THE POWER TO KNOW.

SAS/STAT® 9.22 User's Guide

The NLIN Procedure

(Book Excerpt)

SAS® Documentation

Chapter 60
The NLIN Procedure

Contents

Overview: NLIN Procedure **4892**

Getting Started: NLIN Procedure **4894**

Nonlinear or Linear Model 4894

Notation for Nonlinear Regression Models 4895

Estimating the Parameters in the Nonlinear Model 4896

Syntax: NLIN Procedure **4901**

PROC NLIN Statement 4902

BOUNDS Statement 4907

BY Statement 4908

CONTROL Statement 4909

DER Statements 4909

ID Statement 4909

MODEL Statement 4910

OUTPUT Statement 4910

PARAMETERS Statement 4913

RETAIN Statement 4916

Other Programming Statements 4916

Details: NLIN Procedure **4917**

Automatic Derivatives 4917

Hougaard's Measure of Skewness 4919

Missing Values 4921

Special Variables 4921

Troubleshooting 4923

Computational Methods 4925

Output Data Sets 4930

Confidence Intervals 4930

Covariance Matrix of Parameter Estimates 4931

Convergence Measures 4932

Displayed Output 4933

Incompatibilities with SAS 6.11 and Earlier Versions of PROC NLIN 4934

ODS Table Names 4935

Examples: NLIN Procedure **4937**

Example 60.1: Segmented Model 4937

Example 60.2: Iteratively Reweighted Least Squares 4942

Example 60.3: Probit Model with Likelihood Function 4945
Example 60.4: Affecting Curvature through Parameterization 4948
Example 60.5: Comparing Nonlinear Trends among Groups 4955
References **4965**

Overview: NLIN Procedure

The NLIN procedure fits nonlinear regression models and estimates the parameters by nonlinear least squares or weighted nonlinear least squares. You specify the model with programming statements. This gives you great flexibility in modeling the relationship between the response variable and independent (regressor) variables. It does, however, require additional coding compared to model specifications in linear modeling procedures such as the REG, GLM, and MIXED procedures.

Estimating parameters in a nonlinear model is an iterative process that commences from starting values. You need to declare the parameters in your model and supply their initial values for the NLIN procedure. You do not need to specify derivatives of the model equation with respect to the parameters. Although facilities for specifying first and second derivatives exist in the NLIN procedure, it is not recommended that you specify derivatives this way. Obtaining derivatives from user-specified expressions predates the high-quality automatic differentiator that is now used by the NLIN procedure.

Nonlinear least-squares estimation involves finding those values in the parameter space that minimize the (weighted) residual sum of squares. In a sense, this is a "distribution-free" estimation criterion since the distribution of the data does not need to be fully specified. Instead, the assumption of homoscedastic and uncorrelated model errors with zero mean is sufficient. You can relax the homoscedasticity assumption by using a weighted residual sum of squares criterion. The assumption of uncorrelated errors (independent observations) cannot be relaxed in the NLIN procedure. In summary, the primary assumptions for analyses with the NLIN procedure are as follows:

- The structure in the response variable can be decomposed additively into a mean function and an error component.
- The model errors are uncorrelated and have zero mean. Unless a weighted analysis is performed, the errors are also assumed to be homoscedastic (have equal variance).
- The mean function consists of known regressor (independent) variables and unknown constants (the parameters).

Fitting nonlinear models can be a difficult undertaking. There is no closed-form solution for the parameter estimates, and the process is iterative. There can be one or more local minima in the residual sum of squares surface, and the process depends on the starting values supplied by the user. You can reduce the dependence on the starting values and reduce the chance to arrive at a local minimum by specifying a grid of starting values. The NLIN procedure then computes the residual sum of squares at each point on the grid and starts the iterative process from the point that yields the lowest sum of squares. Even in this case, however, convergence does not guarantee that a global minimum has been found.

The numerical behavior of a model and a model–data combination can depend on the way in which you parameterize the model—for example, whether parameters are expressed on the logarithmic scale or not. Parameterization also has bearing on the interpretation of the estimated quantities and the statistical properties of the parameter estimators. Inferential procedures in nonlinear regression models are typically approximate in that they rely on the asymptotic properties of the parameter estimators that are obtained as the sample size grows without bound. Such asymptotic inference can be questionable in small samples, especially if the behavior of the parameter estimators is "far-from-linear." Reparameterization of the model can yield parameters whose behavior is akin to that of estimators in linear models. These parameters exhibit close-to-linear behavior.

The NLIN procedure solves the nonlinear least squares problem by one of the following four algorithms (methods):

- steepest-descent or gradient method
- Newton method
- modified Gauss-Newton method
- Marquardt method

These methods use derivatives or approximations to derivatives of the SSE with respect to the parameters to guide the search for the parameters producing the smallest SSE. Derivatives computed automatically by the NLIN procedure are analytic, unless the model contains functions for which an analytic derivative is not available.

Using PROC NLIN, you can also do the following:

- confine the estimation procedure to a certain range of values of the parameters by imposing bounds on the estimates
- produce new SAS data sets containing predicted values, parameter estimates, residuals and other model diagnostics, estimates at each iteration, and so forth.

You can use the NLIN procedure for segmented models (see Example 60.1) or robust regression (see Example 60.2). You can also use it to compute maximum-likelihood estimates for certain models (see Jennrich and Moore 1975; Charnes, Frome, and Yu 1976). For maximum likelihood estimation in a model with a linear predictor and binomial error distribution, see the LOGISTIC, PROBIT, GENMOD, GLIMMIX, and CATMOD procedures. For a linear model with a Poisson, gamma, or inverse gaussian error distribution, see the GENMOD and GLIMMIX procedures. For likelihood estimation in a linear model with a normal error distribution, see the MIXED, GENMOD, and GLIMMIX procedures. The PHREG and LIFEREG procedures fit survival models by maximum likelihood. For general maximum likelihood estimation, see the NLP procedure in the *SAS/OR User's Guide* and the NLMIXED procedure in the *SAS/STAT User's Guide*. These procedures are recommended over the NLIN procedure for solving maximum likelihood problems.

PROC NLIN uses the Output Delivery System (ODS), a SAS subsystem that provides capabilities for displaying and controlling the output from SAS procedures. ODS enables you to convert any of the output from PROC NLIN into a SAS data set. See the section "ODS Table Names" on page 4935 for a listing of the ODS tables that are produced by the NLIN procedure.

Getting Started: NLIN Procedure

Nonlinear or Linear Model

The NLIN procedure performs univariate nonlinear regression by using the least squares method. Nonlinear regression analysis is indicated when the functional relationship between the response variable and the predictor variables is nonlinear. Nonlinearity in this context refers to a nonlinear relationship in the *parameters*. Many linear regression models exhibit a relationship in the regressor (predictor) variables that is not simply a straight line. This does not make the models nonlinear. A model is nonlinear in the parameters if the derivative of the model with respect to a parameter depends on this or other parameters.

Consider, for example the models

$$\begin{aligned}
\mathrm{E}[Y|x] &= \beta_0 + \beta_1 x \\
\mathrm{E}[Y|x] &= \beta_0 + \beta_1 x + \beta_2 x^2 \\
\mathrm{E}[Y|x] &= \beta + x/\alpha
\end{aligned}$$

In these expressions, $\mathrm{E}[Y|x]$ denotes the expected value of the response variable Y at the fixed value of x. (The conditioning on x simply indicates that the predictor variables are assumed to be non-random in models fit by the NLIN procedure. Conditioning is often omitted for brevity in this chapter.)

Only the third model is a nonlinear model. The first model is a simple linear regression. It is linear in the parameters β_0 and β_1 since the model derivatives do not depend on unknowns:

$$\begin{aligned}
\frac{\partial}{\beta_0}(\beta_0 + \beta_1 x) &= 1 \\
\frac{\partial}{\beta_1}(\beta_0 + \beta_1 x) &= x
\end{aligned}$$

The model is also linear in its relationship with x (a straight line). The second model is also linear in the parameters, since

$$\frac{\partial}{\beta_0}\left(\beta_0 + \beta_1 x + \beta_2 x^2\right) = 1$$
$$\frac{\partial}{\beta_1}\left(\beta_0 + \beta_1 x + \beta_2 x^2\right) = x$$
$$\frac{\partial}{\beta_2}\left(\beta_0 + \beta_1 x + \beta_2 x^2\right) = x^2$$

It is a *curvilinear* model since it exhibits a curved relationship when plotted against x. The third model, finally, is a nonlinear model since

$$\frac{\partial}{\beta}(\beta + x/\alpha) = 1$$
$$\frac{\partial}{\alpha}(\beta + x/\alpha) = -\frac{x}{\alpha^2}$$

The second of these derivatives depends on a parameter α. A model is nonlinear if it is not linear in at least one parameter.

Notation for Nonlinear Regression Models

This section briefly introduces the basic notation for nonlinear regression models that applies in this chapter. Additional notation is introduced throughout as needed.

The $(n \times 1)$ vector of observed responses is denoted as $\mathbf{y}$. This vector is the realization of an $(n \times 1)$ random vector $\mathbf{Y}$. The NLIN procedure assumes that the variance matrix of this random vector is $\sigma^2\mathbf{I}$. In other words, the observations have equal variance (are homoscedastic) and are uncorrelated. By defining the special variable _WEIGHT_ in your NLIN programming statements, you can introduce heterogeneous variances. If a _WEIGHT_ variable is present, then $\mathrm{Var}[\mathbf{Y}] = \sigma^2\mathbf{W}^{-1}$, where $\mathbf{W}$ is a diagonal matrix containing the values of the _WEIGHT_ variable.

The mean of the random vector is represented by a nonlinear model that depends on parameters $\beta_1, \cdots, \beta_p$ and regressor (independent) variables $z_1, \cdots, z_k$:

$$\mathrm{E}[Y_i] = f\left(\beta_1, \beta_2, \cdots, \beta_p; z_{i1}, \cdots, z_{ik}\right)$$

In contrast to linear models, the number of regressor variables (k) does not necessarily equal the number of parameters (p) in the mean function $f()$. For example, the model fitted in the next subsection contains a single regressor and two parameters.

To represent the mean of the vector of observations, boldface notation is used in an obvious extension of the previous equation:

$$\mathrm{E}[\mathbf{Y}] = \mathbf{f}(\boldsymbol{\beta}; \mathbf{z}_1, \cdots, \mathbf{z}_k)$$

The vector $\mathbf{z}_1$, for example, is an $(n \times 1)$ vector of the values for the first regressor variables. The explicit dependence of the mean function on $\boldsymbol{\beta}$ and/or the $\mathbf{z}$ vectors is often omitted for brevity.

In summary, the stochastic structure of models fit with the NLIN procedure is mathematically captured by

$$\begin{aligned} \mathbf{Y} &= \mathbf{f}(\boldsymbol{\beta}; \mathbf{z}_1, \cdots, \mathbf{z}_k) + \boldsymbol{\epsilon} \\ \mathrm{E}[\boldsymbol{\epsilon}] &= \mathbf{0} \\ \mathrm{Var}[\boldsymbol{\epsilon}] &= \sigma^2 \mathbf{I} \end{aligned}$$

Note that the residual variance σ^2 is typically also unknown. Since it is not estimated in the same fashion as the other p parameters, it is often not counted in the number of parameters of the nonlinear regression. An estimate of σ^2 is obtained after the model fit by the method of moments based on the residual sum of squares.

