matrix by the LINEQS Model Specification

EQSGAMMA Matrix

	f_SES	e1	e3	e5	e2
Anomie67	0	1.0000	0	0	0
Anomie71	0	0	1.0000	0	0
Education	1.0000	0	0	1.0000	0
Powerless67	0	0	0	0	1.0000
Powerless71	0	0	0	0	0
SEI	5.3688 [lambda]	0	0	0	0
f_Alien67	-0.6299 [gamma1]	0	0	0	0
f_Alien71	-0.2409 [gamma2]	0	0	0	0

EQSGAMMA Matrix

	e4	e6	d1	d2
Anomie67	0	0	0	0
Anomie71	0	0	0	0
Education	0	0	0	0
Powerless67	0	0	0	0
Powerless71	1.0000	0	0	0
SEI	0	1.0000	0	0
f_Alien67	0	0	1.0000	0
f_Alien71	0	0	0	1.0000

Finally, Output 26.28.19 shows the estimates of the **Phi** matrix by the COSAN model specification. These estimates are essentially the same as the estimates of the _EQSPHI_ matrix obtained from the LINEQS model specification, as shown in Output 26.28.20.

Output 26.28.19 Estimate of the **Phi** Matrix by the COSAN Model Specification

Model Matrix Phi
(9 x 9 Symmetric Matrix)

	f_SES	e1	e3	e5	e2
f_SES	6.6162 [phi]	0	0	0	0
e1	0	3.6078 [theta1]	0.9058 [theta5]	0	0
e3	0	0.9058 [theta5]	3.6078 [theta1]	0	0
e5	0	0	0	2.9938 [theta3]	0
e2	0	0	0	0	3.5950 [theta2]
e4	0	0	0	0	0.9058 [theta5]
e6	0	0	0	0	0
d1	0	0	0	0	0
d2	0	0	0	0	0

Output 26.28.19 *continued*

Model Matrix Phi
(9 x 9 Symmetric Matrix)

	e4	e6	d1	d2
f_SES	0	0	0	0
e1	0	0	0	0
e3	0	0	0	0
e5	0	0	0	0
e2	0.9058 [theta5]	0	0	0
e4	3.5950 [theta2]	0	0	0
e6	0	259.5738 [theta4]	0	0
d1	0	0	5.6705 [psi1]	0
d2	0	0	0	4.5148 [psi2]

Output 26.28.20 Estimate of the _EQSPHI_ Matrix by the LINEQS Model Specification

EQSPHI Matrix

	f_SES	e1	e3	e5	e2
f_SES	6.6163 [phi]	0	0	0	0
e1	0	3.6080 [theta1]	0.9058 [theta5]	0	0
e3	0	0.9058 [theta5]	3.6080 [theta1]	0	0
e5	0	0	0	2.9937 [theta3]	0
e2	0	0	0	0	3.5949 [theta2]
e4	0	0	0	0	0.9058 [theta5]
e6	0	0	0	0	0
d1	0	0	0	0	0
d2	0	0	0	0	0

Output 26.28.20 *continued*

EQSPHI Matrix

	e4	e6	d1	d2
f_SES	0	0	0	0
e1	0	0	0	0
e3	0	0	0	0
e5	0	0	0	0
e2	0.9058 [theta5]	0	0	0
e4	3.5949 [theta2]	0	0	0
e6	0	259.5764 [theta4]	0	0
d1	0	0	5.6705 [psi1]	0
d2	0	0	0	4.5148 [psi2]

Example 26.29: Second-Order Confirmatory Factor Analysis

A second-order confirmatory factor analysis model is applied to a correlation matrix of Thurstone reported by McDonald (1985). The data set is shown in the following DATA step:

```
data Thurst(TYPE=CORR);
title "Example of THURSTONE resp. McDONALD (1985, p.57, p.105)";
   _TYPE_ = 'CORR'; Input _NAME_ $ Obs1-Obs9;
   label obs1='Sentences' obs2='Vocabulary' obs3='Sentence Completion'
         obs4='First Letters' obs5='Four-letter Words' obs6='Suffices'
         obs7='Letter series' obs8='Pedigrees' obs9='Letter Grouping';
   datalines;
obs1  1.       .      .      .      .      .      .      .      .
obs2   .828   1.      .      .      .      .      .      .      .
obs3   .776   .779   1.      .      .      .      .      .      .
obs4   .439   .493    .460  1.      .      .      .      .      .
obs5   .432   .464    .425   .674  1.      .      .      .      .
obs6   .447   .489    .443   .590   .541  1.      .      .      .
obs7   .447   .432    .401   .381   .402   .288  1.      .      .
obs8   .541   .537    .534   .350   .367   .320   .555  1.      .
obs9   .380   .358    .359   .424   .446   .325   .598   .452  1.
;
```

Using the LINEQS modeling language, you specify the three-term second-order factor analysis model in the following statements:

```
proc calis data=Thurst nobs=213 corr nose;
lineqs
   obs1 =  x1 * f1 + e1,
   obs2 =  x2 * f1 + e2,
   obs3 =  x3 * f1 + e3,
   obs4 =  x4 * f2 + e4,
   obs5 =  x5 * f2 + e5,
   obs6 =  x6 * f2 + e6,
   obs7 =  x7 * f3 + e7,
   obs8 =  x8 * f3 + e8,
   obs9 =  x9 * f3 + e9,
   f1   = x10 * f4 + e10,
   f2   = x11 * f4 + e11,
   f3   = x12 * f4 + e12;
variance
   f4      = 1.,
   e1-e9   = u1-u9,
   e10-e12 = 3 * 1.;
bounds
   0. <= u1-u9;
run;
```

In the PROC CALIS statement, you specify the data set in the DATA= option and the number of observations in the NOBS= option. With the CORR option, you request the correlations be analyzed. You use the NOSE option to suppress the computation of standard error estimates.

In the LINEQS statement, the first-order loadings for the three factors, f1, f2, and f3, each refer to three variables, X1-X3, X4-X6, and X7-X9, respectively. One second-order factor, f4, reflects the correlations among the three first-order factors, f1, f2, and f3.

In the VARIANCE statement, you fix the variance of f4 to 1.0 for identification. The variances of error terms e1–e9 are free parameters u1–u9. The error variances for the three first-order factors are also fixed at 1.0 for identification purposes.

You also specify the boundary constraints for the error variance parameters u1–u9. You require them to be positive in the estimation.

Output 26.29.1 shows the estimation results.

