THE POWER TO KNOW.

# SAS/ETS® 15.1 User's Guide

# The TMODEL Procedure

This document is an individual chapter from *SAS/ETS® 15.1 User's Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2018. *SAS/ETS® 15.1 User's Guide*. Cary, NC: SAS Institute Inc.

**SAS/ETS® 15.1 User's Guide**



SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

November 2018

# Chapter 39
# The TMODEL Procedure

**Contents**



## Overview: TMODEL Procedure

The TMODEL procedure is a new, experimental version of the MODEL procedure. The code that you use to perform nearly all analyses in PROC MODEL can be used unchanged in PROC TMODEL; however, PROC TMODEL incorporates high-performance computational techniques and offers new features that enhance the functionality of PROC MODEL. For an explanation of the capabilities and operation of both PROC MODEL and PROC TMODEL, see Chapter 24, "The MODEL Procedure."

### Comparison of PROC TMODEL and PROC MODEL

PROC TMODEL includes changes to the underlying computational algorithms that are used in the majority of PROC MODEL analyses. The new algorithms improve the stability and convergence characteristics along

with the computational efficiency for most problems; however, for some problems these improvements can cause PROC TMODEL to produce different results than PROC MODEL. In particular, both estimation and simulation tasks rely on matrix ordering and factorization algorithms that have been enhanced in PROC TMODEL to work more efficiently, especially for large problems. Also, PROC TMODEL processes input data in a different order than PROC MODEL to improve performance, and this can cause some estimation tasks to produce different results.

In addition to performance improvements, PROC TMODEL has the following new features:

- estimation and simulation of models that use panel data by specifying cross-sectional variables in the CROSSSECTION statement
- estimation of models with nonlinear random-effects parameters when cross-sectional variables are identified in the input data
- use of analytic expressions for Hessian matrices in the optimization process for most estimation methods by default
- use of the nonlinear programming (NLP) solver available in SAS/OR software for performing the optimizations during estimation tasks

The ability to specify cross-sectional variables in PROC TMODEL allows for the estimation of dynamic models by using multiple time series. By contrast, PROC MODEL can estimate dynamic models only for data that contain a single time series. Also, PROC TMODEL enhances the modeling capabilities of PROC MODEL by supporting models of the correlations among cross sections through the specification of random-effects parameters.

Models that depend on highly nonlinear parameters can cause the estimation process either to converge slowly or to fail to converge. PROC TMODEL includes two new features that address these problems. In PROC MODEL, a first-order approximation of the model problem's Hessian matrix is used in the parameter search. PROC TMODEL has the option to use exact Hessian matrix values in the parameter search. PROC TMODEL also supports the use of an alternative nonlinear programming solver that improves convergence characteristics for many estimation problems.

PROC TMODEL breaks computationally intensive operations into multiple, concurrent threads to reduce the time it takes to complete many of the estimation and simulation tasks available in PROC MODEL. PROC MODEL performs all calculations sequentially. PROC TMODEL can break calculations up in the following ways:

- multithreading across partitions of the input data set
- multithreading across BY groups
- multithreading across repetitions in Monte Carlo simulations
- multithreading the optimization process in estimations across sets of initial estimates

For estimation tasks that do not involve dynamic models, PROC TMODEL breaks up the observed data into partitions and computes each partition's contribution to the estimation of parameters concurrently. When PROC TMODEL analyzes a model for many BY groups, the BY groups can be analyzed concurrently. In Monte Carlo simulations, the random perturbations of the model variables can be evaluated concurrently in

PROC TMODEL. In problems that involve a numerical optimization, it is sometimes necessary to perform many local optimizations by using separate initial estimates in order to determine a global solution. PROC TMODEL can perform these local optimizations concurrently to find the global solution more quickly than PROC MODEL, which solves local optimization problems sequentially.

## PROC MODEL Features Not Available in PROC TMODEL

The following features in PROC MODEL are not currently available in PROC TMODEL:

- some features in the model file used by the OUTMODEL= and MODEL= options
- BY groups in the SDATA= and ESTDATA= data sets
- covariance matrices in output data sets
- the OUTSUSED= data set
- some diagnostic information in output tables
- Durbin-Watson autocorrelation statistics
- some features in the CMP system, such as the RUN_MACRO function

Model specifications in either PROC MODEL or PROC TMODEL can be saved to a file for use in subsequent PROC MODEL or PROC TMODEL operations; however, there are some limitations to the model files that can be shared between PROC MODEL and PROC TMODEL. The specification of instrumental variables that is saved to a model file in PROC MODEL or PROC TMODEL cannot be used in PROC TMODEL or PROC MODEL estimation tasks, respectively.

Monte Carlo simulations in PROC MODEL can apply a different error covariance matrix, different parameter values, and a different parameter covariance matrix for each BY group in the DATA= data set through the specification of corresponding BY groups in the SDATA= and ESTDATA= data sets. Also, in PROC MODEL estimations, different initial parameter estimates for each BY group can be specified in the ESTDATA= data set. In contrast, PROC TMODEL does not currently support the use of BY groups in either the SDATA= or ESTDATA= data set. All BY groups share the same covariance matrices and parameter value specifications. PROC TMODEL will support BY groups in the SDATA= and ESTDATA= data sets in a future release.

PROC MODEL adds perturbations to parameter values during Monte Carlo simulations when a parameter covariance matrix is specified in the ESTDATA= data set. PROC TMODEL does not currently support perturbation of parameters, but this functionality will be available in a future release.

PROC MODEL stores both parameter estimates and parameter covariance matrices in the OUTEST= data set when the OUTCOV option is specified. PROC TMODEL does not support the output of covariance matrices in the OUTEST= data set; however, the COV option, which prints parameter covariance matrices, is supported.

Many tables that PROC MODEL produces are not available in PROC TMODEL because they fall into one or more of the following categories:

| | |
|---|---|
| Not available | report the intermediate states of PROC MODEL calculations that are not available or are computed differently in PROC TMODEL. |
| Replaced | are replaced by equivalent diagnostic output in PROC TMODEL in a different format. |

Future are not yet available in PROC TMODEL but are planned for a future release.

The reason that each table is not available in PROC TMODEL is summarized in Table 39.1.

**Table 39.1** ODS Tables Not Available in PROC TMODEL

| ODS Table Name | Description | Reason[1] |
|---|---|---|
| **ODS Tables Created by the FIT Statement** | | |
| AugGMMCovariance | Crossproducts matrix | F |
| ConfInterval | Profile likelihood confidence intervals | F |
| Crossproducts | Crossproducts matrix | F |
| DatasetOptions | Data sets used | F |
| DetResidCov | Determinant of the residuals | F |
| DWTest | Durbin-Watson test | F |
| EstSummaryMiss | Summary statistics for PAIRWISE option | F |
| GMMCovariance | Crossproducts matrix | F |
| GMMTestStats | GMM test statistics | F |
| Godfrey | Godfrey's serial correlation test | F |
| HausmanTest | Hausman's test table | F |
| InvXPXMat | X′X inverse for system | N |
| IterInfo | Iteration printing | N R |
| ObsSummary | Identifies observations that contain errors | R |
| ObsUsed | Observations read, used, and missing | R |
| ParmChange | Parameter change vector | N |
| SizeInfo | Storage requirement for estimation | F |
| XPXMat | X′X for system | N |
| YkVector | Marquardt iteration vector | N |
| **ODS Tables Created by the SOLVE Statement** | | |
| DatasetOptions | Data sets used | F |
| DescriptiveStatistics | Descriptive statistics | F |
| FitStatistics | Fit statistics for simulation | F |
| ObsSummary | Simulation trace output | R |
| ObsUsed | Observations read, used, and missing | R |
| SolutionVarList | Solution variable lists | F |
| TheilRelStats | Theil relative change error statistics | F |
| TheilStats | Theil forecast error statistics | F |
| ErrorVec | Iteration error vector | N |
| ResidualValues | Iteration residual values | N |
| PredictedValues | Iteration predicted values | N |
| SolutionValues | Iteration solved for variable values | N |
| **ODS Tables Created by the FIT and SOLVE Statements** | | |
| AdjacencyMatrix | Adjacency graph | R |
| CodeDependency | Variable cross reference | N |
| CodeList | Listing of compiled program code | N |
| CrossReference | Cross-reference listing for program | N |