A matrix that plays an important role in fitting nonlinear regression models is the $(n \times p)$ matrix of the first partial derivatives of the mean function $\mathbf{f}$ with respect to the p model parameters. It is frequently denoted as

$$\mathbf{X} = \frac{\partial \mathbf{f}\,(\boldsymbol{\beta}; \mathbf{z}_1, \cdots, \mathbf{z}_k)}{\partial \boldsymbol{\beta}}$$

The use of the symbol $\mathbf{X}$—common in linear statistical modeling—is no accident here. The first derivative matrix plays a similar role in nonlinear regression to that of the $\mathbf{X}$ matrix in a linear model. For example, the asymptotic variance of the nonlinear least-squares estimators is proportional to $(\mathbf{X}'\mathbf{X})^{-1}$, and projection-type matrices in nonlinear regressions are based on $\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$. Also, fitting a nonlinear regression model can be cast as an iterative process where a nonlinear model is approximated by a series of linear models in which the derivative matrix is the regressor matrix. An important difference between linear and nonlinear models is that the derivatives in a linear model do not depend on any parameters (see previous subsection). In contrast, the derivative matrix $\partial \mathbf{f}(\boldsymbol{\beta})/\partial \boldsymbol{\beta}$ is a function of at least one element of $\boldsymbol{\beta}$. It is this dependence that lies at the core of the fact that estimating the parameters in a nonlinear model cannot be accomplished in closed form, but it is an iterative process that commences with user-supplied starting values and attempts to continually improve on the parameter estimates.

Estimating the Parameters in the Nonlinear Model

As an example of a nonlinear regression analysis, consider the following theoretical model of enzyme kinetics. The model relates the initial velocity of an enzymatic reaction to the substrate concentration.

$$f(\mathbf{x}, \boldsymbol{\theta}) = \frac{\theta_1 x_i}{\theta_2 + x_i}, \quad \text{for } i = 1, 2, \ldots, n$$

where x_i represents the amount of substrate for n trials and $f(\mathbf{x}, \boldsymbol{\theta})$ is the velocity of the reaction. The vector $\boldsymbol{\theta}$ contains the rate parameters. This model is known as the Michaelis-Menten model in biochemistry (Ratkowsky, 1990, p. 59). The model exists in many parameterizations. In the form shown here, θ_1 is the maximum velocity of the reaction that is theoretically attainable. The parameter θ_2 is the substrate concentration at which the velocity is 50% of the maximum.

Suppose that you want to study the relationship between concentration and velocity for a particular enzyme/substrate pair. You record the reaction rate (velocity) observed at different substrate concentrations. A SAS data set is created for this experiment in the following DATA step:

```
data Enzyme;
   input Concentration Velocity @@;
   datalines;
0.26 124.7   0.30 126.9
0.48 135.9   0.50 137.6
0.54 139.6   0.68 141.1
0.82 142.8   1.14 147.6
1.28 149.8   1.38 149.4
1.80 153.9   2.30 152.5
2.44 154.5   2.48 154.7
;
```

The SAS data set Enzyme contains the two variables Concentration (substrate concentration) and Velocity (reaction rate). The following statements fit the Michaelis-Menten model by nonlinear least squares:

```
proc nlin data=Enzyme method=marquardt hougaard;
   parms theta1=155
         theta2=0 to 0.07 by 0.01;
   model Velocity = theta1*Concentration / (theta2 + Concentration);
run;
```

The DATA= option specifies that the SAS data set Enzyme be used in the analysis. The METHOD= option directs PROC NLIN to use the MARQUARDT iterative method. The HOUGAARD option requests that a skewness measure be calculated for the parameters.

The PARMS statement declares the parameters and specifies their initial values. Suppose that V represents the velocity and C represents the substrate concentration. In this example, the initial estimates listed in the PARMS statement for θ_1 and θ_2 are obtained as follows:

θ_1: Because the model is a monotonic increasing function in C, and because

$$\lim_{C\to\infty}\left(\frac{\theta_1 C}{\theta_2 + C}\right) = \theta_1$$

you can take the largest observed value of the variable Velocity (154.7) as the initial value for the parameter Theta1. Thus, the PARMS statement specifies 155 as the initial value for Theta1, which is approximately equal to the maximum observed velocity.

θ_2: To obtain an initial value for the parameter θ_2, first rearrange the model equation to solve for θ_2:

$$\theta_2 = \frac{\theta_1 C}{V} - C$$

By substituting the initial value of Theta1 for θ_1 and taking each pair of observed values of Concentration and Velocity for C and V, respectively, you obtain a set of possible starting values for Theta2 ranging from about 0.01 to 0.07.

You can choose any value within this range as a starting value for Theta2, or you can direct PROC NLIN to perform a preliminary search for the best initial Theta2 value within that range of values. The PARMS statement specifies a range of values for Theta2, resulting in a search over the grid points from 0 to 0.07 in increments of 0.01.

The MODEL statement specifies the enzymatic reaction model

$$V = \frac{\theta_1 C}{\theta_2 + C}$$

in terms of the data set variables Velocity and Concentration and in terms of the parameters in the PARMS statement.

The results from this PROC NLIN invocation are displayed in the following figures.

PROC NLIN evaluates the model at each point on the specified grid for the Theta2 parameter. Figure 60.1 displays the calculations resulting from the grid search.

Figure 60.1 Nonlinear Least-Squares Grid Search

```
                    The NLIN Procedure
                Dependent Variable Velocity

                        Grid Search
                                          Sum of
                theta1       theta2      Squares

                 155.0            0       3075.4
                 155.0       0.0100       2074.1
                 155.0       0.0200       1310.3
                 155.0       0.0300        752.0
                 155.0       0.0400        371.9
                 155.0       0.0500        147.2
                 155.0       0.0600      58.1130
                 155.0       0.0700      87.9662
```

The parameter Theta1 is held constant at its specified initial value of 155, the grid is traversed, and the residual sum of squares is computed at each point. The "best" starting value is the point that corresponds to the smallest value of the residual sum of squares. The best set of starting values is obtained for $\theta_1 = 155, \theta_2 = 0.06$ (Figure 60.1). PROC NLIN uses this point from which to start the following, iterative phase of nonlinear least-squares estimation.

Figure 60.2 displays the iteration history. Note that the first entry in the "Iterative Phase" table echoes the starting values and the residual sum of squares for the best value combination in Figure 60.1. The subsequent rows of the table show the updates of the parameter estimates and the improvement (decrease) in the residual sum of squares. For this data-and-model combination, the first iteration yielded a large improvement in the sum of squares (from 58.113 to 19.7017). Further steps were necessary to improve the estimates in order to achieve the convergence criterion. The NLIN procedure by default determines convergence by using R, the relative offset measure of Bates and Watts (1981). Convergence is declared when this measure is less than 10^{-5}—in this example, after three iterations.

Figure 60.2 Iteration History and Convergence Status

```
                    The NLIN Procedure
               Dependent Variable Velocity
                    Method: Marquardt

                     Iterative Phase
                                            Sum of
          Iter       theta1      theta2     Squares

             0        155.0      0.0600     58.1130
             1        158.0      0.0736     19.7017
             2        158.1      0.0741     19.6606
             3        158.1      0.0741     19.6606

  NOTE: Convergence criterion met.
```

Figure 60.3 Estimation Summary

```
                  Estimation Summary

        Method                        Marquardt
        Iterations                            3
        R                              5.861E-6
        PPC(theta2)                    8.569E-7
        RPC(theta2)                    0.000078
        Object                         2.902E-7
        Objective                      19.66059
        Observations Read                    14
        Observations Used                    14
        Observations Missing                  0
```

A summary of the estimation including several convergence measures (R, PPC, RPC, and Object) is displayed in Figure 60.3.

The “R” measure in Figure 60.3 is the relative offset convergence measure of Bates and Watts. A “PPC” value of $8.569E-7$ indicates that the parameter Theta2 (which has the largest PPC value of the parameters) would change by that relative amount, if PROC NLIN were to take an additional iteration step. The “RPC” value indicates that Theta2 changed by 0.000078, relative to its value in the last iteration. These changes are measured before step length adjustments are made. The “Object” measure indicates that the objective function changed by $2.902E-7$ in relative value from the last iteration.

Figure 60.4 displays the analysis of variance table for the model. The table displays the degrees of freedom, sums of squares, and mean squares along with the model *F* test.

Figure 60.4 Nonlinear Least-Squares Analysis of Variance

```
               NOTE: An intercept was not specified for this model.

                                       Sum of        Mean                 Approx
    Source                     DF     Squares      Square    F Value    Pr > F

    Model                       2      290116      145058    88537.2    <.0001
    Error                      12     19.6606      1.6384
    Uncorrected Total          14      290135
```

Figure 60.5 Parameter Estimates and Approximate 95% Confidence Intervals

```
                                    Approx       Approximate 95%
     Parameter       Estimate     Std Error     Confidence Limits      Skewness

     theta1             158.1        0.6737      156.6        159.6      0.0152
     theta2            0.0741       0.00313     0.0673       0.0809      0.0362
```

Figure 60.5 displays the estimates for each parameter, the associated asymptotic standard error, and the upper and lower values for the asymptotic 95% confidence interval. PROC NLIN also displays the asymptotic correlations between the estimated parameters (not shown).

The skewness measures of 0.0152 and 0.0362 indicate that the parameter estimators exhibit close-to-linear behavior and that their standard errors and confidence intervals can be safely used for inferences.

Thus, the estimated nonlinear model relating reaction velocity and substrate concentration can be written as

$$\widehat{V} = \frac{158.1C}{0.0741 + C}$$

where $\widehat{V}$ represents the predicted velocity or rate of the reaction, and C represents the substrate concentration.

Syntax: NLIN Procedure

> **PROC NLIN** < *options* > **;**
> **BOUNDS** *inequality* < *, . . . , inequality* > **;**
> **BY** *variables* **;**
> **CONTROL** *variable* < *=values* > < *. . . variable* < *=values* > > **;**
> **DER.** *parameter=expression* **;**
> **DER.** *parameter.parameter=expression* **;**
> **ID** *variables* **;**
> **MODEL** *dependent=expression* **;**
> **OUTPUT OUT=***SAS-data-set keyword=names* < *. . . keyword=names* > **;**
> **PARAMETERS** < *parameter-specification* > < *,. . . , parameter-specification* >
> < / **PDATA=***SAS-data-set* > **;**
> **RETAIN** *variable* < *=values* > < *. . . variable* < *=values* > > **;**
> **Programming Statements ;**

The statements in the NLIN procedure, in addition to the PROC NLIN statement, are as follows:

BOUNDS	constrains the parameter estimates within specified bounds
BY	specifies variables to define subgroups for the analysis
DER	specifies the first or second partial derivatives
ID	specifies additional variables to add to the output data set
MODEL	defines the relationship between the dependent and independent variables (the mean function)
OUTPUT	creates an output data set containing observation-wise statistics
PARAMETERS	identifies parameters to be estimated and their starting values
Programming Statements	includes, for example, assignment statements, ARRAY statements, DO loops, and other program control statements. These are valid SAS expressions that can appear in the DATA step. PROC NLIN enables you to create new variables within the procedure and use them in the nonlinear analysis. These programming statements can appear anywhere in the PROC NLIN code, but new variables must be created before they appear in other statements. The NLIN procedure automatically creates several variables that are also available for use in the analysis. See the section "Special Variables" on page 4921 for more information.

The PROC NLIN, PARAMETERS, and MODEL statements are required.

PROC NLIN Statement

PROC NLIN < *options* > **;**

The PROC NLIN statement invokes the procedure.

The following table lists important options available in the PROC NLIN statement. All options are subsequently discussed in alphabetical order.