Output 26.29.1 Estimation Results of the Second-Order Factor Model for Thurstone Data: LINEQS Model

```
                         Linear Equations

                Obs1 =     0.5151*f1   +   1.0000 e1
                                  x1
                Obs2 =     0.5203*f1   +   1.0000 e2
                                  x2
                Obs3 =     0.4874*f1   +   1.0000 e3
                                  x3
                Obs4 =     0.5211*f2   +   1.0000 e4
                                  x4
                Obs5 =     0.4971*f2   +   1.0000 e5
                                  x5
                Obs6 =     0.4381*f2   +   1.0000 e6
                                  x6
                Obs7 =     0.4524*f3   +   1.0000 e7
                                  x7
                Obs8 =     0.4173*f3   +   1.0000 e8
                                  x8
                Obs9 =     0.4076*f3   +   1.0000 e9
                                  x9
                f1   =     1.4438*f4   +   1.0000 e10
                                  x10
                f2   =     1.2538*f4   +   1.0000 e11
                                  x11
                f3   =     1.4065*f4   +   1.0000 e12
                                  x12
```

Output 26.29.1 *continued*

Estimates for Variances of Exogenous Variables

Variable Type	Variable	Parameter	Estimate
Latent	f4		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	0.50509
Disturbance	e10		1.00000
	e11		1.00000
	e12		1.00000

Alternatively, you can use the COSAN model specification for analyzing the same data set. First, under the second-order factor model, the covariance structures of the observed variables can be derived as

$$\mathbf{\Sigma} = \mathbf{F1} * \mathbf{F2} * \mathbf{P} * \mathbf{F2}' * \mathbf{F1}' + \mathbf{F1} * \mathbf{U2} * \mathbf{F1}' + \mathbf{U1}$$

where **F1** is the 9×3 first-order loading matrix for the observed variables, **F2** is the 3×1 second-order loading matrix for the first-order factors, **P** is the 1×1 covariance matrix for the second-order factor f4, **U2** is the 3×3 error covariance matrix of the first-order factors f1–f3 (or the covariance matrix of the error terms e10–12), and **U1** is the 9×9 error covariance matrix for the observed variables (or the covariance matrix of the error terms e1–9).

Matrix **F1** contains the loading parameters x1–x9 and matrix **F2** contains the loading parameters x10–x12. Because there is only one second-order factor f4 in the model, matrix **P** is a scalar, which is a fixed constant 1 in the LINEQS model. Matrix **U2** is an identity matrix because all error variances are fixed at 1 and they are not correlated. Matrix **U2** is a diagonal matrix that contains the parameters u1–u9. Given this information, you can use the following statements to specify the second-order factor model as a COSAN model:

```
proc calis data=Thurst nobs=213 corr nose;
   cosan
      var = obs1-obs9,
      F1(3) * F2(1) * P(1,IDE) + F1(3) * U2(3,IDE) + U1(9,DIA);
   matrix F1
      [1 , @1] = x1-x3,
      [4 , @2] = x4-X6,
      [7 , @3] = x7-x9;
   matrix F2
      [ ,1]    = x10-x12;
   matrix U1
      [1,1]    = u1-u9;
   bounds
      0. <= u1-u9;
   vnames
      F1 = [f1 f2 f3],
      F2 = [f4],
      U1 = [e1-e9];
run;
```

In the PROC CALIS statement, you specify the observed variables in the VAR= option and the covariance structures for the observed variables. In the terms of the covariance structure formula, you need to specify the expressions only up the central symmetric matrices. The latter parts of these expressions are redundant and can be generated automatically by PROC CALIS, as shown in Output 26.29.2.

Output 26.29.2 The Covariance Structures and Model Matrices of the Second-Order Factor Model: COSAN Model

```
                    COSAN Model Structures

      Sigma =    F1*F2*P*F2`*F1`  + F1*U2*F1`  + U1

                  Summary of Model Matrices

    Matrix      N Row      N Col     Matrix Type

    F1              9          3     GEN: Rectangular
    F2              3          1     GEN: Vector
    P               1          1     IDE: Identity
    U1              9          9     DIA: Diagonal
    U2              3          3     IDE: Identity
```

Output 26.29.2 shows that the intended covariance structures for the observed variables are being analyzed. The matrix types are shown next. Matrix **F1** is a rectangular matrix and matrix **F2** is a vector, although they have the default general (GEN) matrix type. Matrices **P** and **U2** are fixed identity (IDE) matrices in the model. For these two matrices, you do not need to specify any of their elements by using the MATRIX statement because they are already well-defined with the IDE type. Lastly, matrix **U1** is a diagonal (DIA) matrix in the model.

Output 26.29.3 shows the estimates of the first-order factor loading matrix **F1**.

Output 26.29.3 Estimation of the **F1** Matrix of the Second-Order Factor Model: COSAN Model

Model Matrix F1
(9 x 3 General Rectangular Matrix)

	f1	f2	f3
Obs1	0.5151 [x1]	0	0
Obs2	0.5203 [x2]	0	0
Obs3	0.4874 [x3]	0	0
Obs4	0	0.5211 [x4]	0
Obs5	0	0.4971 [x5]	0
Obs6	0	0.4381 [x6]	0
Obs7	0	0	0.4524 [x7]
Obs8	0	0	0.4173 [x8]
Obs9	0	0	0.4076 [x9]

In the MATRIX statement for **F1**, you specify the pattern of the loadings. In the first entry of the MATRIX statement, you specify the loadings in the following elements: `[1,1]`, `[2,1]`, and `[3,1]`. They are free parameters x1–x3, respectively. Notice that the @ sign is necessary in the first entry because the elements being defined would have been `[1,1]`, `[2,2]`, and `[3,3]` otherwise. The @ sign fixes the column number to 1. See the MATRIX statement for more details about the notation. Similarly, you define the other clusters of loading in the second and third entries in the MATRIX statement for **F1**. This explains the pattern of factor loadings in Output 26.29.3. These loading estimates x1–x9 match those by the LINEQS model specification, as shown in Output 26.29.1.

Output 26.29.3 shows the estimates of the second-order factor loading matrix **F2**.