[1] N - Not available, R - Replaced, F - Future

**Table 39.1** *continued*

| ODS Table Name | Description | Reason[1] |
|---|---|---|
| DepStructure | Dependency structure of the system | N |
| FirstDerivatives | First derivative table | N |
| IterIntg | Integration iteration output | N |
| MemUsage | Memory usage statistics | F |
| MissingDependencies | Missing values by dependency | F |
| MissingObservations | Missing values by observation | F |
| MissingSymbols | Missing values by symbol | F |
| ParmReadIn | Parameter estimates read in | F |
| SortAdjacencyMatrix | Sorted adjacency graph | R |
| TransitiveClosure | Transitive closure graph | R |

[1]N - Not available, R - Replaced, F - Future

You can find information in PROC TMODEL for the PROC MODEL tables in category R as follows:

- IterInfo displays iteration information for the optimization process in the log for the ORMP optimizer.
- ObsSummary displays diagnostic information for observations that produce missing values in the log.
- ObsUsed displays observation counts in the EstSummaryStats table.
- AdjacencyMatrix, SortAdjacencyMatrix, TransitiveClosure have dependency information that you display by using the ANALYZEDEPS= option in the SOLVE statement.

## Getting Started: TMODEL Procedure

One of the most powerful enhancements in PROC TMODEL compared to PROC MODEL is the ability to reduce the time required to perform computationally intensive tasks, such as estimating parameters in a nonlinear ordinary differential equation (ODE) model. This estimation task requires both the numerical integration of derivative variables in the model over time steps and the repeated evaluation of the time steps during the estimation's minimization process. The following example estimates the parameters in a system of two coupled differential equations by using simulated data for four time series:

```
data soln;
   keep exprun t x y;
   length exprun $ 8;
   array experno[4] $ _temporary_ ( "one" "two" "three" "four" );
   call streaminit (1);
   do i = 1 to 4;
      exprun = experno[i];
      do t = 0 to 5 by 0.1;
         /* analytic solution for the ODE system */
         x = 1/2*(exp(-3*t) - exp(-t)) + rand('normal',0,0.01);
```

```
         y = 1/2*(exp(-3*t) + exp(-t)) + rand('normal',0,0.01);
         output;
      end;
   end;
run;

proc model outmodel=ode;
   endo x y;
   parms a b;
   g = exp (x + y);
   dert.x = -a*x - log (g);
   dert.y = -b*y - log (g);
quit;

proc model data=soln model=ode;
   fit / time=t dynamic;
quit;

proc tmodel data=soln model=ode;
   crosssection exprun;
   fit / time=t dynamic;
quit;
```

The model file Work.ODE is created and used in this example to avoid redundant specification of the model program. Although the ODE model and data are identical in the PROC MODEL and PROC TMODEL steps, you must specify the CROSSSECTION statement to take advantage of the multithreading capabilities of PROC TMODEL. Figure 39.1 shows how much faster PROC TMODEL performs this estimation by integrating the model over each of the four time series concurrently. Real time measures how long it takes to execute the PROC step, and CPU time is a measure of the computing resources that the PROC step consumes.

**Figure 39.1** Performance Comparison of PROC MODEL and PROC TMODEL

```
NOTE: PROCEDURE MODEL used (Total process time):
      real time           2.73 seconds
      cpu time            2.66 seconds
NOTE: PROCEDURE TMODEL used (Total process time):
      real time           2.47 seconds
      cpu time            6.26 seconds
```

# Syntax: TMODEL Procedure

The following statements are available in PROC TMODEL:

**PROC TMODEL** *options* **;**
**ARRAY** *arrayname variable-list . . .* **;**
**ATTRIB** *variable-list1 attribute-list1* < *variable-list2 attribute-list2 . . .* > **;**
**BOUNDS** *bound1* < *, bound2 . . .* > **;**
**BY** *variable-list* **;**
**CALL** *name* **;**
**CALL** *name*(*expression1* < *, expression2 . . .* > ) **;**
**CONTROL** *variable* < *value* > *. . .* **;**
**CROSSSECTION** *variable-list* **;**
**DO** **;**
**DO** *variable = expression* < **TO** *expression* > < **BY** *expression* > < *, expression* **TO** *expression* <
**BY** *expression* > *. . .* > < **WHILE** *expression* > < **UNTIL** *expression* > **;**
**END** **;**
**DROP** *variable . . .* **;**
**ENDOGENOUS** *variable* < *initial-values* > *. . .* **;**
**ERRORMODEL** *equation-name* ∼ *distribution* < **CDF=**(*CDF*(*options* )) > **;**
**ESTIMATE** *item1* < *, item2 . . .* > < *,/ options* > **;**
**EXOGENOUS** *variable* < *initial values* > *. . .* **;**
**FIT** *equations* < **PARMS=**(*parameter values . . .* ) > < **START=**(*parameter values . . .* ) >
< **DROP=**(*parameters* ) > < */ options* > **;**
**FORMAT** *variable-list* < *format* > < **DEFAULT=** *default-format* > **;**
**GOTO** *statement-label* **;**
**ID** *variable-list* **;**
**IF** *expression* **;**
**IF** *expression* **THEN** *programming-statement1 ;* < **ELSE** *programming-statement2* > **;**
**variable** *= expression* **;**
**variable** *+ expression* **;**
**INCLUDE** *model-file . . .* **;**
**INSTRUMENTS** < *instruments* > < **_EXOG_** > < **EXCLUDE=**(*parameters* ) > < */ options* > **;**
**KEEP** *variable . . .* **;**
**LABEL** *variable =*'*label*' *. . .* **;**
**LENGTH** *variable-list* < **$** > *length . . .* < **DEFAULT=***length* > **;**
**LINK** *statement-label* **;**
**MOMENT** *variable-list = moment-specification . . .* **;**
**OUTVARS** *variable . . .* **;**
**PARAMETERS** *variable1* < *value1* > < *variable2* < *value2 . . .* > > **;**
**PERFORMANCE** < **NTHREADS=** *n* > < **BYPRIORITY=** *priority* > < **REPPRIORITY=** *priority* > <
**MSPRIORITY** | **GRIDPRIORITY=** *priority* > < **PARTPRIORITY=** *priority* > **;**
**PUT** *print-item . . .* < *@* > < *@@* > **;**
**RANDOM** *random-effects* ∼ *distribution* < *options* > **;**
**RANGE** *variable* < *= first* > < **TO** *last* > **;**

**RENAME** *old-name1 = new-name1 < . . . old-name2 = new-name2 >* **;**
**RESET** *options* **;**
**RESTRICT** *restriction1 < , restriction2 . . . >* **;**
**RETAIN** *variable-list1 value1 < variable-list2 value2 . . . >* **;**
**RETURN** **;**
**SOLVE** *variable-list* **<SATISFY=(***equations***)** *> < / options >* **;**
**SUBSTR (***variable, index, length* **)=** *expression* **;**
**SELECT** *<* **(***expression* **)>** **;**
**OTHERWISE** *programming-statement* **;**
**TEST** *< "name" > test1 <, test2 . . . > < ,/ options >* **;**
**VAR** *variable < initial-values > . . .* **;**
**WEIGHT** *variable* **;**
**WHEN** **(***expression* **)***programming-statement* **;**

The following sections describe statements that are available in PROC TMODEL but not in PROC MODEL or statements that have options in PROC TMODEL that are not available in PROC MODEL. For information about all other statements in the PROC TMODEL syntax, see Chapter 24, "The MODEL Procedure."