Table 60.1 Summary of Important Options in PROC NLIN Statement

Option	**Description**
Options Related to Data Sets	
DATA=	specifies input data set
OUTEST=	specifies output data set for parameter estimates, covariance matrix, etc.
SAVE	requests that final estimates be added to the OUTEST= data set
Optimization Options	
BEST=	limits display of grid search
METHOD=	chooses the optimization method
MAXITER=	specifies maximum number of iterations
MAXSUBIT=	specifies maximum number of step halvings
NOHALVE	allows objective function to increase between iterations
RHO=	controls the step-size search
SMETHOD=	specifies the step-size search method
TAU=	controls the step-size search
G4	uses Moore-Penrose inverse
UNCORRECTEDDF	does not expense degrees of freedom when bounds are active
SIGSQ=	specifies fixed value for residual variance
Singularity and Convergence Criteria	
CONVERGE=	tunes the convergence criterion
CONVERGEOBJ=	uses the change in loss function as the convergence criterion and tunes its value
CONVERGEPARM=	uses the maximum change in parameter estimates as the convergence criterion and tunes its value
SINGULAR=	tunes the singularity criterion used in matrix inversions
Displayed Output	
HOUGAARD	adds Hougaard’s skewness measure to the “Parameter Estimates” table
NOITPRINT	suppresses the “Iteration History” table
NOPRINT	suppresses displayed output
LIST	displays model program and variable list
LISTALL	selects the LIST, LISTDEP, LISTDER, and LISTCODE options
LISTCODE	displays the model program code

Table 60.1 *continued*

Option	Description
LISTDEP	displays dependencies of model variables
LISTDER	displays the derivative table
TOTALSS	adds uncorrected or corrected total sum of squares to the analysis of variance table
XREF	displays cross-reference of variables
Trace Model Execution	
FLOW	displays execution messages for program statements
PRINT	displays results of statements in model program
TRACE	displays results of operations in model program

ALPHA=α

specifies the level of significance α used in the construction of $100(1-\alpha)\%$ confidence intervals. The value must be strictly between 0 and 1; the default value of $\alpha = 0.05$ results in 95% intervals. This value is used as the default confidence level for limits computed in the “Parameter Estimates” table and with the LCLM, LCL, UCLM, and UCL options in the OUTPUT statement.

BEST=*n*

requests that PROC NLIN display the residual sums of squares only for the best n combinations of possible starting values from the grid. If you do not specify the BEST= option, PROC NLIN displays the residual sum of squares for every combination of possible parameter starting values.

CONVERGE=*c*

specifies the convergence criterion for PROC NLIN. For all iterative methods the relative offset convergence measure of Bates and Watts is used by default to determine convergence. This measure is labeled “R” in the “Estimation Summary” table. The iterations are said to have converged for CONVERGE=c if

$$\sqrt{\frac{\mathbf{r}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{r}}{\mathrm{LOSS}^{(i)}}} < c$$

where $\mathbf{r}$ is the residual vector and $\mathbf{X}$ is the $(n \times p)$ matrix of first derivatives with respect to the parameters. The default LOSS function is the sum of squared errors (SSE), and $\mathrm{LOSS}^{(i)}$ denotes the value of the loss function at the ith iteration. By default, CONVERGE=10^{-5}. The R convergence measure cannot be computed accurately in the special case of a perfect fit (residuals close to zero). When the SSE is less than the value of the SINGULAR= criterion, convergence is assumed.

CONVERGEOBJ=*c*

uses the change in the LOSS function as the convergence criterion and tunes the criterion. The iterations are said to have converged for CONVERGEOBJ=c if

$$\frac{|\mathrm{LOSS}^{(i-1)} - \mathrm{LOSS}^{(i)}|}{|\mathrm{LOSS}^{(i-1)} + 10^{-6}|} < c$$

where LOSS$^{(i)}$ is the LOSS for the ith iteration. The default LOSS function is the sum of squared errors (SSE), the residual sum of squares. The constant c should be a small positive number. For more details about the LOSS function, see the section "Special Variable Used to Determine Convergence Criteria" on page 4922. For more details about the computational methods in the NIN procedure, see the section "Computational Methods" on page 4925.

Note that in SAS 6 the CONVERGE= and CONVERGEOBJ= options both requested that convergence be tracked by the relative change in the loss function. If you specify the CONVERGEOBJ= option in newer releases, the CONVERGE= option is disabled. This enables you to track convergence as in SAS 6.

CONVERGEPARM=*c*

uses the maximum change among parameter estimates as the convergence criterion and tunes the criterion. The iterations are said to have converged for CONVERGEPARM=c if

$$\max_j \left(\frac{|\beta_j^{(i-1)} - \beta_j^{(i)}|}{|\beta_j^{(i-1)}|} \right) < c$$

where $\beta_j^{(i)}$ is the value of the jth parameter at the ith iteration.

The default convergence criterion is CONVERGE. If you specify CONVERGEPARM=c, the maximum change in parameters is used as the convergence criterion. If you specify both the CONVERGEOBJ= and CONVERGEPARM= options, PROC NLIN continues to iterate until the decrease in LOSS is sufficiently small (as determined by the CONVERGEOBJ= option) and the maximum change among the parameters is sufficiently small (as determined by the CONVERGEPARM= option).

DATA=*SAS-data-set*

specifies the input SAS data set to be analyzed by PROC NLIN. If you omit the DATA= option, the most recently created SAS data set is used.

FLOW

displays a message for each statement in the model program as it is executed. This debugging option is rarely needed, and it produces large amounts of output.

G4

uses a Moore-Penrose inverse (g_4-inverse) in parameter estimation. See Kennedy and Gentle (1980) for details.

HOUGAARD

adds Hougaard's measure of skewness to the "Parameter Estimates" table (Hougaard 1982, 1985). The skewness measure is one method of assessing a parameter estimator's close-to-linear behavior in the sense of Ratkowsky (1983, 1990). The behavior of estimators that are close to linear approaches that of least-squares estimators in linear models, which are unbiased and have minimum variance. When you specify the HOUGAARD option, the standardized skewness measure of Hougaard (1985) is added for each parameter to the "Parameter Estimates" table. Because of the linkage between nonlinear behavior of a parameter estimator in nonlinear regression and the nonnormality of the estimator's sampling distribution, Ratkowsky (1990, p. 28) provides the following rules to interpret the (standardized) Hougaard skewness measure:

- Values less than 0.1 in absolute value indicate very close-to-linear behavior.
- Values between 0.1 and 0.25 in absolute value indicate reasonably close-to-linear behavior.
- The nonlinear behavior is apparent for absolute values above 0.25 and is considerable for absolute values above 1.

See the section "Hougaard's Measure of Skewness" on page 4919 for further details. Example 60.4 shows how to use this measure to evaluate changes in the parameterization of a nonlinear model. Computation of the Hougaard skewness measure requires first and second derivatives. If you do not provide derivatives with the DER statement—and it is recommended that you do not—the analytic derivatives are computed for you.

LIST

displays the model program and variable lists, including the statements added by macros. Note that the expressions displayed by the LIST option do not necessarily represent the way the expression is actually calculated—because intermediate results for common subexpressions can be reused—but are shown in expanded form. To see how the expression is actually evaluated, use the LISTCODE option.

LISTALL

selects the LIST, LISTDEP, LISTDER, and LISTCODE options.

LISTCODE

displays the derivative tables and the compiled model program code. The LISTCODE option is a debugging feature and is not normally needed.

LISTDEP

produces a report that lists, for each variable in the model program, the variables that depend on it and the variables on which it depends.

LISTDER

displays a table of derivatives. The derivatives table lists each nonzero derivative computed for the problem. The derivative listed can be a constant, a variable in the model program, or a special derivative variable created to hold the result of an expression.

MAXITER=*n*

specifies the maximum number n of iterations in the optimization process. The default is $n = 100$.

MAXSUBIT=*n*

places a limit on the number of step halvings. The value of MAXSUBIT must be a positive integer and the default value is $n = 30$.

METHOD=GAUSS

METHOD=MARQUARDT

METHOD=NEWTON

METHOD=GRADIENT

specifies the iterative method employed by the NLIN procedure in solving the nonlinear least-squares problem. The GAUSS, MARQUARDT, and NEWTON methods are more robust than the GRADIENT method. If you omit the METHOD= option, METHOD=GAUSS is used. See the section "Computational Methods" on page 4925 for more information.

NOITPRINT

suppresses the display of the "Iteration History" table.

NOHALVE

removes the restriction that the objective value must decrease at every iteration. Step halving is still used to satisfy BOUNDS and to ensure that the number of observations that can be evaluated does not decrease. The NOHALVE option can be useful in weighted nonlinear least-squares problems where the weights depend on the parameters, such as in iteratively reweighted least-squares (IRLS) fitting. See Example 60.2 for an application of IRLS fitting.

NOPRINT

suppresses the display of the output. Note that this option temporarily disables the Output Delivery System (ODS). For more information, see Chapter 20, "Using the Output Delivery System."

OUTEST=*SAS-data-set*

specifies an output data set that contains the parameter estimates produced at each iteration. See the section "Output Data Sets" for details. If you want to create a permanent SAS data set, you must specify a two-level name. See the chapter "SAS Files" in *SAS Language Reference: Concepts* for more information about permanent SAS data sets.

PRINT

displays the result of each statement in the program as it is executed. This option is a debugging feature that produces large amounts of output and is normally not needed.

RHO=*value*

specifies a value that controls the step-size search. By default RHO=0.1, except when METHOD=MARQUARDT. In that case, RHO=10. See the section "Step-Size Search" on page 4930 for more details.

SAVE

specifies that, when the iteration limit is exceeded, the parameter estimates from the final iteration be output to the OUTEST= data set. These parameter estimates are associated with the observation for which _TYPE_="FINAL". If you omit the SAVE option, the parameter estimates from the final iteration are not output to the data set unless convergence has been attained.

SIGSQ=*value*

specifies a value to use as the estimate of the residual variance in lieu of the estimated mean-squared error. This value is used in computing the standard errors of the estimates. Fixing the value of the residual variance can be useful, for example, in maximum likelihood estimation.

SINGULAR=*s*

specifies the singularity criterion, s, which is the absolute magnitude of the smallest pivot value allowed when inverting the Hessian or the approximation to the Hessian. The default value is 1E4 times the machine epsilon; this product is approximately $1\text{E}-12$ on most computers.

SMETHOD=HALVE

SMETHOD=GOLDEN

SMETHOD=CUBIC

specifies the step-size search method. The default is SMETHOD=HALVE. See the section "Step-Size Search" on page 4930 for details.

TAU=*value*

specifies a value that is used to control the step-size search. The default is TAU=1, except when METHOD=MARQUARDT. In that case the default is TAU=0.01. See the section "Step-Size Search" on page 4930 for details.

TOTALSS

adds to the analysis of variance table the uncorrected total sum of squares in models that have an (implied) intercept, and adds the corrected total sum of squares in models that do not have an (implied) intercept.

TRACE

displays the result of each operation in each statement in the model program as it is executed, in addition to the information displayed by the FLOW and PRINT options. This debugging option is needed very rarely, and it produces even more output than the FLOW and PRINT options.

XREF

displays a cross-reference of the variables in the model program showing where each variable is referenced or given a value. The XREF listing does not include derivative variables.

UNCORRECTEDDF

specifies that no degrees of freedom be lost when a bound is active. When the UNCORRECTEDDF option is not specified, an active bound is treated as if a restriction were applied to the set of parameters, so one parameter degree of freedom is deducted.