Output 26.29.4 Estimation of the **F2** Matrix of the Second-Order Factor Model: COSAN Model

```
                    Model Matrix F2
                 (3 x 1 Column Vector)

                                  f4

                 f1           1.4438
                               [x10]

                 f2           1.2538
                               [x11]

                 f3           1.4066
                               [x12]
```

In the MATRIX statement for **F2**, you do not specify the row numbers in the `[ ,1]` specification. PROC CALIS interprets this as stating that all the valid elements in the first column are being specified in the parameter list. In the current example, this means that elements **F2**[1, 1], **F2**[2, 1], and **F2**[3, 1] are filled with the free parameters x10, x11, and x12, respectively. Output 26.29.3 shows these specification and the corresponding estimates, which match those of the LINEQS model specification, as shown in Output 26.29.1.

Output 26.29.5 shows the estimates of the error covariance matrix **U1**.

Output 26.29.5 Estimation of the **U1** Matrix of the Second-Order Factor Model: COSAN Model

Model Matrix U1
(9 x 9 Diagonal Matrix)

	e1	e2	e3	e4	e5
e1	0.1815 [u1]	0	0	0	0
e2	0	0.1649 [u2]	0	0	0
e3	0	0	0.2671 [u3]	0	0
e4	0	0	0	0.3015 [u4]	0
e5	0	0	0	0	0.3645 [u5]
e6	0	0	0	0	0
e7	0	0	0	0	0
e8	0	0	0	0	0
e9	0	0	0	0	0

Output 26.29.5 *continued*

Model Matrix U1
(9 x 9 Diagonal Matrix)

	e6	e7	e8	e9
e1	0	0	0	0
e2	0	0	0	0
e3	0	0	0	0
e4	0	0	0	0
e5	0	0	0	0
e6	0.5064 [u6]	0	0	0
e7	0	0.3903 [u7]	0	0
e8	0	0	0.4814 [u8]	0
e9	0	0	0	0.5051 [u9]

In the MATRIX statement for **U1**, you specify the diagonal elements of the matrix by using the starting element at `[1,1]`. The parameter assignment proceeds to `[2,2]`, `[3,3]` and so on such that all the trailing parameters u1–u9 are filled. This means that the last element **U1**[9, 9] is a free parameter named u9. Output 26.29.5 confirms this intended pattern. Again, all these error variance estimates match those by the LINEQS model specification, as shown in Output 26.29.1.

Example 26.30: Linear Relations among Factor Loadings: COSAN Model Specification

This example reanalyzes the models in Example 26.26 by using the COSAN modeling language. The correlation matrix of six variables from Kinzer and Kinzer (N=326) is used (see Guttman 1957). McDonald (1980) uses this data set to demonstrate the fitting of a factor analysis model with linear constraints on factor loadings. Two factors are assumed for the data. The factor loading matrix **B** is shown in the following:

$$\mathbf{B} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \\ b_{51} & b_{52} \\ b_{61} & b_{62} \end{pmatrix}$$

The loadings on the second factor are linearly related to the loadings on the first factor, as described by the following formula:

$$b_{j2} = \alpha - b_{j1}, \quad j = 1, \ldots, 6$$

The correlation structures are represented by

$$\mathbf{P} = \mathbf{BB}' + \mathbf{\Psi}$$

where $\Psi = \text{diag}(\psi_{11}, \psi_{22}, \psi_{33}, \psi_{44}, \psi_{55}, \psi_{66})$ represents the diagonal matrix of unique variances for the variables. Because matrix **P** is a correlation matrix, its diagonal elements are fixed constants 1. This means that the diagonal elements of the correlation structures must also satisfy the following condition:

$$\mathbf{\Psi}_{jj} = 1. - b_{j1}^2 - b_{j2}^2, \quad j = 1, \ldots, 6$$

To analyze the correlation structures by using PROC CALIS, you formulate a covariance structure model with such correlation structures embedded in the model. That is, you want to fit the following covariance structure model to the Kinzer data:

$$\mathbf{\Sigma} = \mathbf{DPD}' = \mathbf{D}(\mathbf{BB}' + \mathbf{\Psi})\mathbf{D}' = \mathbf{DBB}'\mathbf{D}' + \mathbf{D\Psi D}'$$

where **D** is a 6 x 6 diagonal matrix that contains the population standard deviations of the observed variables.

The following statements use the COSAN modeling language to specify this covariance structure model:

```
proc calis data=Kinzer nobs=326 nose;
   cosan
      var= var1-var6,
      D(6,DIA) * B(2,GEN) + D(6,DIA) * Psi(6,DIA);
   matrix B
      [ ,1] = b11 b21 b31 b41 b51 b61,
      [ ,2] = b12 b22 b32 b42 b52 b62;
   matrix Psi
      [1,1] = psi1-psi6;
   matrix D
      [1,1] = d1-d6;
   parameters alpha (1.);

   /* SAS Programming Statements to Define Dependent Parameters*/
   /* 6 constraints on the factor loadings */
   b12  = alpha - b11;
   b22  = alpha - b21;
   b32  = alpha - b31;
   b42  = alpha - b41;
   b52  = alpha - b51;
   b62  = alpha - b61;

   /* 6 Constraints on Correlation structures */
   psi1 = 1. - b11 * b11 - b12 * b12;
   psi2 = 1. - b21 * b21 - b22 * b22;
   psi3 = 1. - b31 * b31 - b32 * b32;
   psi4 = 1. - b41 * b41 - b42 * b42;
   psi5 = 1. - b51 * b51 - b52 * b52;
   psi6 = 1. - b61 * b61 - b62 * b62;
   vnames
      D   = [var1-var6],
      B   = [factor1 factor2],
      Psi = D;
run;
```

In the PROC CALIS statement, you specify the data set by the DATA= option and the number of observations by the NOBS= option. You also use the NOSE option to suppress the printing of the standard error estimates.

In the COSAN statement, you specify the variables for the covariance structure analysis in the VAR= option. Next, you specify the covariance structure formula for the variables. When generating the covariance structure expressions for the terms, PROC CALIS examines the matrix type of the last matrix in each term to determine how the expression is generated. If the last matrix in a term is not a symmetric matrix (including diagonal or identity matrix), the transpose of the last matrix would be included in the expression. This ensures that a symmetric matrix expression is formed for the covariance structures. For example, the first term in the current covariance structure formula is **`D(6,DIA)*B(2,GEN)`**. Because $\mathbf{B}$ is not a symmetric matrix, the expression generated by PROC CALIS is

$$\mathbf{D} * \mathbf{B} * \mathbf{B}' * \mathbf{D}'$$

However, for the second term `D(6,DIA)* Psi(6,DIA)`, matrix **Psi** is a symmetric matrix so that the expression generated by PROC CALIS is

$$\mathbf{D} * \mathbf{Psi} * \mathbf{D}'$$

Output 26.30.1 shows the covariance structure model and the model matrices. With **Psi** representing the unique variance matrix $\boldsymbol{\Psi}$, the printed covariance structure formula for **Sigma** is clearly what you intend to specify.