## PROC TMODEL Statement

**PROC TMODEL** *options* **;**

The PROC TMODEL statement invokes the TMODEL procedure. The *options* that you can specify in the PROC TMODEL statement are the same as those you can specify in the PROC MODEL statement. For more information, see the section "PROC MODEL Statement" on page 1448 in Chapter 24, "The MODEL Procedure."

In addition to all the *options* described in PROC MODEL, you can specify the following *options*. These *options* can also be specified in a FIT statement or a SOLVE statement.

### Options to Control the Solution of a System of Linear Equations

**LUSOLVER=NUMPIVOT | STRUCTPIVOT | OLD |** *n*

specifies the method to use to factor and solve systems of linear equations. These linear systems are encountered when the FIML option (which requests that the FIML method be used to estimate models) is specified in the FIT statement or the PROC TMODEL statement. These linear systems are also encountered when either the NEWTON or OPTIMIZE option is specified in the SOLVE statement (these options specify the numerical solution method). Also, model programs that include differential equations require the solution of linear systems of equations. You can specify the following values:

| | |
|---|---|
| *n* | considers numerical values of matrix elements only every *n*th time a lower triangular times upper triangular (LU) matrix factorization is computed. |
| **NUMPIVOT** | considers numerical values of matrix elements each time an LU factorization is computed. |
| **OLD** | uses the factorization and solution algorithm that the MODEL procedure uses. |
| **STRUCTPIVOT** | considers only the structure of nonzero elements in the linear system for ordering equations in the LU factorization. |

By default, LUSOLVER=NUMPIVOT.

## CROSSSECTION Statement

**CROSSSECTION** *variables* **;**

The CROSSSECTION statement specifies the variables that identify observations in the input data set that form time series. This statement is useful when the input data set contains more than one time series and when a dynamic model is being estimated using the FIT statement or simulated using the SOLVE statement. When you use the CROSSSECTION statement, observations within each cross section must be grouped together, and the observations in each cross section must be sorted.

## FIT Statement

**FIT** < *equations* > < **PARMS=(***parameter* < *values* > ... **)** > < **START=(***parameter values* ... **)** > < **DROP=(***parameter* ... **)** > < **INITIAL=(***variable* < = *parameter* | *constant* > ... **)** > < / *options* > **;**

PROC TMODEL includes additional options in the FIT statement to provide more control of the estimation process than PROC MODEL provides. For a complete description of the syntax of the FIT statement, see the section "FIT Statement" on page 1464 in Chapter 24, "The MODEL Procedure." The syntax for specifying the new options follows:

**FIT** < ... > < / **QUADHESS=LINEAR** | **ANALYTIC** | **FDA** < **OPTIMIZER=***type* < **(***ORMP-optimizer-options***)** > > > **;**

### Options to Control the Estimation Process

**QUADHESS=LINEAR** | **ANALYTIC** | **FDA** (Experimental )
specifies which method to use to compute the Hessian matrix during the optimization process. For FIML and *t* distribution estimations, the HESSIAN= option is used to specify how the Hessian matrix is computed, and the QUADHESS= option has no effect.

**ANALYTIC** uses the exact analytical representation of the Hessian matrix during the optimization process. This option might improve convergence properties for certain nonlinear models. It is not available for feasible GLS estimations or random-effects estimations.

**FDA** uses a finite difference approximation to the Hessian matrix during the optimization process. This option is available only when the OPTIMIZER=ORMP option is specified.

**LINEAR** uses the crossproduct of the Jacobian matrix as an approximation to the Hessian matrix during the optimization process.

By default, QUADHESS=LINEAR.

## Options to Control the Optimization Process

The following options control the optimization process. For more information about the optimizers available in PROC TMODEL, see the section "Nonlinear Optimization" on page 2806.

**OPTIMIZER=***type***<(***ORMP-optimizer-options***)>**
specifies which optimizer to use to perform the numerical minimization. You can specify only one of the following *types*. By default, OPTIMIZER=ORMP.

**ZOPT** uses the optimizer that is used in PROC MODEL in the minimization.

You can also specify the following *options* after the slash in the FIT statement: CONVERGE=, MAXITER=, MAXSUBITER=, METHOD=.

**ORMP** uses the nonlinear programming solver available in SAS/OR software in the minimization. For more information about the ORMP nonlinear solver, see Chapter 11, "The Nonlinear Programming Solver" (*SAS/OR User's Guide: Mathematical Programming*).

You can specify the following *ORMP-optimizer-options*:

**ALGORITHM=ACTIVESET | CONCURRENT | INTERIORPOINT**
specifies the optimization technique to use to solve the problem. By default, ALGORITHM=INTERIORPOINT.

**FEASTOL=**$\epsilon$
defines the feasible tolerance. By default, FEASTOL=1E–6.

**MAXITER=***n*
requests that the NLP solver take at most *n* major iterations to determine an optimum. By default, MAXITER=5,000.

**MAXTIME=***t*
specifies an upper limit of *t* units of time for the optimization process. If you do not specify this option, the NLP solver does not stop because of time elapsed.

**MSBNDRANGE=***m*
defines the range from which each variable can take values during the sampling process. By default, MSBNDRANG=200.

**MSDISTTOL=**$\epsilon$
specifies the tolerance by which two optimal points are considered distinct. Optimal points are considered distinct if the Euclidean distance between them is at least $\epsilon$. By default, MSDISTTOL=1.0E–6.

**MSMAXSTARTS=***n*
specifies the maximum number of starting points to use for local optimization. By default, MSMAXSTARTS=100.

**MSMAXTIME=***t*
specifies the maximum time *t* (in seconds) for the NLP solver to locate the best local optimum in multistart mode. If you do not specify this option, the multistart algorithm does not stop because of the amount of time elapsed.

**MULTISTART**
enables multistart mode. In this mode, the local solver solves the problem from multiple starting points, possibly finding a better local minimum as a result. By default, this option is disabled.

**OBJLIMIT=***m*
specifies a limit on the magnitude of the objective value. The algorithm terminates when the objective value becomes less than $-m$. The minimum acceptable value of $m$ is 1E+8. If the specified value of $m$ is less than 1E+8, the value is reset to the default value. By default, OBJLIMIT=1E+20.

**OPTTOL=**$\epsilon$
defines the measure by which the ORMP optimizer decides whether the current iterate is an acceptable approximation of a local minimum, where $\epsilon$ is a positive real number. The ORMP optimizer determines that the current iterate is a local minimum when the norm of the scaled vector of the optimality conditions is less than $\epsilon$ and the true constraint violation is less than the value of the FEASTOL= option. By default OPTTOL=1E–6.

**SEED=***n*
specifies a positive integer to use as the seed for generating random number sequences in multistart mode. To ensure reproducible results, specify a nonzero value. By default, SEED=0.