BOUNDS Statement

BOUNDS *inequality* < , . . . , *inequality* > **;**

The BOUNDS statement restricts the parameter estimates so that they lie within specified regions. In each BOUNDS statement, you can specify a series of boundary values separated by commas. The series of bounds is applied simultaneously. Each boundary specification consists of a list of parameters, an inequality comparison operator, and a value. In a single-bounded expression, these three elements follow one another in the order described. The following are examples of valid single-bounded expressions:

```
bounds a1-a10 <= 20;
bounds c > 30;
bounds a b c > 0;
```

Multiple-bounded expressions are also permitted. For example:

```
bounds 0 <= B<= 10;
bounds 15 < x1 <= 30;
bounds r <= s <= p < q;
```

If you need to restrict an expression involving several parameters (for example, $\alpha + \beta < 1$), you can reparameterize the model so that the expression becomes a parameter or so that the boundary constraint can be expressed as a simple relationship between two parameters. For example, the boundary constraint $\alpha + \beta < 1$ in the model

```
model y = alpha + beta*x;
```

can be achieved by parameterizing $\theta = 1 - \beta$ as follows:

```
bounds alpha < theta;
model y = alpha + (1-theta)*x;
```

Starting with SAS 7.01, Lagrange multipliers are reported for all bounds that are enforced (active) when the estimation terminates. In the “Parameter Estimates” table, the Lagrange multiplier estimates are identified with names *Bound1*, *Bound2*, and so forth. An active bound is treated as if a restriction were applied to the set of parameters so that one parameter degree of freedom is deducted. You can use the UNCORRECTEDDF option to prevent the loss of degrees of freedom when bounds are active.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC NLIN to obtain separate analyses for observations in groups 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 your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data by using the SORT procedure with a similar BY statement.
- Specify the BY statement option NOTSORTED or DESCENDING in the BY statement for the NLIN 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).

For more information about the BY statement, see *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the *Base SAS Procedures Guide*.

CONTROL Statement

CONTROL *variable* < **=***values* > < . . . *variable* < **=***values* > > **;**

The CONTROL statement declares control variables and specifies their values. A control variable is like a retained variable (see the section "RETAIN Statement" on page 4916) except that it is retained *across* iterations, and the derivative of the model with respect to a control variable is always zero.

DER Statements

DER. *parameter***=***expression* **;**

DER. *parameter.parameter***=***expression* **;**

The DER statement specifies first or second partial derivatives. By default, analytical derivatives are automatically computed. However, you can specify the derivatives yourself by using the DER.parm syntax. Use the first form shown to specify first partial derivatives, and use the second form to specify second partial derivatives. Note that the DER.parm syntax is retained for backward compatibility. The automatic analytical derivatives are, in general, a better choice. For additional information about automatic analytical derivatives, see the section "Automatic Derivatives" on page 4917.

For most of the computational methods, you need only specify the first partial derivative with respect to each parameter to be estimated. For the NEWTON method, specify both the first and the second derivatives. If any needed derivatives are not specified, they are automatically computed.

The expression can be an algebraic representation of the partial derivative of the expression in the MODEL statement with respect to the parameter or parameters that appear on the left side of the DER statement. Numerical derivatives can also be used. The expression in the DER statement must conform to the rules for a valid SAS expression, and it can include any quantities that the MODEL statement expression contains.

ID Statement

ID *variables* **;**

The ID statement specifies additional variables to place in the output data set created by the OUTPUT statement. Any variable on the left side of any assignment statement is eligible. Also, the special variables created by the procedure can be specified. Variables in the input data set do not need to be specified in the ID statement since they are automatically included in the output data set.

MODEL Statement

MODEL *dependent=expression* **;**

The MODEL statement defines the prediction equation by declaring the dependent variable and defining an expression that evaluates predicted values. The expression can be any valid SAS expression that yields a numeric result. The expression can include parameter names, variables in the data set, and variables created by programming statements in the NLIN procedure. Any operators or functions that can be used in a DATA step can also be used in the MODEL statement.

A statement such as

```
model y=expression;
```

is translated into the form

```
model.y=expression;
```

using the compound variable name model.y to hold the predicted value. You can use this assignment directly as an alternative to the MODEL statement. Either a MODEL statement or an assignment to a compound variable such as model.y must appear.

OUTPUT Statement

OUTPUT OUT= *SAS-data-set keyword=names* < . . . *keyword=names* > < / *options* > **;**

The OUTPUT statement specifies an output data set to contain statistics calculated for each observation. For each statistic, specify the keyword, an equal sign, and a variable name for the statistic in the output data set. All of the names appearing in the OUTPUT statement must be valid SAS names, and none of the new variable names can match a variable already existing in the data set to which PROC NLIN is applied.

If an observation includes a missing value for one of the independent variables, both the predicted value and the residual value are missing for that observation. If the iterations fail to converge, all the values of all the variables named in the OUTPUT statement are missing values.

You can specify the following options in the OUTPUT statement. For a description of computational formulas, see Chapter 4, "Introduction to Regression Procedures."

OUT=*SAS-data-set*
specifies the SAS data set to be created by PROC NLIN when an OUTPUT statement is included. The new data set includes the variables in the input data set. Also included are any ID variables specified in the ID statement, plus new variables with names that are specified in the OUTPUT statement.

The following values can be calculated and output to the new data set.

H=*name*

specifies a variable that contains the leverage, $\mathbf{x}_i(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_i'$, where $\mathbf{X} = \partial\mathbf{f}/\partial\boldsymbol{\beta}$ and $\mathbf{x}_i$ is the ith row of $\mathbf{X}$. If you specify the special variable _WEIGHT_, the leverage is computed as $w_i\mathbf{x}_i(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{x}_i'$.

L95=*name*

specifies a variable that contains the lower bound of an approximate 95% confidence interval for an individual prediction. This includes the variance of the error as well as the variance of the parameter estimates. See also the description for the U95= option later in this section.

L95M=*name*

specifies a variable that contains the lower bound of an approximate 95% confidence interval for the expected value (mean). See also the description for the U95M= option later in this section.

LCL=*name*

specifies a variable that contains the lower bound of an approximate $100(1-\alpha)\%$ confidence interval for an individual prediction. The α level is equal to the value of the ALPHA= option in the OUTPUT statement or, if this option is not specified, to the value of the ALPHA= option in the PROC NLIN statement. If neither of these options is specified, then $\alpha = 0.05$ by default, resulting in a lower bound for an approximate 95% confidence interval. For the corresponding upper bound, see the UCL keyword.

LCLM=*name*

specifies a variable that contains the lower bound of an approximate $100(1-\alpha)\%$ confidence interval for the expected value (mean). The α level is equal to the value of the ALPHA= option in the OUTPUT statement or, if this option is not specified, to the value of the ALPHA= option in the PROC NLIN statement. If neither of these options is specified, then $\alpha = 0.05$ by default, resulting in a lower bound for an approximate 95% confidence interval. For the corresponding lower bound, see the UCLM keyword.

PARMS=*names*

specifies variables in the output data set that contains parameter estimates. These can be the same variable names that are listed in the PARAMETERS statement; however, you can choose new names for the parameters identified in the sequence from the parameter estimates table. A note in the log indicates which variable in the output data set is associated with which parameter name. Note that, for each of these new variables, the values are the same for every observation in the new data set.

PREDICTED=*name*

P=*name*

specifies a variable in the output data set that contains the predicted values of the dependent variable.

RESIDUAL=*name*

R=*name*

specifies a variable in the output data set that contains the residuals (observed minus predicted values).

SSE=*name*
ESS=*name*
specifies a variable in the output data set that contains the residual sum of squares finally determined by the procedure. The value of the variable is the same for every observation in the new data set.

STDI=*name*
specifies a variable that contains the standard error of the individual predicted value.

STDP=*name*
specifies a variable that contains the standard error of the mean predicted value.

STDR=*name*
specifies a variable that contains the standard error of the residual.

STUDENT=*name*
specifies a variable that contains the studentized residuals. These are residuals divided by their estimated standard deviation.

U95=*name*
specifies a variable that contains the upper bound of an approximate 95% confidence interval for an individual prediction. See also the description for the L95= option.

U95M=*name*
specifies a variable that contains the upper bound of an approximate 95% confidence interval for the expected value (mean). See also the description for the L95M= option.

UCL=*name*
specifies a variable that contains the upper bound of an approximate $100(1-\alpha)\%$ confidence interval an individual prediction. The α level is equal to the value of the ALPHA= option in the OUTPUT statement or, if this option is not specified, to the value of the ALPHA= option in the PROC NLIN statement. If neither of these options is specified, then $\alpha = 0.05$ by default, resulting in an upper bound for an approximate 95% confidence interval. For the corresponding lower bound, see the LCL keyword.

UCLM=*name*
specifies a variable that contains the upper bound of an approximate $100(1-\alpha)\%$ confidence interval for the expected value (mean). The α level is equal to the value of the ALPHA= option in the OUTPUT statement or, if this option is not specified, to the value of the ALPHA= option in the PROC NLIN statement. If neither of these options is specified, then $\alpha = 0.05$ by default, resulting in an upper bound for an approximate 95% confidence interval. For the corresponding lower bound, see the LCLM keyword.

WEIGHT=*name*
specifies a variable in the output data set that contains values of the special variable _WEIGHT_.

You can specify the following options in the OUTPUT statement after a slash (/) :

ALPHA=α

specifies the level of significance α for $100(1-\alpha)\%$ confidence intervals. By default, α is equal to the value of the ALPHA= option in the PROC NLIN statement or 0.05 if that option is not specified. You can supply values that are strictly between 0 and 1.

DER

saves the first derivatives of the model with respect to the parameters to the OUTPUT data set. The derivatives are named DER_parmname, where "parmname" is the name of the model parameter in your NLIN statements. You can use the DER option to extract the $\mathbf{X} = \partial \mathbf{f} / \partial \boldsymbol{\beta}$ matrix into a SAS data set. For example, the following statements create the data set nlinX, which contains the $\mathbf{X}$ matrix:

```
proc nlin;
   parms theta1=155 theta3=0.04;
   model V = theta1*c / (theta3 + c);
   output out=nlinout / der;
run;

data nlinX;
   set nlinout(keep=DER_:);
run;
```

The derivatives are evaluated at the final parameter estimates.

PARAMETERS Statement

PARAMETERS < *parameter-specification* > < ,. . . , *parameter-specification* >
< / **PDATA=***SAS-data-set* > **;**

PARMS < *parameter-specification* > < ,. . . , *parameter-specification* >
< / **PDATA=***SAS-data-set* > **;**

A PARAMETERS (or PARMS) statement is required. The purpose of this statement is to provide starting values for the NLIN procedure. You can provide values that define a point in the parameter space or a set of points.

All parameters must be assigned starting values through the PARAMETERS statement. The NLIN procedure does not assign default starting values to parameters in your model that do not appear in the PARAMETERS statement. However, it is not necessary to supply parameters and starting values explicitly through a *parameter-specification*. Starting values can also be provided through a data set. The names assigned to parameters must be valid SAS names and must not coincide with names of variables in the input data set (see the DATA= option in the PROC NLIN statement). Parameters that are assigned starting values through the PARAMETERS statement can be omitted from the estimation, for example, if the expression in the MODEL statement does not depend on them.

Assigning Starting Values with Parameter-Specification

A *parameter-specification* has the general form

name = *value-list*

where *name* identifies the parameter and *value-list* provides the set of starting values for the parameter.

Very often, the *value-list* contains only a single value, but more general and flexible list specifications are possible:

m	a single value
$m1, m2, \ldots, mn$	several values
m TO n	a sequence where m equals the starting value, n equals the ending value, and the increment equals 1
m TO n BY i	a sequence where m equals the starting value, n equals the ending value, and the increment is i
$m1$, $m2$ TO $m3$	mixed values and sequences

When you specify more than one value for a parameter, PROC NLIN sorts the values in ascending sequence and removes duplicate values from the parameter list before forming the grid for the parameter search. If you specify several values for each parameter, PROC NLIN evaluates the model at each point on the grid. The iterations then commence from the point on the grid that yields the smallest objective function value.