Output 26.30.1 The Covariance Structures and Model Matrices: Linearly Constrained Loadings

```
                    COSAN Model Structures

             Sigma =    D*B*B`*D`  + D*Psi*D`

                  Summary of Model Matrices

          Matrix     N Row     N Col    Matrix Type

          B              6         2    GEN: Rectangular
          D              6         6    DIA: Diagonal
          Psi            6         6    DIA: Diagonal
```

In the MATRIX statements, you specify the parameters in the model matrices. You use parameters with the b prefix to name the two columns of loadings of the factor matrix **B**. You use free parameters psi1–psi6 for the diagonal elements of the **Psi** matrix, and free parameters d1–d6 for the diagonal elements of the **D** matrix. Next, you use a PARAMETERS statement to define an independent parameter alpha in the model. This parameter takes an initial value of 1.0. Using this independent parameter and six SAS programming statements, you define the loadings in the second column of matrix **B** as functions of the loadings in the first column of the same matrix.

You use six more SAS programming statements to define the unique variance parameters psi1–psi6 as dependent parameters of the factor loadings. These constraints ensure that the embedded correlation structures have diagonal elements fixed at 1.0.

Lastly, you use the VNAMES statement to label the column names of the model matrices. The column names of the diagonal matrix **D** are the same as the observed variables. The column names of matrix **B** are for the factor names.

As compared with the covariance structure specification (that is, the second specification) by the LINEQS model in Example 26.26, the current COSAN specification seems to be more direct and concise in specifying the parameter constraints. Because of the direct references to the matrix elements in the COSAN modeling language, you can set the required 12 constraints in a very straightforward way as the 12 SAS programming statements in the preceding specification. However, with the LINEQS model specification language in Example 26.26, you need 18 more SAS programming statements to define the correct constraints for the same covariance structure model.

Output 26.30.2 shows the fit summary table. The chi-square test statistic is 14.63 with df=8 (p = 0.067). These are the same model fitting results as using the LINEQS model specification, as shown in Output 26.26.4 of Example 26.26.

Output 26.30.2 Model Fit: Linearly Constrained Loadings with Embedded Correlation Structures

Fit Summary

Chi-Square	14.6269
Chi-Square DF	8
Pr > Chi-Square	0.0668

Output 26.30.3 shows the estimation of the loading matrix **B**. These estimates of factor loadings are essentially the same as those obtained from the LINEQS model specification, as shown in Output 26.26.6, except that the two columns of the loading matrix **B** are switched. The column switching is not a concern because the factor labels are arbitrary.

Output 26.30.3 Estimation of the **B** Matrix by the COSAN Model Specification

Model Matrix B
(6 x 2 General Rectangular Matrix)

	factor1	factor2
var1	0.6318 [b11]	0.3422 [b12]
var2	0.6531 [b21]	0.3210 [b22]
var3	0.4822 [b31]	0.4918 [b32]
var4	0.3985 [b41]	0.5755 [b42]
var5	0.1971 [b51]	0.7769 [b52]
var6	0.3074 [b61]	0.6666 [b62]

Output 26.30.4 shows the estimation of the scaling matrix **D**. All these standard deviation estimates for the observed variables match those obtained from the LINEQS model specification, as shown in Output 26.26.6.

Output 26.30.4 Estimation of the **D** Matrix by the COSAN Model Specification

Model Matrix D
(6 x 6 Diagonal Matrix)

	var1	var2	var3	var4	var5	var6
var1	1.0077 [d1]	0	0	0	0	0
var2	0	0.9971 [d2]	0	0	0	0
var3	0	0	0.9908 [d3]	0	0	0
var4	0	0	0	0.9909 [d4]	0	0
var5	0	0	0	0	0.9964 [d5]	0
var6	0	0	0	0	0	1.0169 [d6]

Output 26.30.5 shows the estimation of the unique covariance matrix **Psi**. All these unique variance parameter estimates match those obtained from the LINEQS model specification, as shown in Output 26.26.6.

Output 26.30.5 Estimation of the **Psi** Matrix by the COSAN Model Specification

Model Matrix Psi
(6 x 6 Diagonal Matrix)

	var1	var2	var3	var4	var5	var6
var1	0.4837 [psi1]	0	0	0	0	0
var2	0	0.4705 [psi2]	0	0	0	0
var3	0	0	0.5256 [psi3]	0	0	0
var4	0	0	0	0.5100 [psi4]	0	0
var5	0	0	0	0	0.3576 [psi5]	0
var6	0	0	0	0	0	0.4612 [psi6]

Finally, Output 26.30.6 shows the estimation of the independent parameter alpha. The same estimate of alpha is shown in Output 26.26.6.

Output 26.30.6 Estimation of the Independent Parameter alpha by the COSAN Model Specification

Additional Parameters

Type	Parameter	Estimate
Independent	alpha	0.97400

Example 26.31: Ordinal Relations among Factor Loadings

The same data set as in Example 26.30 is used in McDonald (1980) for analysis with ordinally constrained factor loadings. In Example 26.26, the results of the linearly constrained factor analysis show that the loadings of the two factors are ordered as 2, 1, 3, 4, 6, 5. McDonald (1980) then tests the hypothesis that the factor loadings are all nonnegative and can be ordered in the following manner:

$$b_{11} \geq b_{21} \geq b_{31} \geq b_{41} \geq b_{51} \geq b_{61}$$

$$b_{12} \leq b_{22} \leq b_{32} \leq b_{42} \leq b_{52} \leq b_{62}$$

In this example, you implement these ordinal relationships by using the LINCON statement in the following COSAN model specification:

```
proc calis data=Kinzer nobs=326 nose;
   cosan
      var= var1-var6,
      D(6,DIA) * B(2,GEN) + D(6,DIA) * Psi(6,DIA);
   matrix B
      [ ,1]= b11 b21 b31 b41 b51 b61,
      [ ,2]= 0.  b22 b32 b42 b52 b62;
   matrix Psi
      [1,1]= psi1-psi6;
   matrix D
      [1,1]= d1-d6 ;
   lincon
      b61  <= b51,
      b51  <= b41,
      b41  <= b31,
      b31  <= b21,
      b21  <= b11,
       0.  <= b22,
      b22  <= b32,
      b32  <= b42,
      b42  <= b52,
      b52  <= b62;

   /* SAS Programming Statements */
   /* 6 Constraints on Correlation structures */
   psi1 = 1. - b11 * b11;
   psi2 = 1. - b21 * b21 - b22 * b22;
   psi3 = 1. - b31 * b31 - b32 * b32;
   psi4 = 1. - b41 * b41 - b42 * b42;
   psi5 = 1. - b51 * b51 - b52 * b52;
   psi6 = 1. - b61 * b61 - b62 * b62;
   vnames
       B   = [factor1 factor2],
       Psi = [var1-var6],
       D   = Psi;
run;
```