**TIMETYPE=CPU | REAL**
specifies the units of time that the MAXTIME= option uses. If you do not specify this option, the multistart algorithm does not stop because of the amount of time elapsed.

## Options to Control the Solution of a System of Linear Equations

**LUSOLVER=NUMPIVOT | STRUCTPIVOT | OLD |** *n*
specifies the method to use to factor and solve systems of linear equations. These linear systems are encountered when the FIML option (which requests that the FIML method be used to estimate models) is specified in the FIT statement or the PROC TMODEL statement. These linear systems are also encountered when either the NEWTON or OPTIMIZE option is specified in the SOLVE statement (these options specify the numerical solution method). Also, model programs that include differential equations require the solution of linear systems of equations. You can specify the following values:

| | |
|---|---|
| *n* | considers numerical values of matrix elements only every *n*th time a lower triangular times upper triangular (LU) matrix factorization is computed. |
| **NUMPIVOT** | considers numerical values of matrix elements each time an LU factorization is computed. |
| **OLD** | uses the factorization and solution algorithm that the MODEL procedure uses. |
| **STRUCTPIVOT** | considers only the structure of nonzero elements in the linear system for ordering equations in the LU factorization. |

By default, LUSOLVER=NUMPIVOT.

## PERFORMANCE Statement

**PERFORMANCE** < **NTHREADS=***n* > < **BYPRIORITY=***priority* > < **REPPRIORITY=***priority* >
< **MSPRIORITY** | **GRIDPRIORITY=***priority* > < **PARTPRIORITY=***priority* > **;**

The PERFORMANCE statement controls how an estimation or simulation task in PROC TMODEL uses multiple execution threads. You can specify two types of information in a PERFORMANCE statement: the *number* of threads and the *priority* of calculations to which the threads are assigned. Calculations with a higher priority are allocated a greater number of threads, and calculations with a lower priority are allocated fewer threads. When a calculation is assigned a priority of zero, it is executed in one thread. When priority options are not specified, PROC TMODEL assigns default priority values based on properties of the model program and data.

Each PERFORMANCE statement is associated with the FIT or SOLVE statement that precedes it. When there is no preceding FIT or SOLVE statement, the PERFORMANCE statement is associated with the FIT or SOLVE statement that follows it.

The following options apply to both estimation and simulation tasks:

**BYPRIORITY=***priority*
specifies the priority for allocating the computation threads to process BY groups concurrently in the input data set. The value of *priority* must be between 0 and 1, where 0 specifies the lowest priority and 1 specifies the highest priority.

**CPUCOUNT=***n*

**NTHREADS=***n*
specifies the approximate number of concurrent computation threads to use. By default, the global CPUCOUNT= option is used to specify the number of threads. The actual number of threads that are used might vary from the value that you specify in the CPUCOUNT= or NTHREADS= option based on the properties of the model program, the properties of the input data set, and the priority options specified in the PERFORMANCE statement.

### Options to Configure Estimation Threads

**MSPRIORITY=***priority*

**GRIDPRIORITY=***priority*
specifies the priority for allocating computation threads for the concurrent execution of the optimizer during the estimation process. Concurrent execution of the optimizer is possible when the OPTIMIZER=ORMP(MULTISTART) option or the START= option is specified in the FIT statement. The value of *priority* must be between 0 and 1, where 0 specifies the lowest priority and 1 specifies the highest priority.

**PARTPRIORITY=***priority*
specifies the priority for allocating computation threads for concurrent execution across partitions of the input data set. The value of *priority* must be between 0 and 1, where 0 specifies the lowest priority and 1 specifies the highest priority.

### Option to Configure Simulation Threads

**REPPRIORITY=***priority*

specifies the priority for allocating computation threads for the concurrent execution of repetitions of the input data set when you are performing Monte Carlo simulations. The value of *priority* must be between 0 and 1, where 0 specifies the lowest priority and 1 specifies the highest priority.

For more information about multithreading in PROC TMODEL, see the section "Multithreaded Calculations" on page 2808.

## RANDOM Statement (Experimental)

**RANDOM** *random-effects* ~ *distribution* < *options* > **;**

The RANDOM statement specifies which parameters in the model program are random effects and defines their distribution. The statement consists of a list of the random effects, a tilde (~), and then the distribution of the random effects. The RANDOM statement must also be accompanied by a CROSSSECTION statement, which specifies the subject variables. Only one RANDOM statement can be associated with each FIT statement.

The only distribution available for the random effects is normal($m$,$v$), with mean $m$ and variance $v$. This syntax is illustrated as follows for one effect:

```
random u ~ normal(0,s2u);
```

For multiple effects, you should specify bracketed vectors for $m$ and $v$, the latter consisting of the lower triangular elements of the random-effects variance matrix listed in row order. This is illustrated for two random effects as follows:

```
random b1 b2 ~ normal([0,0],[g11,g21,g22]);
```

Similarly, the syntax for three random effects is illustrated as follows:

```
random b1 b2 b3 ~ normal([0,0,0],[g11,g21,g22,g31,g32,g33]);
```

The variables that you specify in the CROSSSECTION statement determine the unique realizations of the random effects. The observations for each value of the CROSSSECTION variables must be grouped together in the input data set. PROC TMODEL processes the input data set sequentially and considers an observation to be from a new cross section whenever the values of the CROSSSECTION variables change from the previous observation.

You can specify the following *options* in the RANDOM statement:

**EBESOPT**

specifies that the empirical Bayes estimates of the random effects that are computed for each value of the CROSSSECTION variables during the optimization process be computed by performing a nonlinear optimization. When you specify this option, the optimizer that you specify in the OPTIMIZER= option in the FIT statement is used to compute the empirical Bayes estimates. By default, a Newton search algorithm computes the empirical Bayes estimates.

**NOPSD**
specifies that the covariance matrix of random effects not be constrained to be positive semidefinite. This option might improve convergence properties for certain parameterizations of the random-effects covariance matrix.

**NUMQUADPTS=***n*
specifies the number of quadrature points to use in the adaptive Gaussian quadrature approximation of the likelihood function. Each random effect is evaluated at *n* points during the approximation of the likelihood function, so if there are *r* random effects, the likelihood function is evaluated at $n^r$ points. By default, NUMQUADPTS=1.

**PSD**
specifies that the covariance matrix of random effects be constrained to be positive semidefinite. PSD is the default.

## SOLVE Statement

**SOLVE** *variables* < **SATISFY=** *equations* > < */options* > **;**

PROC TMODEL includes additional *options* in the SOLVE statement to provide more control of the solution process than PROC MODEL provides. For a complete description of the syntax of the SOLVE statement, see the section "SOLVE Statement" on page 1480 in Chapter 24, "The MODEL Procedure."

You can specify the following *options* after the slash.

### Options to Control the Solution of a System of Linear Equations

**LUSOLVER=NUMPIVOT | STRUCTPIVOT | OLD |** *n*
specifies the method to use to factor and solve systems of linear equations. These linear systems are encountered when the FIML option (which requests that the FIML method be used to estimate models) is specified in the FIT statement or the PROC TMODEL statement. These linear systems are also encountered when either the NEWTON or OPTIMIZE option is specified in the SOLVE statement (these options specify the numerical solution method). Also, model programs that include differential equations require the solution of linear systems of equations. You can specify the following values:

| | |
|---|---|
| *n* | considers numerical values of matrix elements only every *n*th time a lower triangular times upper triangular (LU) matrix factorization is computed. |
| **NUMPIVOT** | considers numerical values of matrix elements each time an LU factorization is computed. |
| **OLD** | uses the factorization and solution algorithm that the MODEL procedure uses. |
| **STRUCTPIVOT** | considers only the structure of nonzero elements in the linear system for ordering equations in the LU factorization. |

By default, LUSOLVER=NUMPIVOT.