For example, the following PARMS statement specifies five parameters and sets their possible starting values as shown in the table:

```
parms  b0 = 0
       b1 = 4 to 8
       b2 = 0 to .6 by .2
       b3 = 1, 10, 100
       b4 = 0, .5, 1 to 4;
```

Possible starting values

B0	B1	B2	B3	B4
0	4	0.0	1	0.0
	5	0.2	10	0.5
	6	0.4	100	1.0
	7	0.6		2.0
	8			3.0
				4.0

Residual sums of squares are calculated for each of the $1 \times 5 \times 4 \times 3 \times 6 = 360$ combinations of possible starting values. (Specifying a large grid of values can take considerable computing time.)

If you specify a starting value with a *parameter-specification*, any starting values provided for this parameter through the PDATA= data set are ignored. The *parameter-specification* overrides the information in the PDATA= data set. When you specify a BY statement, the same *parameter-specification* is applied in each BY group. To vary starting values with BY groups, use the PDATA= option in the PARAMETERS statement as described in the following paragraphs.

Assigning Starting Values from a SAS Data Set

The PDATA= option in the PARAMETERS statement enables you to assign starting values for parameters through a SAS data set. The data set must contain at least two variables that identify the parameter and contain starting values, respectively. The *character* variable identifying the parameters must be named Parameter or Parm. The *numeric* variable containing the starting value must be named Estimate or Est. This enables you, for example, to use the contents of the "ParameterEstimates" table from one PROC NLIN run to supply starting values for a subsequent run, as in the following example:

```
proc nlin data=FirstData;
   parameters alpha=100 beta=3 gamma=4;
   < other NLIN statements >
   model y = ... ;
   ods output ParameterEstimates=pest;
run;

proc nlin data=SecondData;
   parameters / pdata=pest;
   Switch = 1/(1+gamma*exp(beta*log(dose)));
   model y = alpha*Switch;
run;
```

You can specify multiple values for a parameter in the PDATA= data set, and the parameters can appear in any order. The starting values are collected by parameter and arranged in ascending order, and duplicate values are removed. The parameter names in the PDATA= data set are not case sensitive. For example, the following DATA step defines starting values for three parameters and a starting grid with $1 \times 3 \times 1 = 3$ points:

```
data test;
   input Parameter $ Estimate;
   datalines;
alpha  100
 BETA    4
 beta    4.1
beta     4.2
beta     4.1
 gamma  30
;
```

If starting values for a parameter are provided through the PDATA= data set and through an explicit *parameter-specification*, the latter takes precedence.

When you specify a BY statement, you can control whether the same starting values are applied to each BY group or whether the starting values are varied. If the BY variables are not present in the PDATA= data set, the entire contents of the PDATA= data set are applied in each BY group. If the BY variables are present in the PDATA= data set, then BY-group-specific starting values are assigned.

RETAIN Statement

RETAIN *variable* < **=***values* > < . . . *variable* < **=***values* > > **;**

The RETAIN statement declares retained variables and specifies their values. A retained variable is like a control variable (see the section "CONTROL Statement" on page 4909) except that it is retained only *within* iterations. An iteration involves a single pass through the data set.

Other Programming Statements

PROC NLIN supports many statements that are similar to SAS programming statements used in a DATA step. However, there are some differences in capabilities; for additional information, see also the section "Incompatibilities with SAS 6.11 and Earlier Versions of PROC NLIN" on page 4934.

Several SAS programming statements can be used after the PROC NLIN statement. These statements can appear anywhere in the PROC NLIN code, but new variables must be created before they appear in other statements. For example, the following statements are valid since they create the variable temp before it is used in the MODEL statement:

```
proc nlin;
   parms b0=0 to 2 by 0.5 b1=0.01 to 0.09 by 0.01;
   temp = exp(-b1*x);
   model y=b0*(1-temp);
run;
```

The following statements result in missing values for y because the variable temp is undefined before it is used:

```
proc nlin;
   parms b0=0 to 2 by 0.5 b1=0.01 to 0.09 by 0.01;
   model y = b0*(1-temp);
   temp = exp(-b1*x);
run;
```

PROC NLIN can process assignment statements, explicitly or implicitly subscripted ARRAY statements, explicitly or implicitly subscripted array references, IF statements, SAS functions, and program control statements. You can use programming statements to create new SAS variables for the duration of the procedure. These variables are not included in the data set to which PROC NLIN is applied. Program statements can include variables in the DATA= data set, parameter names, variables created by preceding programming statements within PROC NLIN, and special variables used by PROC NLIN. The following SAS programming statements can be used in PROC NLIN:

ARRAY;
variable = expression;
variable + expression;
CALL *name [(expression [, expression . . .])]*;
DO *[variable = expression*
    *[***TO** *expression] [***BY** *expression]*
    [, expression [**TO** *expression] [* **BY** *expression] . . .]*
    ]
    [**WHILE** *expression] [* **UNTIL** *expression]*;
END;
FILE;
GOTO *statement_label*;
IF *expression*;
IF *expression* **THEN** *program_statement*;
   ELSE *program_statement*;
variable = expression;
variable + expression;
LINK *statement_label*;
PUT *[variable] [=] [...]*;
RETURN;
SELECT[(*expression* **)];**

These statements can use the special variables created by PROC NLIN. See the section "Special Variables" on page 4921 for more information.

Details: NLIN Procedure

Automatic Derivatives

Depending on the optimization method you select, analytical first- and second-order derivatives are computed automatically. Derivatives can still be supplied using the DER.parm syntax. These DER.parm derivatives are not verified by the differentiator. If any needed derivatives are not supplied, they are computed and added to the programming statements. To view the computed derivatives, use the LISTDER or LIST option.

The following model is solved using Newton's method. Analytical first- and second-order derivatives are automatically computed. The compiled program code is shown in Figure 60.6.

```
proc nlin data=Enzyme method=newton list;
   parms x1=4 x2=2;
   model Velocity = x1 * exp (x2 * Concentration);
run;
```

Figure 60.6 Model and Derivative Code Output

```
                                The NLIN Procedure

                         Listing of Compiled Program Code
           Stmt     Line:Col       Statement as Parsed

              1     1057:4         MODEL.Velocity = x1 * EXP(x2
                                   * Concentration);
              1     1057:4         @MODEL.Velocity/@x1 = EXP(x2
                                   * Concentration);
              1     1057:4         @MODEL.Velocity/@x2 = x1 * Concentration
                                   * EXP(x2 * Concentration);
              1     1057:4         @@MODEL.Velocity/@x1/@x2 = Concentration
                                   * EXP(x2 * Concentration);
              1     1057:4         @@MODEL.Velocity/@x2/@x1 = Concentration
                                   * EXP(x2 * Concentration);
              1     1057:4         @@MODEL.Velocity/@x2/@x2 = x1
                                   * Concentration * Concentration
                                   * EXP(x2 * Concentration);
```

Note that all the derivatives require the evaluation of EXP(X2 * Concentration). The actual machine-level code is displayed if you specify the LISTCODE option, as in the following statements:

```
proc nlin data=Enzyme method=newton listcode;
   parms x1=4 x2=2;
   model Velocity = x1 * exp (x2 * Concentration);
run;
```

Note that, in the generated code, only one exponentiation is performed (Figure 60.7). The generated code reuses previous operations to be more efficient.

Figure 60.7 LISTCODE Output

```
                              The NLIN Procedure

                                Code Listing


     1 Stmt MODEL       line 1064 column
                        4. (1)
                        arg=MODEL.Velocity
                        argsave=MODEL.
                        Velocity
                        Source Text:           model Velocity = x1 * exp
                                               (x2 * Concentration);
         Oper *         at 1064:34 (30,0,2).   * : _temp1 <- x2 Concentration
         Oper EXP       at 1064:30             EXP : _temp2 <- _temp1
                        (103,0,1).
         Oper *         at 1064:24 (30,0,2).   * : MODEL.Velocity <- x1 _temp2
         Oper eeocf     at 1064:24 (18,0,1).   eeocf : _DER_ <- _DER_
         Oper =         at 1064:24 (1,0,1).    = : @MODEL.Velocity/@x1 <- _temp2
         Oper *         at 1064:30 (30,0,2).   * : @1dt1_1 <- Concentration _temp2
         Oper *         at 1064:24 (30,0,2).   * : @MODEL.Velocity/@x2
                                               <- x1 @1dt1_1
         Oper =         at 1064:24 (1,0,1).    = : @@MODEL.Velocity/@x1/@x2
                                               <- @1dt1_1
         Oper =         at 1064:24 (1,0,1).    = : @@MODEL.Velocity/@x2/@x1
                                               <- @1dt1_1
         Oper *         at 1064:30 (30,0,2).   * : @2dt1_1 <- Concentration
                                               @1dt1_1
         Oper *         at 1064:24 (30,0,2).   * : @@MODEL.Velocity/@x2/@x2
                                               <- x1 @2dt1_1
```

Hougaard's Measure of Skewness

A "close-to-linear" nonlinear regression model, in the sense of Ratkowsky (1983, 1990), is a model in which parameter estimators have properties similar to those in a linear regression model. That is, the least-squares estimators of the parameters are close to being unbiased and normally distributed, and they have minimum variance.

A nonlinear regression model sometimes fails to be close to linear due to the properties of a single parameter. When this occurs, bias in the parameter estimates can render inferences that use the reported standard errors and confidence limits invalid. Ratkowsky (1990) notes that of the two curvature components in a nonlinear model—intrinsic curvature and parameter-effects curvature (Bates and Watts 1981)—the parameter-effects curvature is typically larger. It is this component that you can affect by changing the parameterization of a model. One motivation for fitting a nonlinear model in a different parameterization is to obtain a particular interpretation and to give parameter estimators more close-to-linear behavior. Example 60.4 shows how changing the parameterization of a four-parameter logistic model can reduce the parameter-effects curvature and can yield a useful parameter interpretation at the same time.