As in Example 26.30, correlation structures are analyzed in the current example so that the unique variance parameters psi1-psi6 are defined as functions of the loadings in the SAS programming statements. However, the loading parameters are no longer not constrained in the current model. Instead, you impose ordinal constraints on the loading parameters. First, b21 is fixed at 0 for identification purposes. Then, you use the LINCON statement to specify the ordinal relations of the factor loadings.

As shown in Output 26.31.1, the solution converges in 12 iterations. In the fit summary table, the chi-square test statistic is 8.48 ($df = 6$, $p = 0.20$). This indicates a good fit. However, in the model there are 11 loading parameters (the b's) and 6 population standard deviation parameters (the d's). The degrees of freedom should have been $4 = 21 - 11 - 6$, but why is this number 6 in the fit summary table?

Output 26.31.1 Final Iteration Status and Fit

```
                          Optimization Results

Iterations                          12  Function Calls                    29
Jacobian Calls                      14  Active Constraints                 2
Objective Function        0.0260990149  Max Abs Gradient Element  2.7626747E-6
Lambda                               0  Actual Over Pred Change   1.1572072766
Radius                    0.0000851592

Convergence criterion (ABSGCONV=0.00001) satisfied.

                              Fit Summary

                         Chi-Square              8.4822
                         Chi-Square DF                6
                         Pr > Chi-Square         0.2049
```

The reason is that there are two active constraints in the solution, resulting in two free parameters fewer in the final solution than originally specified. Active constraints are those inequality constraints that are fulfilled on the boundary equalities. As shown in the "Optimization Results" table, the number of active constraints for the current fitting is two. The default treatment in PROC CALIS is to treat these active constraints as if they were going to happen for all possible repeated sampling. This might as well be seen as fitting the active equality constraints on every possible repeated sample. This results in an increase of the degrees of freedom for model fit, as adjusted in the current fit summary table in Output 26.31.1. To warn you about the degrees-of-freedom adjustment, the following messages are also printed with the output:

```
WARNING: There are 2 active boundary or linear inequality constraints at
         the solution. The standard errors and chi-square test statistic
         assume that the solution is located in the interior of the
         parameter space; hence, they do not apply if it is likely that
         some different set of inequality a constraints could be ctive.

NOTE: The degrees of freedom are increased by the number of active
      constraints.  The number of parameters in calculating fit indices
      is decreased by the number of active constraints. To turn off the
      adjustment, use the NOADJDF option.
```

When active constraints are encountered, you need to be cautious about two implications. First, the estimates fall on the boundary of the parameter space originally specified. As shown in Output 26.31.2, estimates for b11 and b21 are the same, and so are the pair of estimates for b52 and b62. These pairs of parameters were originally constrained by inequalities in the model. For example, b62 was constrained to be at least as large as b52. The fact that this constraint is honored only on the bound means that a better model fit might exist with b62 being smaller than b52. Similarly, a better model fit might result without requiring b11 to be at least as large as b21. Therefore, solutions with active boundary constraints might imply that the original strict inequality constraints are not appropriate for the data.

Output 26.31.2 Estimation of the Factor Loading Matrix B

Model Matrix B
(6 x 2 General Rectangular Matrix)

	factor1	factor2
var1	0.7100 [b11]	0
var2	0.7100 [b21]	0.0393 [b22]
var3	0.6799 [b31]	0.2463 [b32]
var4	0.6561 [b41]	0.3295 [b42]
var5	0.5541 [b51]	0.5432 [b52]
var6	0.4733 [b61]	0.5432 [b62]

Output 26.31.3 Estimation of the Scaling Matrix D and Unique Covariance Matrix Psi

Model Matrix D
(6 x 6 Diagonal Matrix)

	var1	var2	var3	var4	var5	var6
var1	1.0022 [d1]	0	0	0	0	0
var2	0	0.9985 [d2]	0	0	0	0
var3	0	0	1.0004 [d3]	0	0	0
var4	0	0	0	1.0004 [d4]	0	0
var5	0	0	0	0	0.9990 [d5]	0
var6	0	0	0	0	0	1.0021 [d6]

Model Matrix Psi
(6 x 6 Diagonal Matrix)

	var1	var2	var3	var4	var5	var6
var1	0.4959 [psi1]	0	0	0	0	0
var2	0	0.4944 [psi2]	0	0	0	0
var3	0	0	0.4771 [psi3]	0	0	0
var4	0	0	0	0.4610 [psi4]	0	0
var5	0	0	0	0	0.3979 [psi5]	0
var6	0	0	0	0	0	0.4809 [psi6]

The second implication for the presence of active constraints is that the chi-square test statistic and the standard error estimates are computed as if repeated samples were fitted by the model with the presence of the active equality constraints. The degrees-of-freedom adjustment by PROC CALIS is based on this assumption. However, if the particular active constraints reflect only a rare sampling event, the degrees-of-freedom adjustment (or even the computation of the chi-square statistic and standard error estimates) might not be justified. Unfortunately, whether the active constraints are reflecting the truth of the model or pure sampling fluctuation is usually difficult to determine.