# Details: TMODEL Procedure

## Panel Data

Panel data, also known as longitudinal data, consist of observations made over time for multiple subjects or cross sections. Data that are recorded in this form are often used to analyze dynamic models, which use past information to model the relationships among variables. In PROC TMODEL, you can analyze dynamic models that use panel data by identifying the cross-sectional variables in a CROSSSECTION statement. The following example illustrates how to use the CROSSSECTION statement to estimate an autoregressive model that has one parameter shared among five time series:

```
data d;
   length cs $ 8;
   array csname{5} $ _temporary_ ( 'first' 'second' 'third' 'fourth' 'fifth' );
   call streaminit (1);
   do pp = 1 to dim(csname);
      lagx = 0;
      do t = 1 to 10;
         x = 0.8*lagx + rand('normal');
         lagx = x;
         cs = csname[pp];
         output;
      end;
   end;
run;

proc tmodel data=d;
   endo x;
   crosssection cs;
   parms p;

   x = p*lag(x);
   fit;
quit;
```

In this example, PROC TMODEL skips the first observation in each time series because the first lag of $x$ is not available. Figure 39.2 shows that only 45 of the 50 observations contribute to the estimation, because the first observation in each series is skipped.

**Figure 39.2** Observation Counts in a Dynamic Model Estimation

**The TMODEL Procedure**

| Number of Observations | | Statistics for System | |
|---|---|---|---|
| Used | 45 | Objective | 0.9694 |
| Missing | 0 | Objective*N | 43.623 |

Panel data are also treated differently from ordinary observational data for the purpose of multithreading. When no cross-sectional variables are specified, the observations in the DATA= data set are partitioned in

a round-robin fashion among computational threads. Figure 39.3 shows how observations are partitioned among threads in the presence and absence of a cross-sectional variable specification.

**Figure 39.3** Data Partitioning Strategies in a Multithreaded Environment

Original Data Set

| OBS | CS | TIME |
|---|---|---|
| 1 | A | 1 |
| 2 | A | 2 |
| 3 | A | 3 |
| 4 | A | 4 |
| 5 | B | 1 |
| 6 | B | 2 |
| 7 | B | 3 |
| 8 | B | 4 |
| 9 | B | 5 |
| 10 | B | 6 |

Round-Robin Partitioning

| Thread #1 | | | Thread #2 | | | Thread #3 | | | Thread #4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| OBS | CS | TIME | OBS | CS | TIME | OBS | CS | TIME | OBS | CS | TIME |
| 1 | A | 1 | 2 | A | 2 | 3 | A | 3 | 4 | A | 4 |
| 5 | B | 1 | 6 | B | 2 | 7 | B | 3 | 8 | B | 4 |
| 9 | B | 5 | 10 | B | 6 | | | | | | |

Partitioning with CROSSSECTION Statement

| Thread #1 | | | Thread #2 | | | Thread #3 | | | Thread #4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| OBS | CS | TIME | OBS | CS | TIME | OBS | CS | TIME | OBS | CS | TIME |
| 1 | A | 1 | 5 | B | 1 | | | | | | |
| 2 | A | 2 | 6 | B | 2 | | | | | | |
| 3 | A | 3 | 7 | B | 3 | | | | | | |
| 4 | A | 4 | 8 | B | 4 | | | | | | |
| | | | 9 | B | 5 | | | | | | |
| | | | 10 | B | 6 | | | | | | |

In the case where no cross sectional-variables are specified and a dynamic model with lag terms is being analyzed, all the observations are processed sequentially in a single thread.

Another use for panel data is in random-effects models, which model the variation among subjects in the data. In these models, the grouping of observations into subjects is achieved by specifying cross-sectional variables in the same manner as in the other panel data applications in PROC TMODEL.

## Random-Effects Models (Experimental)

The general nonlinear model that is estimated by PROC MODEL and PROC TMODEL can be extended to accommodate observations on $M$ subjects as follows,

$$\mathbf{q}(\mathbf{y}_{it}, \mathbf{x}_{it}, \boldsymbol{\phi}, \mathbf{u}_i) = \boldsymbol{\epsilon}_{it}, \qquad i = 1, \ldots, M \quad t = 1, \ldots, M_i$$

where $\mathbf{q} \in R^g$ is a real-vector-valued function of $\mathbf{y}_{it} \in R^g$, $\mathbf{x}_{it} \in R^l$, $\boldsymbol{\phi} \in R^p$, and $\mathbf{u}_i \in R^r$, where $g$ is the number of equations, $l$ is the number of exogenous variables (lagged endogenous variables are considered exogenous here), $p$ is the number of fixed parameters, $r$ is the number of random effects, $M$ is the number of subjects, and $M_i$ is the number of observations on the $i$th subject. The random effects, $\mathbf{u}_i$, that are associated with the $i$th subject are distributed as follows,

$$\mathbf{u}_i \sim N(\boldsymbol{\mu}, \mathbf{D})$$
$$\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{\xi})$$
$$\mathbf{D} = \mathbf{D}(\boldsymbol{\xi})$$

where $\boldsymbol{\mu}$ is an $r \times 1$ vector of means of the random effects, $\mathbf{D}$ is an $r \times r$ covariance matrix of the random effects, and $\boldsymbol{\xi}$ is a vector of parameters of the random-effects distribution.

The vector of unknown parameters for the random-effects models is $\boldsymbol{\theta} = [\boldsymbol{\phi}, \boldsymbol{\xi}]$. PROC TMODEL performs a maximum likelihood estimation of $\boldsymbol{\theta}$ and the covariance of $\boldsymbol{\theta}$ by using the following marginal distribution of $\mathbf{y}$ based on the joint distribution of $\mathbf{y}$ and the random effects, $\mathbf{u}_i$,

$$p(\boldsymbol{\theta}) = \prod_{i=1}^{M} \int p(\mathbf{y}_i | \mathbf{x}_i, \boldsymbol{\phi}, \mathbf{u}_i)\, p(\mathbf{u}_i | \boldsymbol{\xi})\, d\mathbf{u}_i$$

where $p(\mathbf{y}_i | \mathbf{x}_i, \boldsymbol{\phi}, \mathbf{u}_i)$ is the conditional distribution of $\mathbf{y}$ for the $i$th subject and $p(\mathbf{u}_i | \boldsymbol{\xi})$ is the conditional distribution of the random effects.

## Random-Effects Estimation

Estimating the maximum likelihood values of $\boldsymbol{\theta}$ requires computation of an integral over the random effects. PROC TMODEL approximates this integral by using the adaptive Gaussian quadrature method described in Pinheiro and Bates (1995).