The degree to which a parameter estimator exhibits close-to-linear behavior can be assessed with Hougaard's measure of skewness, g_{1i} (Hougaard 1982, 1985). This measure is available through the HOUGAARD option in the PROC NLIN statement. Hougaard's skewness measure for the ith parameter is defined as follows:

$$\mathrm{E}\left[\widehat{\theta}_i - \mathrm{E}\left(\widehat{\theta}_i\right)\right]^3 = -\left(\sigma^2\right)^2 \sum_{jkl} [\mathbf{L}]_{ij}[\mathbf{L}]_{ik}[\mathbf{L}]_{il}\left(\left[\mathbf{W}_j\right]_{kl} + [\mathbf{W}_k]_{jl} + [\mathbf{W}_l]_{jk}\right)$$

where the sum is a triple sum over the number of parameters and

$$\mathbf{L} = \left(\mathbf{X}'\mathbf{X}\right)^{-1} = \left(\frac{\partial \mathbf{f}}{\partial \boldsymbol{\beta}'}\frac{\partial \mathbf{f}}{\partial \boldsymbol{\beta}}\right)^{-1}$$

The term $[\mathbf{L}]_{ij}$ denotes the value in row i, column j of the matrix $\mathbf{L}$. (Hougaard (1985) uses superscript notation to denote elements in this inverse.) The matrix $\mathbf{W}$ is a three-dimensional $(p \times p \times p)$ array

$$\mathbf{W} = \left(\frac{\partial \mathbf{f}}{\partial \boldsymbol{\beta}'}\right)\mathbf{H}$$

$$\mathbf{H} = \frac{\partial^2 \mathbf{f}}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}'}$$

$$\left[\mathbf{W}_j\right]_{kl} = \sum_{m=1}^{n} \frac{\partial F_m}{\partial \beta_j}\frac{\partial^2 F_m}{\partial \beta_k \partial \beta_l}$$

The third central moment is then normalized using the standard error as

$$G_{1i} = \mathrm{E}\left[\widehat{\theta}_i - E(\widehat{\theta}_i)\right]^3 / \left(\sigma^2 \times [\mathbf{L}]_{ii}\right)^{3/2}$$

The previous expressions depend on the unknown values of the parameters and on the residual variance σ^2. In order to evaluate the Hougaard measure in a particular data set, the NLIN procedure computes

$$g_{1i} = \widehat{\mathrm{E}}\left[\widehat{\theta}_i - \mathrm{E}(\widehat{\theta}_i)\right]^3 / \left(\mathrm{mse} \times [\widehat{\mathbf{L}}]_{ii}\right)^{3/2}$$

$$\widehat{\mathrm{E}}\left[\widehat{\theta}_i - \mathrm{E}(\widehat{\theta}_i)\right]^3 = -\mathrm{mse}^2 \sum_{jkl} [\widehat{\mathbf{L}}]_{ij}[\widehat{\mathbf{L}}]_{ik}[\widehat{\mathbf{L}}]_{il}\left([\widehat{\mathbf{W}}_j]_{kl} + [\widehat{\mathbf{W}}_k]_{jl} + [\widehat{\mathbf{W}}_l]_{jk}\right)$$

Following Ratkowsky (1990, p. 28), the following rules can be applied to the standardized Hougaard measure computed by the NLIN procedure:

$\lvert g_{1i}\rvert < 0.1$	the estimator $\widehat{\theta}_i$ of parameter θ_i is very close to linear in behavior
$0.1 < \lvert g_{1i}\rvert < .25$	the estimator is reasonably close to linear
$\lvert g_{1i}\rvert > .25$	the skewness is very apparent
$\lvert g_{1i}\rvert > 1$	the nonlinear behavior is considerable

Missing Values

If the value of any one of the SAS variables involved in the model is missing from an observation, that observation is omitted from the analysis. If only the value of the dependent variable is missing, that observation has a predicted value calculated for it when you use an OUTPUT statement and specify the PREDICTED= option.

If an observation includes a missing value for one of the independent variables, both the predicted value and the residual value are missing for that observation. If the iterations fail to converge, the values for all variables named in the OUTPUT statement are set to missing.

Special Variables

Several special variables are created automatically and can be used in PROC NLIN programming statements.

Special Variables with Values That Are Set by PROC NLIN

The values of the following six special variables are set by PROC NLIN and should not be reset to a different value by programming statements:

ERROR	is set to 1 if a numerical error or invalid argument to a function occurs during the current execution of the program. It is reset to 0 before each new execution.
ITER	represents the current iteration number. The variable _ITER_ is set to -1 during the grid search phase.
MODEL	is set to 1 for passes through the data when only the predicted values are needed, and not the derivatives. It is 0 when both predicted values and derivatives are needed. If your derivative calculations consume a lot of time, you can save resources by using the following statement after your MODEL statement but before your derivative calculations: `if _model_ then return;` The derivatives generated by PROC NLIN do this automatically.
_N_	indicates the number of times the PROC NLIN step has been executed. It is never reset for successive passes through the data set.
OBS	indicates the observation number in the data set for the current program execution. It is reset to 1 to start each pass through the data set (unlike the _N_ variable).
SSE	has the error sum of squares of the last iteration. During the grid search phase, the _SSE_ variable is set to 0. For iteration 0, the _SSE_ variable is set to the SSE associated with the point chosen from the grid search.

Special Variable Used to Determine Convergence Criteria

The special variable _LOSS_ can be used to determine the criterion function for convergence and step shortening. PROC NLIN looks for the variable _LOSS_ in the programming statements and, if it is defined, uses the (weighted) sum of this value instead of the residual sum of squares to determine the criterion function for convergence and step shortening. This feature is useful in certain types of maximum-likelihood estimation.

CAUTION: Even if you specify the _LOSS_ variable in your programming statements, the NLIN procedure continues to solve a least-squares problem. The specified _LOSS_ function does **not** define or alter the objective function for parameter estimation.

Weighted Regression with the Special Variable _WEIGHT_

To obtain weighted nonlinear least-squares estimates of parameters, make an assignment to the _WEIGHT_ variable as in the following statement:

```
_weight_ = expression;
```

When this statement is included, the expression on the right side of the assignment statement is evaluated for each observation in the data set. The values are then taken as inverse elements of the diagonal variance-covariance matrix of the dependent variable.

When a variable name is given after the equal sign, the values of the variable are taken as the inverse elements of the variance-covariance matrix. The larger the _WEIGHT_ value, the more importance the observation is given.

The _WEIGHT_ variable can be a function of the estimated parameters. For estimation purposes, the derivative of the _WEIGHT_ variable with respect to the parameters is not included in the gradient and the Hessian of the loss function. This is normally the desired approach for iteratively reweighted least-squares estimation. When the _WEIGHT_ variable is a function of the parameters, the gradient and the Hessian used can lead to poor convergence or nonconvergence of the requested estimation. To have the derivative of the _WEIGHT_ variable with respect to the parameters included in the gradient and the Hessian of the loss function, do not use the _WEIGHT_ variable. Instead, redefine the model as

$$(y - f(x, \beta)) \times \sqrt{wgt(\beta)}$$

where y is the original dependent variable, $f(x, \beta)$ is the nonlinear model, and $wgt(\beta)$ is the weight that is a function of the parameters.

If the _WEIGHT_= statement is not used, the default value of 1 is used, and regular least-squares estimates are obtained.

Troubleshooting

This section describes a number of problems that might occur in your analysis with PROC NLIN.

Excessive Computing Time

If you specify a grid of starting values that contains many points, the analysis might take excessive time since the procedure must go through the entire data set for each point on the grid.

The analysis might also take excessive time if your problem takes many iterations to converge, since each iteration requires as much time as a linear regression with predicted values and residuals calculated.

Dependencies

The matrix of partial derivatives can be singular, possibly indicating an overparameterized model. For example, if b0 starts at zero in the following model, the derivatives for b1 are all zero for the first iteration:

```
parms b0=0 b1=.022;
model pop=b0*exp(b1*(year-1790));
der.b0=exp(b1*(year-1790));
der.b1=(year-1790)*b0*exp(b1*(year-1790));
```

The first iteration changes a subset of the parameters; then the procedure can make progress in succeeding iterations. This singularity problem is local. The next example displays a global problem. The term b2 in the exponent is not identifiable since it trades roles with b0.

```
parms b0=3.9 b1=.022 b2=0;
model pop=b0*exp(b1*(year-1790)+b2);
der.b0 = exp(b1*(year-1790)+b2);
der.b1 = (year-1790)*b0*exp(b1*(year-1790)+b2);
der.b2 = b0*exp(b1*(year-1790)+b2);
```

Unable to Improve

The method can lead to steps that do not improve the estimates even after a series of step halvings. If this happens, the procedure issues a message stating that it is unable to make further progress, but it then displays the following warning message:

```
PROC NLIN failed to converge
```

Then it displays the results. This often means that the procedure has not converged at all. If you provided your own derivatives, check them carefully and then examine the residual sum of squares surface. If PROC NLIN has not converged, try a different set of starting values, a different METHOD= specification, the G4 option, or a different model.

Divergence

The iterative process might diverge, resulting in overflows in computations. It is also possible that parameters enter a space where arguments to such functions as LOG and SQRT become invalid. For example, consider the following model:

```
parms b=0;
model y = x / b;
```

Suppose that y contains only zeros, and suppose that the values for variable x are not zero. There is no least-squares estimate for b since the SSE declines as b approaches infinity or minus infinity. To avoid the problem, the same model could be parameterized as **y = a*x**.

If you have divergence problems, try reparameterizing the model, selecting different starting values, increasing the maximum allowed number of iterations (the MAXITER= option), specifying an alternative METHOD= option, or including a BOUNDS statement.

Local Minimum

The program might converge to a local rather than a global minimum. For example, consider the following model:

```
parms a=1 b=-1;
model y=(1-a*x)*(1-b*x);
```

Once a solution is found, an equivalent solution with the same SSE can be obtained by swapping the values of a and b.

Discontinuities

The computational methods assume that the model is a continuous and smooth function of the parameters. If this is not true, the method does not work. For example, the following models do not work:

```
model y=a+int(b*x);

model y=a+b*x+4*(z>c);
```

Responding to Trouble

PROC NLIN does not necessarily produce a good solution the first time. Much depends on specifying good initial values for the parameters. You can specify a grid of values in the PARMS statement to search for good starting values. While most practical models should give you no trouble, other models can require switching to a different iteration method or a different computational method for matrix inversion. Specifying the option METHOD=MARQUARDT sometimes works when the default method (Gauss-Newton) does not work.

Computational Methods

Nonlinear Least Squares

Recall the basic notation for the nonlinear regression model from the section "Notation for Nonlinear Regression Models" on page 4895. The parameter vector $\boldsymbol{\beta}$ belongs to $\boldsymbol{\Omega}$, a subset of R^p. Two points of this set are of particular interest: the true value $\widetilde{\boldsymbol{\beta}}$ and the least-squares estimate $\widehat{\boldsymbol{\beta}}$. The general nonlinear model fit with the NLIN procedure is represented by the equation

$$\mathbf{Y} = \mathbf{f}(\widetilde{\beta_0}, \widetilde{\beta_1}, \ldots, \widetilde{\beta_p}; \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k) + \boldsymbol{\epsilon} = \mathbf{f}(\widetilde{\boldsymbol{\beta}}; \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k) + \boldsymbol{\epsilon}$$

where $\mathbf{z}_j$ denotes the $(n \times 1)$ vector of the jth regressor (independent) variable, $\widetilde{\boldsymbol{\beta}}$ is the true value of the parameter vector, and $\boldsymbol{\epsilon}$ is the $(n \times 1)$ vector of homoscedastic and uncorrelated model errors with zero mean.

To write the model for the ith observation, the ith elements of $\mathbf{z}_1, \cdots, \mathbf{z}_k$ are collected in the row vector $\mathbf{z}_i'$, and the model equation becomes

$$Y_i = f(\boldsymbol{\beta}; \mathbf{z}_i') + \epsilon_i$$

The shorthand $f_i(\boldsymbol{\beta})$ will also be used throughout to denote the mean of the ith observation.

For any given value $\boldsymbol{\beta}$ we can compute the residual sum of squares

$$\begin{aligned} \mathrm{SSE}(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \left(y_i - f\left(\boldsymbol{\beta}; \mathbf{z}_i'\right)\right)^2 \\ &= \sum_{i=1}^{n} \left(y_i - f_i\left(\boldsymbol{\beta}\right)\right)^2 = \mathbf{r}(\boldsymbol{\beta})'\mathbf{r}(\boldsymbol{\beta}) \end{aligned}$$

The aim of nonlinear least-squares estimation is to find the value $\widehat{\boldsymbol{\beta}}$ that minimizes $\mathrm{SSE}(\boldsymbol{\beta})$. Because $\mathbf{f}$ is a nonlinear function of $\boldsymbol{\beta}$, a closed-form solution does not exist for this minimization problem. An iterative process is used instead. The iterative techniques that PROC NLIN uses are similar to a series of linear regressions involving the matrix $\mathbf{X}$ and the residual vector $\mathbf{r} = \mathbf{y} - \mathbf{f}(\boldsymbol{\beta})$, evaluated at the current values of $\boldsymbol{\beta}$.