Example 26.32: Longitudinal Factor Analysis

The following example (McDonald 1980) illustrates both the ability of PROC CALIS to formulate complex covariance structure analysis problems by the generalized COSAN matrix model and the use of programming statements to impose nonlinear constraints on the parameters. The example is a longitudinal factor analysis that uses the Swaminathan (1974) model. For $m = 3$ tests, $k = 3$ occasions, and $r = 2$ factors, the covariance structure model is formulated as follows:

$$\mathbf{\Sigma} = \mathbf{F}_1\mathbf{F}_2\mathbf{F}_3\mathbf{L}\mathbf{F}_3^{-1}\mathbf{F}_2^{-1}\mathbf{P}(\mathbf{F}_2^{-1})'(\mathbf{F}_3^{-1})'\mathbf{L}'\mathbf{F}_3'\mathbf{F}_2'\mathbf{F}_1' + \mathbf{U}^2$$

$$\mathbf{F}_1 = \begin{pmatrix} \mathbf{B}_1 & & \\ & \mathbf{B}_2 & \\ & & \mathbf{B}_3 \end{pmatrix}, \quad \mathbf{F}_2 = \begin{pmatrix} \mathbf{I}_2 & & \\ & \mathbf{D}_2 & \\ & & \mathbf{D}_2 \end{pmatrix}, \quad \mathbf{F}_3 = \begin{pmatrix} \mathbf{I}_2 & & \\ & \mathbf{I}_2 & \\ & & \mathbf{D}_3 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} \mathbf{I}_2 & 0 & 0 \\ \mathbf{I}_2 & \mathbf{I}_2 & 0 \\ \mathbf{I}_2 & \mathbf{I}_2 & \mathbf{I}_2 \end{pmatrix}, \quad \mathbf{P} = \begin{pmatrix} \mathbf{I}_2 & & \\ & \mathbf{S}_2 & \\ & & \mathbf{S}_3 \end{pmatrix} \quad , \quad \mathbf{U} = \begin{pmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} & \mathbf{U}_{13} \\ \mathbf{U}_{21} & \mathbf{U}_{22} & \mathbf{U}_{23} \\ \mathbf{U}_{31} & \mathbf{U}_{32} & \mathbf{U}_{33} \end{pmatrix}$$

$$\mathbf{S}_2 = \mathbf{I}_2 - \mathbf{D}_2^2, \qquad \mathbf{S}_3 = \mathbf{I}_2 - \mathbf{D}_3^2$$

The Swaminathan longitudinal factor model assumes that the factor scores for each (m) common factor change from occasion to occasion (k) according to a first-order autoregressive scheme. The matrix $\mathbf{F}_1$ contains the k factor loading matrices $\mathbf{B}_1$, $\mathbf{B}_2$, and $\mathbf{B}_3$ (each is $n \times m$). The matrices $\mathbf{D}_2, \mathbf{D}_3, \mathbf{S}_2, \mathbf{S}_3$ and $\mathbf{U}_{ij}, i, j = 1, \ldots, k$, are diagonal, and the matrices $\mathbf{D}_i$ and $\mathbf{S}_i, i = 2, \ldots, k$, are subjected to the constraint

$$\mathbf{S}_i + \mathbf{D}_i^2 = \mathbf{I}$$

Although the covariance structure model looks pretty complicated, it poses no problem for the COSAN model specifications. Since the constructed correlation matrix given by McDonald (1980) is singular, only unweighted least squares (METHOD=LS) estimates can be computed. The following statements specify the COSAN model for the correlation structures.

```
data Mcdon(TYPE=CORR);
Title "Swaminathan's Longitudinal Factor Model, Data: McDONALD(1980)";
Title2 "Constructed Singular Correlation Matrix, GLS & ML not possible";
   _TYPE_ = 'CORR'; INPUT _NAME_ $ obs1-obs9;
   datalines;
obs1  1.000    .      .      .      .      .      .      .      .
obs2   .100  1.000    .      .      .      .      .      .      .
obs3   .250   .400  1.000    .      .      .      .      .      .
obs4   .720   .108   .270  1.000    .      .      .      .      .
obs5   .135   .740   .380   .180  1.000    .      .      .      .
obs6   .270   .318   .800   .360   .530  1.000    .      .      .
obs7   .650   .054   .135   .730   .090   .180  1.000    .      .
obs8   .108   .690   .196   .144   .700   .269   .200  1.000    .
obs9   .189   .202   .710   .252   .336   .760   .350   .580  1.000
     ;
```

```
proc calis data=Mcdon method=ls nobs=100 corr;
cosan
   var = obs1-obs9,
   F1(6,GEN) * F2(6,DIA) * F3(6,DIA) * L(6,LOW) * F3(6,DIA,INV)
             * F2(6,DIA,INV) * P(6,DIA) + U(9,SYM);
   matrix F1
            [1 , @1] = x1-x3,
            [2 , @2] = x4-x5,
            [4 , @3] = x6-x8,
            [5 , @4] = x9-x10,
            [7 , @5] = x11-x13,
            [8 , @6] = x14-x15;
   matrix F2
            [1,1]= 2 * 1. x16 x17 x16 x17;
   matrix F3
            [1,1]= 4 * 1. x18 x19;
   matrix L
            [1,1]= 6 * 1.,
            [3,1]= 4 * 1.,
            [5,1]= 2 * 1.;
   matrix P
            [1,1]= 2 * 1. x20-x23;
   matrix U
            [1,1]= x24-x32,
            [4,1]= x33-x38,
            [7,1]= x39-x41;
   bounds 0. <= x24-x32,
         -1. <= x16-x19 <= 1.;
   /* SAS programming statements for dependent parameters */
   x20 = 1. - x16 * x16;
   x21 = 1. - x17 * x17;
   x22 = 1. - x18 * x18;
   x23 = 1. - x19 * x19;
run;
```

In the PROC CALIS statement, you use the NOBS= option to specify the number of observations. The CORR option requests the analysis of the correlation matrix.

In the COSAN statement, you list the observed variables for the analysis in the VAR= option. Then you specify the formula for the covariance structures. Notice that in the covariance structure formula, some matrices are specified twice. That is, matrix **F2** and **F3** appear in two different places. Matrices with the same name means that they are identical—which certainly makes sense. In addition, you can apply different transformations to the same matrix in different locations of the matrix formula. For example, you do not transform matrix **F2** in the first location, but the same matrix is inverted (INV) later in the expression. Similarly for matrix **F3**.

Next, you define the parameters in the six distinct model matrices by six MATRIX statements. Each matrix has some specific patterns under the covariance structure model. For the **F1** matrix, it has the following pattern for the free parameters in the model:

```
        col1  col2  col3  col4  col5  col6
row1     x
row2     x     x
row3     x     x
row4                 x
row5                 x     x
row6                 x     x
row7                             x
row8                             x     x
row9                             x     x
```

To specify these parameters, you can use some shorthand notation in the MATRIX statement. For example, in the first entry of the MATRIX statement for matrix **F1**, you use the notation `[1,@1]`. This means that the parameter specification starts with the `[1,1]` element and proceeds to the next element while fixing the column number at 1. Hence, parameters x1–x3 are specified for the **F1**[1, 1], **F1**[2, 1], and **F1**[3, 1] elements, respectively. Similarly, you specify other parameters in the **F1** matrix in a column by column fashion.