PROC TMODEL minimizes the following negative log-likelihood function to estimate the parameters in random-effects models,

$$-\log p(\boldsymbol{\theta}) \approx \sum_{i=1}^{M} \left( \frac{1}{2} \log |\mathbf{G}_i| - \log \sum_{q=1}^{Q} e^{\mathrm{obj}_{i,q}} \right) + \frac{M}{2} \left( \log |\mathbf{D}| + r \log 2\pi \right)$$

where

$$\begin{aligned}
\mathrm{obj}_{i,q} &= -\frac{1}{2} \sum_{j=1}^{M_i} \mathbf{q}_j'(\mathbf{v}_q) \mathbf{H}^{-1} \mathbf{q}_j(\mathbf{v}_q) + \mathbf{z}_q' \mathbf{z}_q + \sum_{k=1}^{r} \log w_k - \frac{1}{2} \mathbf{v}_q' \mathbf{D}^{-1} \mathbf{v}_q \\
\mathbf{H} &= \mathrm{diag}(h_1, \ldots, h_g) \\
\mathbf{v}_q &= \hat{\mathbf{u}}_i + \sqrt{2} \mathbf{G}_i^{-1/2} \mathbf{z}_q \\
\mathbf{G}_i &= -\nabla^2_{\mathbf{u}_i} \mathrm{obj}_i + \mathbf{D}^{-1}
\end{aligned}$$

where $Q$ is the number of quadrature points used to approximate the integral, $M_i$ is the number of observations on the $i$th subject, $\mathrm{obj}_i$ is the $i$th-subject-specific objective function evaluated at the empirical Bayes estimate of the random effects, $\mathrm{obj}_{i,q}$ is the objective function evaluated at the $q$th quadrature point, $\mathbf{q}_j$ is the vector of equation residuals, $h_i$ is the variance specified for the $i$th equation, $\mathbf{z}_q$ is the $q$th quadrature point, $w_k$ is the weight for the $k$th element of the quadrature point vector, $\mathbf{v}_q$ is the random-effects vector evaluated at the $q$th quadrature point, $\hat{\mathbf{u}}_i$ is the empirical Bayes estimate of the random effects for the $i$th subject, $\mathbf{G}_i$ is the random-effects scale matrix for the $i$th subject, and $\nabla^2_{\mathbf{u}_i} \mathrm{obj}_i$ is the Hessian of the empirical Bayes estimate of the random effects for the $i$th subject. The gradient of the negative log-likelihood function is computed analytically, and the Hessian is computed numerically.

This approach is also used by PROC NLMIXED. For more information about the implementation of the adaptive Gaussian quadrature method, see Chapter 86, "The NLMIXED Procedure" (*SAS/STAT User's Guide*).

The estimation of random-effects models in PROC TMODEL differs from that in PROC NLMIXED in the following ways:

- PROC TMODEL supports only the adaptive Gaussian quadrature method to compute the integral over random effects.
- PROC TMODEL does not compute the number of quadrature points adaptively.
- PROC TMODEL does not support models that contain variance parameters.
- PROC TMODEL constrains the covariance matrix of the random effects to be positive definite by imposing the nonlinear constraint $|\mathbf{D}| > 0$ in the optimization.
- PROC TMODEL does not support hierarchical random effects.
- PROC TMODEL supports models that have more than one endogenous variable.

## Nonlinear Optimization

PROC TMODEL provides two numerical optimization systems, ZOPT and ORMP, to minimize the objective function associated with each of the available estimation methods. The ZOPT system is the same optimization system that PROC MODEL uses, and the nonlinear programming solver, ORMP, is the same optimization system that PROC OPTMODEL uses. The following sections summarize how both optimization systems address issues particular to the problem of estimating model parameters in PROC TMODEL.

### Nonlinear Objective Function

The nonlinear dependence of model programs on parameters complicates the optimization process because it can cause the objective function to become a less predictable function of the parameters. The ZOPT optimizer provides the Gauss and Marquardt minimization methods to manage this nonlinear dependency during the numerical search for a minimum. The Gauss method implements a line search during the search process, and the Marquardt method improves the conditioning of the search for an optimum. The ORMP optimizer uses similar techniques to address nonlinearity and ill-conditioning of the minimization problem. In the ORMP optimizer, these techniques are implemented in a hybrid trust region and line-search algorithm.

### Constraints on Parameters

Another difficulty occurs during the optimization process when constraints are placed on the parameters. In PROC TMODEL, linear and nonlinear constraints can be introduced through the use of the BOUNDS, RESTRICT, and TEST statements. The ZOPT optimizer handles constraints in the minimization by using an active set algorithm to keep track of and enforce constraints. The ORMP optimizer provides two algorithms to handle constraints, an active set algorithm and an interior point algorithm. The active set algorithm manages constraints during the optimization, which is similar to the approach used by the ZOPT optimizer. The interior point algorithm imposes constraints by using barrier functions.

### Multiple Local Minima

Occasionally, characteristics of the data or model program can cause there to be more than one local minimum in the minimization problem. In such cases the optimization process must choose the best minimum from among multiple local minima. In PROC TMODEL, you can specify a grid of initial parameter estimates by using the START= option in the FIT statement, and PROC TMODEL solves the minimization problem by

using each point in the grid as an initial estimate. The grid point optimization that converges to the smallest minimum is then selected as the global minimum. Either the ZOPT system or the ORMP system can be used in the grid search approach to solving the global minimization problem.

The ORMP system also supports a multistart algorithm for finding the global minimum. The multistart algorithm chooses the best global minimum from among many local minima, as in the grid search approach; however, the multistart algorithm does not require you to specify the initial grid point estimates.

### Choosing an Optimizer

For most problems, there is no need to choose between the ZOPT and ORMP optimizers, because they both converge quickly to the same optimum. However, in cases where the optimizers yield different results, the following general guidelines can help you choose which optimizer to use for a particular problem.

Some considerations for choosing the ZOPT optimizer follow:

- compatibility with PROC MODEL estimation results, because the ZOPT system is also used in PROC MODEL
- faster solutions for smaller problems and for problems that are subject to neither extreme nonlinearities nor ill-conditioning

Some considerations for choosing the ORMP optimizer follow:

- more robust convergence properties for larger problems
- faster and more reliable convergence when there are many constraints on the parameters
- improved estimates in the presence of multiple local minima, or when there is insufficient information to choose initial grid point estimates

## Hessian Evaluation (Experimental)

PROC MODEL uses a linear approximation to the Hessian for all estimation methods by default. In most cases, this linear approximation to the Hessian matrix is sufficient to ensure convergence of the optimization. However, for some combinations of models, data, estimation methods, and optimization methods, the estimates that PROC MODEL produces are sensitive to the linear approximation error in the Hessian. To correct for this shortcoming, PROC TMODEL computes an exact analytical representation of the Hessian for many of the estimation methods by default. The exact Hessian that PROC TMODEL uses also improves convergence properties of the ORMP optimizer for some problems. Table 39.2 summarizes estimation methods and exact analytical representations of the Hessian that are available in PROC TMODEL.

**Table 39.2** PROC TMODEL Hessians for Estimation Methods

| Method | Exact Hessian |
|---|---|
| OLS<br>ITOLS | $\mathbf{X}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{I})\mathbf{X}+\frac{1}{2}(\mathbf{Q}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{I})\mathbf{r}+\mathbf{r}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{I})\mathbf{Q})$ |
| SUR<br>ITSUR | $\mathbf{X}'(\mathbf{S}^{-1}\otimes\mathbf{I})\mathbf{X}+\frac{1}{2}(\mathbf{Q}'(\mathbf{S}^{-1}\otimes\mathbf{I})\mathbf{r}+\mathbf{r}'(\mathbf{S}^{-1}\otimes\mathbf{I})\mathbf{Q})$ |
| N2SLS<br>ITN2SLS | $\mathbf{X}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{W})\mathbf{X}+\frac{1}{2}(\mathbf{Q}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{W})\mathbf{r}+\mathbf{r}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{W})\mathbf{Q})$ |
| N3SLS<br>ITN3SLS | $\mathbf{X}'(\mathbf{S}^{-1}\otimes\mathbf{W})\mathbf{X}+\frac{1}{2}(\mathbf{Q}'(\mathbf{S}^{-1}\otimes\mathbf{W})\mathbf{r}+\mathbf{r}'(\mathbf{S}^{-1}\otimes\mathbf{W})\mathbf{Q})$ |