It is more insightful, however, to describe the algorithms in terms of their approach to minimizing the residual sum of squares and in terms of their updating formulas. If $\widehat{\boldsymbol{\beta}}^{(u)}$ denotes the value of the parameter estimates at the uth iteration, and $\widehat{\boldsymbol{\beta}}^{(0)}$ are your starting values, then the NLIN procedure attempts to find values k and $\boldsymbol{\Delta}$ such that

$$\widehat{\boldsymbol{\beta}}^{(u+1)} = \widehat{\boldsymbol{\beta}}^{(u)} + k\boldsymbol{\Delta}$$

and

$$\mathrm{SSE}\left(\widehat{\boldsymbol{\beta}}^{(u+1)}\right) < \mathrm{SSE}\left(\widehat{\boldsymbol{\beta}}^{(u)}\right)$$

The various methods to fit a nonlinear regression model—which you can select with the METHOD= option in the PROC NLIN statement—differ in the calculation of the update vector $\boldsymbol{\Delta}$.

The gradient and Hessian of the residual sum of squares with respect to individual parameters and pairs of parameters are, respectively,

$$\begin{aligned} \mathbf{g}(\beta_j) &= \frac{\partial \mathrm{SSE}(\boldsymbol{\beta})}{\partial \beta_j} = -2\sum_{i=1}^{n} (y_i - f_i(\boldsymbol{\beta}))\,\frac{\partial f_i(\boldsymbol{\beta})}{\partial \beta_j} \\ [\mathbf{H}(\boldsymbol{\beta})]_{jk} &= \frac{\partial \mathrm{SSE}(\boldsymbol{\beta})}{\partial \beta_j \partial \beta_k} = 2\sum_{i=1}^{n} \frac{\partial f_i(\boldsymbol{\beta})}{\partial \beta_j}\frac{\partial f_i(\boldsymbol{\beta})}{\partial \beta_k} - (y_i - f_i(\boldsymbol{\beta}))\,\frac{\partial^2 f_i(\boldsymbol{\beta})}{\partial \beta_j \partial \beta_k} \end{aligned}$$

Denote as $\mathbf{H}_i^*(\boldsymbol{\beta})$ the Hessian matrix of the mean function,

$$\left[\mathbf{H}_i^*(\boldsymbol{\beta})\right]_{jk} = \left[\frac{\partial^2 f_i(\boldsymbol{\beta})}{\partial \beta_j \partial \beta_k}\right]_{jk}$$

Collecting the derivatives across all parameters leads to the expressions

$$\begin{aligned} \mathbf{g}(\boldsymbol{\beta}) &= \frac{\partial \mathrm{SSE}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -2\mathbf{X}'\mathbf{r}(\boldsymbol{\beta}) \\ \mathbf{H}(\boldsymbol{\beta}) &= \frac{\partial^2 \mathrm{SSE}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}'} = 2\left(\mathbf{X}'\mathbf{X} - \sum_{i=1}^{n} r_i(\boldsymbol{\beta})\mathbf{H}_i^*(\boldsymbol{\beta})\right) \end{aligned}$$

The change in the vector of parameter estimates is computed as follows, depending on the estimation method:

$$\begin{aligned} \text{Gauss-Newton: } \boldsymbol{\Delta} &= (-\mathrm{E}[\mathbf{H}(\boldsymbol{\beta})])^{-}\,\mathbf{g}(\boldsymbol{\beta}) = (\mathbf{X}'\mathbf{X})^{-}\mathbf{X}'\mathbf{r} \\ \text{Marquardt: } \boldsymbol{\Delta} &= \left(\mathbf{X}'\mathbf{X} + \lambda\mathrm{diag}(\mathbf{X}'\mathbf{X})\right)^{-}\mathbf{X}'\mathbf{r} \\ \text{Newton: } \boldsymbol{\Delta} &= -\mathbf{H}(\boldsymbol{\beta})^{-}\mathbf{g}(\boldsymbol{\beta}) = \mathbf{H}(\boldsymbol{\beta})^{-}\mathbf{X}'\mathbf{r} \\ \text{Steepest descent: } \boldsymbol{\Delta} &= -\frac{1}{2}\frac{\partial \mathrm{SSE}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \mathbf{X}'\mathbf{r} \end{aligned}$$

The Gauss-Newton and Marquardt iterative methods regress the residuals onto the partial derivatives of the model with respect to the parameters until the estimates converge. You can view the Marquardt algorithm as a Gauss-Newton algorithm with a ridging penalty. The Newton iterative method regresses the residuals onto a function of the first and second derivatives of the model with respect to the parameters until the estimates converge. Analytical first- and second-order derivatives are automatically computed as needed.

The default method used to compute $(\mathbf{X}'\mathbf{X})^{-}$ is the sweep (Goodnight 1979). It produces a reflexive generalized inverse (a g_2-inverse, Pringle and Rayner, 1971). In some cases it might be preferable to use a Moore-Penrose inverse (a g_4-inverse) instead. If you specify the G4 option in the PROC NLIN statement, a g_4-inverse is used to calculate $\boldsymbol{\Delta}$ on each iteration.

The four algorithms are now described in greater detail.

Algorithmic Details

Gauss-Newton and Newton Methods

From the preceding set of equations you can see that the Marquardt method is a ridged version of the Gauss-Newton method. If the ridge parameter λ equals zero, the Marquardt step is identical to the Gauss-Newton step. An important difference between the Newton methods and the Gauss-Newton-type algorithms lies in the use of second derivatives. To motivate this distinctive element between Gauss-Newton and the Newton method, focus first on the objective function in nonlinear least squares. To numerically find the minimum of

$$\mathrm{SSE}(\boldsymbol{\beta}) = \mathbf{r}(\boldsymbol{\beta})'\mathbf{r}(\boldsymbol{\beta})$$

you can approach the problem by approximating the sum of squares criterion by a criterion for which you can compute a closed-form solution. Following Seber and Wild (1989, Sect. 2.1.3), we can achieve that by doing the following:

- approximating the model and substituting the approximation into $\mathrm{SSE}(\boldsymbol{\beta})$
- approximating $\mathrm{SSE}(\boldsymbol{\beta})$ directly

The first method, approximating the nonlinear model with a first-order Taylor series, is the purview of the Gauss-Newton method. Approximating the residual sum of squares directly is the realm of the Newton method.

The first-order Taylor series of the residual $\mathbf{r}(\boldsymbol{\beta})$ at the point $\widehat{\boldsymbol{\beta}}$ is

$$\mathbf{r}(\boldsymbol{\beta}) \approx \mathbf{r}(\widehat{\boldsymbol{\beta}}) - \widehat{\mathbf{X}}\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right)$$

Substitution into $\mathrm{SSE}(\boldsymbol{\beta})$ leads to the objective function for the Gauss-Newton step:

$$\mathrm{SSE}\left(\boldsymbol{\beta}\right) \approx S_G\left(\boldsymbol{\beta}\right) = \mathbf{r}(\widehat{\boldsymbol{\beta}})' - 2\mathbf{r}'(\widehat{\boldsymbol{\beta}})\widehat{\mathbf{X}}\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right) + \left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right)'\left(\widehat{\mathbf{X}}'\widehat{\mathbf{X}}\right)\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right)$$

"Hat" notation is used here to indicate that the quantity in question is evaluated at $\widehat{\boldsymbol{\beta}}$.

To motivate the Newton method, take a second-order Taylor series of $\mathrm{SSE}(\boldsymbol{\beta})$ around the value $\widehat{\boldsymbol{\beta}}$:

$$\mathrm{SSE}\left(\boldsymbol{\beta}\right) \approx S_N\left(\boldsymbol{\beta}\right) = \mathrm{SSE}\left(\widehat{\boldsymbol{\beta}}\right) + \mathbf{g}(\widehat{\boldsymbol{\beta}})'\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right) + \frac{1}{2}\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right)'\mathbf{H}(\widehat{\boldsymbol{\beta}})\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right)$$

Both $S_G\left(\boldsymbol{\beta}\right)$ and $S_N\left(\boldsymbol{\beta}\right)$ are quadratic functions in $\boldsymbol{\beta}$ and are easily minimized. The minima occur when

$$\text{Gauss-Newton: } \boldsymbol{\beta} - \widehat{\boldsymbol{\beta}} = \left(\widehat{\mathbf{X}}'\widehat{\mathbf{X}}\right)^{-}\widehat{\mathbf{X}}'\mathbf{r}(\widehat{\boldsymbol{\beta}})$$

$$\text{Newton : } \boldsymbol{\beta} - \widehat{\boldsymbol{\beta}} = -\mathbf{H}(\widehat{\boldsymbol{\beta}})^{-}\mathbf{g}(\widehat{\boldsymbol{\beta}})$$

and these terms define the preceding $\boldsymbol{\Delta}$ update vectors.

Gauss-Newton Method

Since the Gauss-Newton method is based on an approximation of the model, you can also derive the update vector by first considering the "normal" equations of the nonlinear model

$$\mathbf{X}'\mathbf{f}(\boldsymbol{\beta}) = \mathbf{X}'\mathbf{Y}$$

and then substituting the Taylor approximation

$$\mathbf{f}(\boldsymbol{\beta}) \approx \mathbf{f}(\widehat{\boldsymbol{\beta}}) + \mathbf{X}\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right)$$

for $\mathbf{f}(\boldsymbol{\beta})$. This leads to

$$\begin{aligned} \mathbf{X}'\left(\mathbf{f}(\widehat{\boldsymbol{\beta}}) + \mathbf{X}(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}})\right) &= \mathbf{X}'\mathbf{Y} \\ (\mathbf{X}'\mathbf{X})(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}) &= \mathbf{X}'\mathbf{Y} - \mathbf{X}'\mathbf{f}(\widehat{\boldsymbol{\beta}}) \\ (\mathbf{X}'\mathbf{X})\boldsymbol{\Delta} &= \mathbf{X}'\mathbf{r}(\widehat{\boldsymbol{\beta}}) \end{aligned}$$

and the update vector becomes

$$\boldsymbol{\Delta} = \left(\mathbf{X}'\mathbf{X}\right)^{-}\mathbf{X}'\mathbf{r}(\widehat{\boldsymbol{\beta}})$$

CAUTION: If $\mathbf{X}'\mathbf{X}$ is singular or becomes singular, PROC NLIN computes $\boldsymbol{\Delta}$ by using a generalized inverse for the iterations after singularity occurs. If $\mathbf{X}'\mathbf{X}$ is still singular for the last iteration, the solution should be examined.

Newton Method

The Newton method uses the second derivatives and solves the equation

$$\boldsymbol{\Delta} = \mathbf{H}^{-}\mathbf{X}'\mathbf{r}$$

If the automatic variables _WEIGHT_, _WGTJPJ_, and _RESID_ are used, then

$$\boldsymbol{\Delta} = \mathbf{H}^{-}\mathbf{X}'\mathbf{W}^{SSE}\mathbf{r}^{*}$$

is the direction, where

$$\mathbf{H} = \mathbf{X}'\mathbf{W}^{XPX}\mathbf{X} - \sum_{i=1}^{n}\mathbf{H}_i^{*}(\boldsymbol{\beta})w_i^{XPX}r_i^{*}$$

and

$\mathbf{W}^{SSE}$	is an $n \times n$ diagonal matrix with elements w_i^{SSE} of weights from the _WEIGHT_ variable. Each element w_i^{SSE} contains the value of _WEIGHT_ for the ith observation.
$\mathbf{W}^{XPX}$	is an $n \times n$ diagonal matrix with elements w_i^{XPX} of weights from the _WGTJPJ_ variable. Each element w_i^{XPX} contains the value of _WGTJPJ_ for the ith observation.
$\mathbf{r}^{*}$	is a vector with elements r_i^{*} from the _RESID_ variable. Each element r_i^{*} contains the value of _RESID_ evaluated for the ith observation.