If you do not use the @ sign in the specification, the parameters are assigned differently. For example, in the specification of the **L** matrix, the first entry in the corresponding MATRIX statement also starts with the `[1,1]` element. But it proceeds down to `[2,2]`, `[3,3]`, and so on because the @ sign is not used to fix any column or row number. As a result, the MATRIX statement for **L** specifies the following pattern:

```
        col1  col2  col3  col4  col5  col6
row1     1
row2           1
row3     1           1
row4           1           1
row5     1           1           1
row6           1           1           1
```

The unspecified elements are fixed zeros in the model.

Similarly, you specify the diagonal matrices **F2**, **F3**, and **P**, and the symmetric matrix **U**.

You also set bounds for some parameters in the BOUNDS statement and some dependent parameters in the SAS programming statements.

Output 26.32.1 shows the correlation structures and the model matrices in the analysis. All appear to be intended.

Output 26.32.1 The Correlation Structures and Model Matrices of the Longitudinal Factor Model

```
                              COSAN Model Structures

  Sigma =     F1*F2*F3*L*inv(F3)*inv(F2)*P*(inv(F2))`*(inv(F3))`*L`*F3`*F2`*
              F1` + U
```

Output 26.32.1 *continued*

Summary of Model Matrices

Matrix	N Row	N Col	Matrix Type
F1	9	6	GEN: Rectangular
F2	6	6	DIA: Diagonal
F3	6	6	DIA: Diagonal
L	6	6	LOW: L Triangular
P	6	6	DIA: Diagonal
U	9	9	SYM: Symmetric

PROC CALIS finds a converged solution for the estimation problem. Output 26.32.2, Output 26.32.3, and Output 26.32.4 show the estimation results of the **F1**, **F2**, and **F3** matrices, respectively.

Output 26.32.2 Estimation of the **F1** Matrix of the Longitudinal Factor Model

Model Matrix F1
(9 x 6 General Rectangular Matrix)

	Col1	Col2	Col3	Col4	Col5	Col6
obs1	0.3515 [x1]	0	0	0	0	0
obs2	0.2871 [x2]	0.9528 [x4]	0	0	0	0
obs3	0.7101 [x3]	0.2059 [x5]	0	0	0	0
obs4	0	0	0.4204 [x6]	0	0	0
obs5	0	0	0.4303 [x7]	0.9027 [x9]	0	0
obs6	0	0	0.8591 [x8]	0.1772 [x10]	0	0
obs7	0	0	0	0	0.3487 [x11]	0
obs8	0	0	0	0	0.5924 [x12]	-0.1971 [x14]
obs9	0	0	0	0	0.9987 [x13]	0.0871 [x15]

Output 26.32.3 Estimation of the $\mathbf{F2}$ Matrix of the Longitudinal Factor Model

Model Matrix F2
(6 x 6 Diagonal Matrix)

	Col1	Col2	Col3	Col4	Col5	Col6
Row1	1.0000	0	0	0	0	0
Row2	0	1.0000	0	0	0	0
Row3	0	0	0.8939 [x16]	0	0	0
Row4	0	0	0	0.5806 [x17]	0	0
Row5	0	0	0	0	0.8939 [x16]	0
Row6	0	0	0	0	0	0.5806 [x17]

Output 26.32.4 Estimation of the $\mathbf{F3}$ Matrix of the Longitudinal Factor Model

Model Matrix F3
(6 x 6 Diagonal Matrix)

	Col1	Col2	Col3	Col4	Col5	Col6
Row1	1.0000	0	0	0	0	0
Row2	0	1.0000	0	0	0	0
Row3	0	0	1.0000	0	0	0
Row4	0	0	0	1.0000	0	0
Row5	0	0	0	0	0.5963 [x18]	0
Row6	0	0	0	0	0	1.0000 [x19]

Output 26.32.5 shows the estimation results of the **L** matrix, which is a fixed matrix that contains only 0 or 1 for its elements.

Output 26.32.5 Estimation of the **L** Matrix of the Longitudinal Factor Model

Model Matrix L
(6 x 6 Lower Triangular Matrix)

	Col1	Col2	Col3	Col4	Col5	Col6
Row1	1.0000	0	0	0	0	0
Row2	0	1.0000	0	0	0	0
Row3	1.0000	0	1.0000	0	0	0
Row4	0	1.0000	0	1.0000	0	0
Row5	1.0000	0	1.0000	0	1.0000	0
Row6	0	1.0000	0	1.0000	0	1.0000

Output 26.32.6 shows the estimation results of the **P** matrix. Notice that parameter estimate x23 falls on the lower boundary at zero.

Output 26.32.6 Estimation of the **P** Matrix of the Longitudinal Factor Model

Model Matrix P
(6 x 6 Diagonal Matrix)

	Col1	Col2	Col3	Col4	Col5	Col6
Row1	1.0000	0	0	0	0	0
Row2	0	1.0000	0	0	0	0
Row3	0	0	0.2010 [x20]	0	0	0
Row4	0	0	0	0.6629 [x21]	0	0
Row5	0	0	0	0	0.6444 [x22]	0
Row6	0	0	0	0	0	0 [x23]

In fact, PROC CALIS routinely checks for zero values for the estimates on the diagonal of the central symmetric matrices. In this case, you get the following messages regarding the estimation of matrix **P**:

```
WARNING: Although all predicted variances for the observed variables are
         positive, the corresponding predicted covariance matrix is not
         positive definite. It has one negative eigenvalue.
WARNING: The estimated variance of variable 6 is essentially zero in the
         central matrix P of term 1 of the COSAN model.
WARNING: The central matrix P of term 1 of the COSAN model is not positive
         definite. It has one zero eigenvalue.
```

Output 26.32.7 shows the estimation results of the **U** matrix. Parameter estimates x28 and x32 fall on the lower boundary at zero. PROC CALIS issues the following messages regarding the estimation of matrix **U**:

```
WARNING: The estimated variance of obs5 is essentially zero in the central
         matrix U of term 2 of the COSAN model.
WARNING: The estimated variance of obs9 is essentially zero in the central
         matrix U of term 2 of the COSAN model.
WARNING: The central matrix U of term 2 of the COSAN model is not positive
         definite. It has 3 negative eigenvalues.
```