The variables in this table are defined as follows:

- $n$ the number of nonmissing observations
- $g$ the number of equations
- $k$ the number of instrumental variables
- $p$ the number of parameters
- $\mathbf{r}$ the $ng \times 1$ vector of residuals for the $g$ equations stacked together
- $\mathbf{S}$ a $g \times g$ matrix that estimates $\mathbf{\Sigma}$, the covariances of the errors across equations (referred to as the $\mathbf{S}$ matrix)
- $\mathbf{I}$ an $n \times n$ identity matrix
- $\mathbf{X}$ an $ng \times p$ matrix of partial derivatives of the residuals with respect to the parameters
- $\mathbf{Q}$ an $ng \times p \times p$ vector of matrices of second-order partial derivatives of the residuals with respect to the parameters
- $\mathbf{W}$ an $n \times n$ matrix, $\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'$
- $\mathbf{Z}$ an $n \times k$ matrix of instruments

The exact analytic Hessians are not currently available for the FIML or GMM estimation methods or for models with random effects.

## Multithreaded Calculations

PROC TMODEL uses concurrent computation threads to reduce the time it takes to perform estimation and simulation tasks. Because the characteristics of the input data, model program, and task can vary, PROC TMODEL uses different strategies for breaking its calculations into pieces that can be executed concurrently. For example, when you are estimating a simple model by using a data set with many observations, the most efficient multithreading strategy is to partition the observations and to evaluate the objective function concurrently across the partitions. However, for a highly nonlinear estimation problem with many parameters and fewer observations, the best strategy might be to execute the minimization problem concurrently for multiple regions of the parameter space and then choose the region that yields the optimal estimates.

PROC TMODEL automatically determines which threading strategy to use for each estimation or simulation modeling task by default. Alternatively, you can specify the threading strategy manually by specifying priority options in the PERFORMANCE statement. For each modeling task, the threading strategy is determined by the number of threads allocated to each job in the task. A job is an aspect or dimension of the task that can be executed concurrently with another job. The total number of threads that are used to complete each modeling task is determined as follows,

$$n^* = \prod_{i \in J} \mathrm{ceil}(n \frac{p_i}{p_{\mathrm{tot}}}) \qquad \text{for } p_{\mathrm{tot}} = \sum_{k \in J} p_k$$

where $n^*$ is the actual number of threads used, $n$ is the number of threads specified, $p_i = \max(\mathrm{priority}_i, 10^{-8})$, $\mathrm{priority}_i \in [0, 1]$ is the priority specified for the $i$th job, and $J$ is the set of all jobs in the modeling task. The allocation of threads to jobs in modeling tasks creates the same number or more threads than the number specified in the PERFORMANCE statement and CPUCOUNT= system option.

## Multithreading Estimation Calculations

The following jobs in estimation tasks can be executed currently:

- BY-group processing
- grid search for optimal parameters specified using the START= option in the FIT statement, or multistart global optimization specified by the MULTISTART suboption of the OPTIMIZER= option in the FIT statement
- evaluation of the objective function across partitions of the DATA= data set

For model programs that use lagging functions, the observations in the DATA= data set must be processed sequentially to compute the objective function. In these cases, multithreading across partitions of the data is not possible unless you specify a CROSSSECTION statement or a RANDOM statement. The priority of each estimation job can be specified in the PERFORMANCE statement. When no priorities are specified, PROC TMODEL assigns jobs priorities based on the number of BY groups in the input data set, the number of observations in the input data set, the presence of a START= or MULTISTART option in the FIT statement, and the lag length of the model program.

## Multithreading Simulation Calculations

The following jobs in simulation tasks can be executed concurrently:

- BY-group processing
- processing repetitions in Monte Carlo simulations that are specified using the RANDOM= option in the SOLVE statement

The priority of each simulation job can be specified in the PERFORMANCE statement. When no priorities are specified, PROC TMODEL assigns jobs priorities based on the number of BY groups in the input data set, the presence of a RANDOM= option, and the lag length of the model program.

# Examples: TMODEL Procedure

## Example 39.1: Thread Allocation Using the Performance Statement

This example illustrates how you can use the PERFORMANCE statement to improve the performance of parameter estimation for a data set that contains multiple BY groups. For clarity, a small data set and a simple model are used. In practice, the benefits of configuring the thread allocation strategy are realized only for larger data sets and more computationally demanding models.

In the following PROC TMODEL step, a linear model is estimated for 20 BY groups, each of which contains 1,001 observations:

```
data d;
   call streaminit(1);
   do iby = 1 to 2;
      do jby = 1 to 10;
         do x = -500 to 500;
            y = 2*x + 1 + rand('normal');
            output;
         end;
      end;
   end;
run;

proc tmodel data=d;
   performance nthreads=4 bypriority=1 partpriority=1 / threadconfig timings;
   y = a*x + b;
   by iby jby;
   fit y;
quit;
```

In this example, PROC TMODEL performs the estimations in two concurrent threads, where each thread performs the estimation for 10 BY groups. Also, within the estimation of each BY group, the 1,001 observations are processed concurrently in two partitions, one with 501 observations and the other with 500 observations. Figure 39.1.1 shows how PROC TMODEL divides this estimation problem into threads and the time it takes to complete all 20 estimations.

**Output 39.1.1** Multithreading Performance for Two BY-Group Threads

**The TMODEL Procedure**

| Multithreading Configuration | | |
|---|---|---|
| Calculation Type | Concurrency Factor | Priority |
| BY Group Processing | 2 | 1.00 |
| Nonlinear Optimization | 1 | 0 |
| Data Partitioning | 2 | 1.00 |

**Output 39.1.1** *continued*

| Performance Summary | |
|---|---|
| Threads Used | 4 |
| Task Time (sec) | 4.41974 |
| Program Run Time (sec) | 0.233087 |
| Data Time (sec) | 0.223164 |

Another way to perform these estimation tasks is to divide the 20 BY groups into four concurrent threads, where the estimation of each BY group is performed on a single partition that contains all 1,001 observations. You can do this by setting the PARTPRIORITY= option to 0 in the following PERFORMANCE statement:

```
proc tmodel data=d;
   performance nthreads=4 bypriority=1 partpriority=0 / threadconfig timings;
   y = a*x + b;
   by iby jby;
   fit y;
quit;
```

Figure 39.1.2 shows that maximizing the number of BY-group threads improves the performance of this model and data set.