Output 26.32.7 Estimation of the **U** Matrix of the Longitudinal Factor Model

Model Matrix U
(9 x 9 Symmetric Matrix)

	obs1	obs2	obs3	obs4	obs5
obs1	0.8764 [x24]	0	0	0.5879 [x33]	0
obs2	0	0.009683 [x25]	0	0	0.1302 [x34]
obs3	0	0	0.4533 [x26]	0	0
obs4	0.5879 [x33]	0	0	0.8233 [x27]	0
obs5	0	0.1302 [x34]	0	0	0 [x28]
obs6	0	0	0.2335 [x35]	0	0
obs7	0.5847 [x39]	0	0	0.6426 [x36]	0
obs8	0	0.7084 [x40]	0	0	0.7259 [x37]
obs9	0	0	0.3215 [x41]	0	0

Output 26.32.7 *continued*

Model Matrix U
(9 x 9 Symmetric Matrix)

	obs6	obs7	obs8	obs9
obs1	0	0.5847 [x39]	0	0
obs2	0	0	0.7084 [x40]	0
obs3	0.2335 [x35]	0	0	0.3215 [x41]
obs4	0	0.6426 [x36]	0	0
obs5	0	0	0.7259 [x37]	0
obs6	0.2305 [x29]	0	0	0.2329 [x38]
obs7	0	0.8784 [x30]	0	0
obs8	0	0	0.6102 [x31]	0
obs9	0.2329 [x38]	0	0	0 [x32]

Because this formulation of Swaminathan's model in general leads to an unidentified problem, the results given here are different from those reported by McDonald (1980). The displayed output of PROC CALIS also indicates that the fitted central model matrices **P** and **U** are not positive-definite. The BOUNDS statement constrains the diagonals of the matrices **P** and **U** to be nonnegative, but this cannot prevent **U** from having three negative eigenvalues. The fact that many of the published results for more complex models in covariance structure analysis are connected to unidentified problems implies that more theoretical work should be done to study the general features of such models.

References

Akaike, H. (1974), "A New Look at Statistical Model Identification," *IEEE Transactions on Automatic Control*, 19, 716–723.

Akaike, H. (1987), "Factor Analysis and AIC," *Psychometrika*, 52, 317–332.

Bartlett, M. S. (1950), "Tests of Significance in Factor Analysis," *British Journal of Psychology*, 3, 77–85.

Bartlett, M. S. (1954), "A Note on Multiplying Factors for Various Chi-squared Approximations," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 16, 296–298.

Beale, E. M. L. (1972), "A Derivation of Conjugate Gradients," in F. A. Lootsma, ed., *Numerical Methods for Nonlinear Optimization*, 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. (1975), *Discrete Multivariate Analysis: Theory and Practice*, Cambridge, MA: 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 D. M. Hawkins, ed., *Topics in Applied Multivariate Analysis*, 72–141, Cambridge: 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 K. A. Bollen and S. Long, eds., *Testing Structural Equation Models*, 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.

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 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.

Duncan, O. D., Haller, A. O., and Portes, A. (1968), "Peer Influences on Aspirations: A Reinterpretation," *American Journal of Sociology*, 74, 119–137.

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, UK: 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.

Haller, A. O. and Butterworth, C. E. (1960), "Peer Influences on Levels of Occupational and Educational Aspiration," *Social Forces*, 38, 289–295.

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.

Huynh, H. and Feldt, L. S. (1970), "Conditions Under Which Mean Square Ratios in Repeated Measurements Designs Have Exact F-Distributions," *Journal of the American Statistical Association*, 65, 1582–1589.

James, L. R., Mulaik, S. A., and Brett, J. M. (1982), *Causal Analysis*, Beverly Hills: 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 A. S. Goldberger and O. D. Duncan, eds., *Structural Equation Models in the Social Sciences*, New York: Academic 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.

Keesling, J. W. (1972), *Maximum Likelihood Approaches to Causal Analysis*, Ph.D. thesis, University of Chicago, Chicago.

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: Macmillan.

Lee, S. Y. (1985), "On Testing Functional Constraints in Structural Equation Models," *Biometrika*, 72, 125–131.

Loehlin, J. C. (1987), *Latent Variable Models*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Loehlin, J. C. (2004), *Latent Variable Models: An Introduction to Factor, Path, and Structural Analysis*, Fourth Edition, Mahway, 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 J. R. Nesselroade and R. B. Cattell, eds., *The Handbook of Multivariate Experimental Psychology*, 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. and Hartmann, W. (1992), “A Procedure for Obtaining Initial Values of Parameters in the RAM Model,” *Multivariate Behavioral Research*, 27, 57–176.

McDonald, R. P. and Marsh, H. W. (1988), “Choosing a Multivariate Model: Noncentrality and Goodness of Fit,” Distributed paper.

Moré, J. J. (1978), “The Levenberg-Marquardt Algorithm: Implementation and Theory,” in G. A. Watson, ed., *Lecture Notes in Mathematics*, volume 30, 105–116, Berlin-Heidelberg-New York: 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.

Morrison, D. F. (1990), *Multivariate Statistical Methods*, Third Edition, New York: McGraw-Hill.

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 - San Francisco - London: Academic Press.

Powell, M. J. D. (1977), “Restart Procedures for the Conjugate Gradient Method,” *Mathematical Programming*, 12, 241–254.

Powell, M. J. D. (1978a), “Algorithms for Nonlinear Constraints That Use Lagrangian Functions,” *Mathematical Programming*, 14, 224–248.

Powell, M. J. D. (1978b), “A Fast Algorithm for Nonlinearly Constrained Optimization Calculations,” in G. A. Watson, ed., *Lecture Notes in Mathematics*, volume 630, 144–175, Berlin-Heidelberg-New York: Springer-Verlag.

Powell, M. J. D. (1982a), "Extensions to Subroutine VF02AD," in R. F. Drenick and F. Kozin, eds., *Systems Modeling and Optimization, Lecture Notes in Control and Information Sciences*, volume 38, 529–538, Berlin-Heidelberg-New York: Springer-Verlag.

Powell, M. J. 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*, Technical 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 D. R. Heise, ed., *Sociological Methodology*, San Francisco: Jossey Bass.

Wiley, D. E. (1973), "The Identification Problem for Structural Equation Models with Unmeasured Variables," in A. S. Goldberger and O. D. Duncan, eds., *Structural Equation Models in the Social Sciences*, New York: Academic Press.

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

D

E

F

G

H

T

U

V

W

Syntax Index

D

E

F

G

H

I

P

R

S

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`**.