**Output 39.1.2** Multithreading Performance for Four BY-Group Threads

**The TMODEL Procedure**

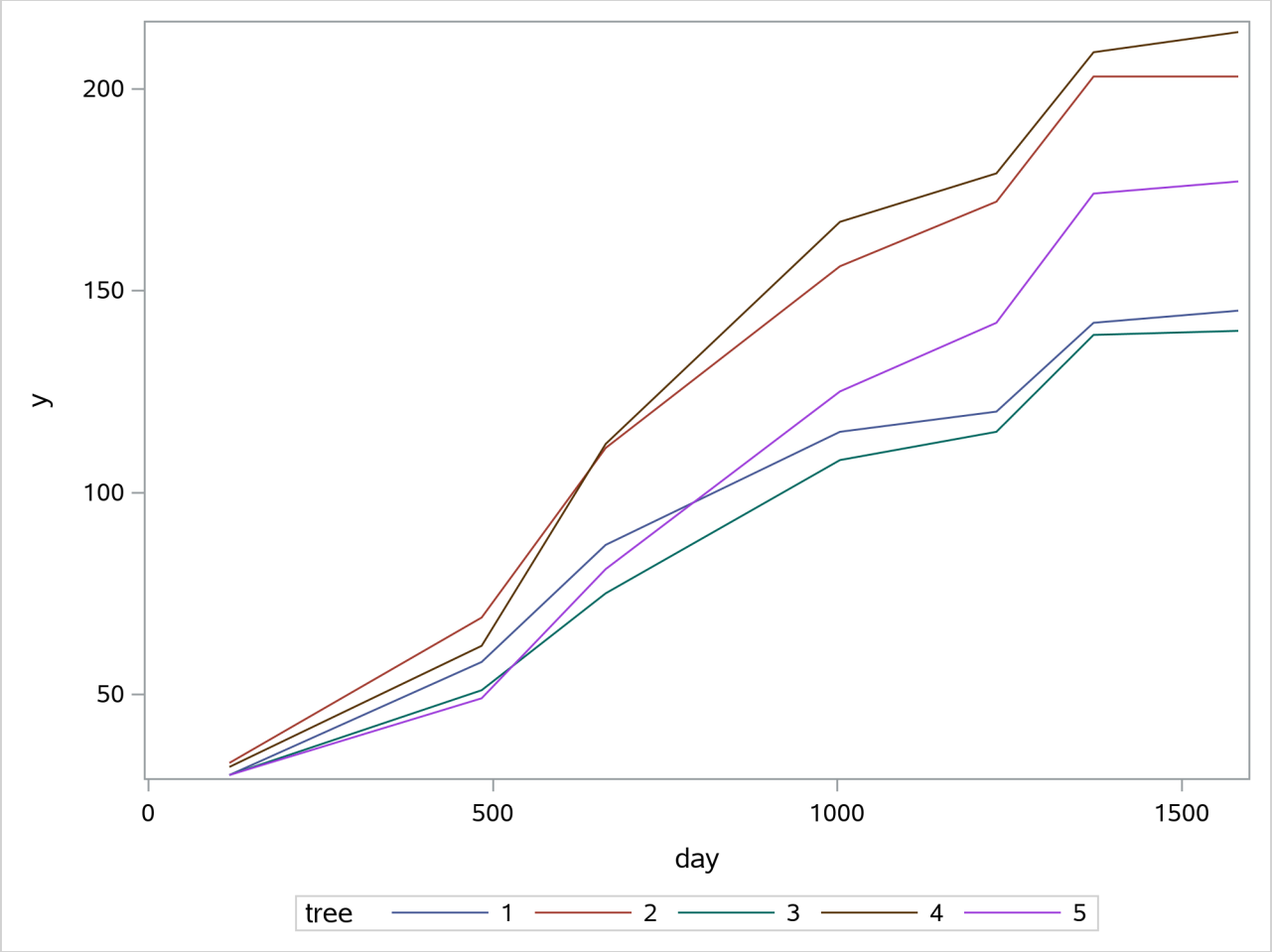
| Multithreading Configuration | | |
|---|---|---|
| Calculation Type | Concurrency Factor | Priority |
| BY Group Processing | 4 | 1.00 |
| Nonlinear Optimization | 1 | 0 |
| Data Partitioning | 1 | 0 |

| Performance Summary | |
|---|---|
| Threads Used | 4 |
| Task Time (sec) | 4.874322 |
| Program Run Time (sec) | 0.248641 |
| Data Time (sec) | 0.224331 |

## Example 39.2: Random-Effects Parameter Estimation (Experimental)

This example models the circumference of five orange trees over 1,600 days by using a logistic curve to represent the growth of each tree over time in conjunction with a random-effects parameter that accounts for variation among the trees. The tree data for this example come from Draper and Smith (1981) and are displayed in Output 39.2.1.

**Output 39.2.1** Orange Tree Growth Curves




Lindstrom and Bates (1990) and Pinheiro and Bates (1995) propose the following logistic nonlinear mixed model for these data:

$$y_{ij} = \frac{b_1 + u_{i1}}{1 + \exp[-(d_{ij} - b_2)/b_3]} + e_{ij}$$

Here, $y_{ij}$ represents the $j$th circumference measurement on the $i$th tree ($i = 1, \ldots, 5$; $j = 1, \ldots, 7$); $d_{ij}$ is the corresponding day; $b_1, b_2$, and $b_3$ are the fixed-effects parameters; $u_{i1}$ are the random-effect parameters assumed to be iid $N(0, \sigma_u^2)$; and $e_{ij}$ are the residual errors assumed to be iid $N(0, \sigma_e^2)$ and independent of the $u_{i1}$. The random-effect parameters $u_{i1}$ enter the model linearly.

The following PROC TMODEL step estimates the fixed-effects and random-effect parameters in this model:

```
proc tmodel data=tree;
   parms    b1  200
            b2  800
            b3  800
            s2u;
   num = b1 + u1;
   ex  = exp(-(day-b2)/b3);
```

```
    den = 1 + ex;
    y = num/den;
    crosssection tree;
    random u1 ~ normal(0,s2u);
    fit y;
quit;
```

The four SAS programming statements in this PROC TMODEL step specify the logistic model for tree growth. The variable u1 identifies the random effect in this model.

The CROSSSECTION statement defines a variable that indicates when new realizations of the random effect are encountered in the DATA= data set; in this case the variable tree is used.

The RANDOM statement defines the single random effect to be u1 and specifies that it follow a normal distribution with mean 0 and variance s2u. For models that have a nonlinear dependence on the random-effects variables, you can use the NUMQUADPTS= option to specify the number of Gaussian quadrature points to use in the approximation of the likelihood function. In this example, the model has a linear dependence on u1, so the default (NUMQUADPTS=1) is used.

For this example, as with many nonlinear random-effects models, the parameter optimization is sensitive to the selection of initial estimates. Therefore, in the PARMS statement, the values for the fixed-effects parameters are initialized with values based on a cursory examination of Output 39.2.1 to assure convergence. The parameter b1 represents an asymptotic limit for the circumference of an average tree. The parameters b2 and b3 are characteristic of the growth period for the orange trees.

Figure 39.2.2 shows the estimates for the fixed-effects and random-effect parameters together with their standard errors and approximate *t*-values.

**Output 39.2.2** Fixed-Effects and Random-Effect Parameters for the Orange Tree Model

**The TMODEL Procedure**

**Nonlinear Random Effects Estimates**

| Parameter | Estimate | Approx Std Err | t Value | Approx Pr > \|t\| |
|---|---|---|---|---|
| **b1** | 192.0409 | 13.6457 | 14.07 | <.0001 |
| **b2** | 727.8899 | 4.4899 | 162.12 | <.0001 |
| **b3** | 347.9685 | 3.4501 | 100.86 | <.0001 |
| **s2u** | 1016.531 | 535.2 | 1.90 | 0.0669 |

# References


Davidian, M., and Giltinan, D. M. (1995). *Nonlinear Models for Repeated Measurement Data*. New York: Chapman & Hall.

Draper, N. R., and Smith, H. (1981). *Applied Regression Analysis*. 2nd ed. New York: John Wiley & Sons.

Lindstrom, M. J., and Bates, D. M. (1990). "Nonlinear Mixed Effects Models for Repeated Measures Data." *Biometrics* 46:673–687.

Pinheiro, J. C., and Bates, D. M. (1995). "Approximations to the Log-Likelihood Function in the Nonlinear Mixed-Effects Model." *Journal of Computational and Graphical Statistics* 4:12–35.

# Index