Marquardt Method

The updating formula for the Marquardt method is as follows:

$$\mathbf{\Delta} = (\mathbf{X}'\mathbf{X} + \lambda \mathrm{diag}(\mathbf{X}'\mathbf{X}))^{-}\mathbf{X}'\mathbf{e}$$

The Marquardt method is a compromise between the Gauss-Newton and steepest descent methods (Marquardt 1963). As $\lambda \rightarrow 0$, the direction approaches Gauss-Newton. As $\lambda \rightarrow \infty$, the direction approaches steepest descent.

Marquardt's studies indicate that the average angle between Gauss-Newton and steepest descent directions is about 90°. A choice of λ between 0 and infinity produces a compromise direction.

By default, PROC NLIN chooses $\lambda = 10^{-7}$ to start and computes $\mathbf{\Delta}$. If $\mathrm{SSE}(\boldsymbol{\beta}_0 + \mathbf{\Delta}) < \mathrm{SSE}(\boldsymbol{\beta}_0)$, then $\lambda = \lambda/10$ for the next iteration. Each time $\mathrm{SSE}(\boldsymbol{\beta}_0 + \mathbf{\Delta}) > \mathrm{SSE}(\boldsymbol{\beta}_0)$, then $\lambda = 10\lambda$.

NOTE: If the SSE decreases on each iteration, then $\lambda \rightarrow 0$, and you are essentially using the Gauss-Newton method. If SSE does not improve, then λ is increased until you are moving in the steepest descent direction.

Marquardt's method is equivalent to performing a series of ridge regressions, and it is useful when the parameter estimates are highly correlated or the objective function is not well approximated by a quadratic.

Steepest Descent (Gradient) Method

The steepest descent method is based directly on the gradient of $0.5\mathbf{r}(\boldsymbol{\beta})'\mathbf{r}(\boldsymbol{\beta})$:

$$\frac{1}{2}\frac{\partial \mathrm{SSE}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -\mathbf{X}'\mathbf{r}$$

The quantity $-\mathbf{X}'\mathbf{r}$ is the gradient along which $\boldsymbol{\epsilon}'\boldsymbol{\epsilon}$ increases. Thus $\mathbf{\Delta} = \mathbf{X}'\mathbf{r}$ is the direction of steepest descent.

If the automatic variables _WEIGHT_ and _RESID_ are used, then

$$\mathbf{\Delta} = \mathbf{X}'\mathbf{W}^{SSE}\mathbf{r}^*$$

is the direction, where

$\mathbf{W}^{SSE}$	is an $n \times n$ diagonal matrix with elements w_i^{SSE} of weights from the _WEIGHT_ variable. Each element w_i^{SSE} contains the value of _WEIGHT_ for the ith observation.
$\mathbf{r}^*$	is a vector with elements r_i^* from _RESID_. Each element r_i^* contains the value of _RESID_ evaluated for the ith observation.

Using the method of steepest descent, let

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \alpha\mathbf{\Delta}$$

where the scalar α is chosen such that

$$\mathrm{SSE}(\boldsymbol{\beta}_i + \alpha\mathbf{\Delta}) < \mathrm{SSE}(\boldsymbol{\beta}_i)$$

CAUTION: The steepest-descent method can converge very slowly and is therefore not generally recommended. It is sometimes useful when the initial values are poor.

Step-Size Search

The default method of finding the step size k is step halving by using SMETHOD=HALVE. If $\text{SSE}(\boldsymbol{\beta}^{(u)} + \boldsymbol{\Delta}) > \text{SSE}(\boldsymbol{\beta}^{(u)})$, compute $\text{SSE}(\boldsymbol{\beta}^{(u)} + 0.5\boldsymbol{\Delta})$, $\text{SSE}(\boldsymbol{\beta}^{(u)} + 0.25\boldsymbol{\Delta}), \ldots,$ until a smaller SSE is found.

If you specify SMETHOD=GOLDEN, the step size k is determined by a golden section search. The parameter TAU determines the length of the initial interval to be searched, with the interval having length TAU (or $2 \times$ TAU), depending on $\text{SSE}(\boldsymbol{\beta}^{(u)} + \boldsymbol{\Delta})$. The RHO parameter specifies how fine the search is to be. The SSE at each endpoint of the interval is evaluated, and a new subinterval is chosen. The size of the interval is reduced until its length is less than RHO. One pass through the data is required each time the interval is reduced. Hence, if RHO is very small relative to TAU, a large amount of time can be spent determining a step size. For more information about the golden section search, see Kennedy and Gentle (1980).

If you specify SMETHOD=CUBIC, the NLIN procedure performs a cubic interpolation to estimate the step size. If the estimated step size does not result in a decrease in SSE, step halving is used.

Output Data Sets

The data set produced by the OUTEST= option in the PROC NLIN statement contains the parameter estimates on each iteration, including the grid search.

The variable _ITER_ contains the iteration number. The variable _TYPE_ denotes whether the observation contains iteration parameter estimates (“ITER”), final parameter estimates (“FINAL”), or covariance estimates (“COVB”). The variable _NAME_ contains the parameter name for covariances, and the variable _SSE_ contains the objective function value for the parameter estimates. The variable _STATUS_ indicates whether the estimates have converged.

The data set produced by the OUTPUT statement contains statistics calculated for each observation. In addition, the data set contains the variables from the input data set and any ID variables that are specified in the ID statement.

Confidence Intervals

Parameter Confidence Intervals

The parameter confidence intervals are computed using the Wald-based formula:

$$\widehat{\beta}_i \pm \text{stderr}_i \times t(n-p, 1-\alpha/2)$$

where $\widehat{\beta}_i$ is the ith parameter estimate, stderr_i is its estimated approximate standard error, $t(n - p, 1 - \alpha/2)$ is a t statistic with $n - p$ degrees of freedom, n is the number of observations, and p is the number of parameters. The confidence intervals are only asymptotically valid. The significance level α used in the construction of these confidence limits can be set with the ALPHA= option in the PROC NLIN statement; the default value is $\alpha = 0.05$.

Model Confidence Intervals

Model confidence intervals are output when an OUT= data set is specified and one or more of the keywords LCLM, UCLM, LCL, UCL, L95M=, U95M=, L95=, and U95= is specified. The expressions for these terms are as follows:

$$\begin{aligned}
\text{LCLM} &= f(\boldsymbol{\beta}, \mathbf{z}_i) - \sqrt{MSE \times h_i/w_i} \times t(n-p, 1-\alpha/2) \\
\text{UCLM} &= f(\boldsymbol{\beta}, \mathbf{z}_i) + \sqrt{MSE \times h_i/w_i} \times t(n-p, 1-\alpha/2) \\
\text{LCL} &= f(\boldsymbol{\beta}, \mathbf{z}_i) - \sqrt{MSE(h_i + 1/w_i)} \times t(n-p, 1-\alpha/2) \\
\text{UCL} &= f(\boldsymbol{\beta}, \mathbf{z}_i) + \sqrt{MSE(h_i + 1/w_i)} \times t(n-p, 1-\alpha/2) \\
\text{L95M} &= f(\boldsymbol{\beta}, \mathbf{z}_i) - \sqrt{MSE \times h_i/w_i} \times t(n-p, 1-0.05/2) \\
\text{U95M} &= f(\boldsymbol{\beta}, \mathbf{z}_i) + \sqrt{MSE \times h_i/w_i} \times t(n-p, 1-0.05/2) \\
\text{L95} &= f(\boldsymbol{\beta}, \mathbf{z}_i) - \sqrt{MSE(h_i + 1/w_i)} \times t(n-p, 1-0.05/2) \\
\text{U95} &= f(\boldsymbol{\beta}, \mathbf{z}_i) + \sqrt{MSE(h_i + 1/w_i)} \times t(n-p, 1-0.05/2)
\end{aligned}$$

where $h_i = w_i \mathbf{x}_i (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{x}_i'$ is the leverage, $\mathbf{X} = \partial\mathbf{f}/\partial\boldsymbol{\beta}$, and $\mathbf{x}_i$ is the ith row of $\mathbf{X}$. These results are derived for linear systems. The intervals are approximate for nonlinear models. The value α in the preceding formulas for LCLM, UCLM, LCL, and UCL can be set with the ALPHA= option in the PROC NLIN statement or with the ALPHA= option in the OUTPUT statement. If both ALPHA= options are specified, the option in the OUTPUT takes precedence.

Covariance Matrix of Parameter Estimates

For unconstrained estimates (no active bounds), the covariance matrix of the parameter estimates is

$$\text{mse} \times (\mathbf{X}'\mathbf{X})^{-1}$$

for the gradient, Marquardt, and Gauss methods and

$$\text{mse} \times \mathbf{H}^{-1}$$

for the Newton method. Recall that $\mathbf{X}$ is the matrix of the first partial derivatives of the nonlinear model with respect to the parameters. The matrices are evaluated at the final parameter estimates. The mean squared error, the estimate of the residual variance σ^2, is computed as

$$\text{mse} = \mathbf{r}'\mathbf{r}/(n-p)$$

where n is the number of nonmissing (used) observations and p is the number of estimable parameters. The standard error reported for the parameter estimates is the square root of the corresponding diagonal element of this matrix. If you specify a value for the residual variance with the SIGSQ= option, then that value replaces mse in the preceding expressions.

Now suppose that restrictions or bounds are active. Equality restrictions can be written as a vector function, $h(\boldsymbol{\theta}) = \mathbf{0}$. Inequality restrictions are either active or inactive. When an inequality restriction is active, it is treated as an equality restriction.

Assume that the vector $h(\boldsymbol{\theta})$ contains the current active restrictions. The constraint matrix $\mathbf{A}$ is then

$$\mathbf{A}(\widehat{\boldsymbol{\theta}}) = \frac{\partial h(\widehat{\boldsymbol{\theta}})}{\partial \widehat{\boldsymbol{\theta}}}$$

The covariance matrix for the restricted parameter estimates is computed as

$$\mathbf{Z}(\mathbf{Z}'\mathbf{H}\mathbf{Z})^{-1}\mathbf{Z}'$$

where $\mathbf{H}$ is the Hessian (or approximation to the Hessian) and $\mathbf{Z}$ collects the last $(p - n_c)$ columns of $\mathbf{Q}$ from an LQ factorization of the constraint matrix. Further, n_c is the number of active constraints, and p denotes the number of parameters. See Gill, Murray, and Wright (1981) for more details about the LQ factorization. The covariance matrix for the Lagrange multipliers is computed as

$$\left(\mathbf{A}\mathbf{H}^{-1}\mathbf{A}'\right)^{-1}$$

Convergence Measures

The NLIN procedure computes and reports four convergence measures, labeled R, PPC, RPC, and OBJECT.

R is the primary convergence measure for the parameters. It measures the degree to which the residuals are orthogonal to the columns of $\mathbf{X}$, and it approaches 0 as the gradient of the objective function becomes small. R is defined as

$$\sqrt{\frac{\mathbf{r}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{r}}{\mathrm{LOSS}^i}}$$

PPC is the prospective parameter change measure. PPC measures the maximum relative change in the parameters implied by the parameter-change vector computed for the next iteration. At the kth iteration, PPC is the maximum over the parameters

$$\frac{|\widehat{\theta}_i^{(k+1)} - \widehat{\theta}_i^{(k)}|}{|\widehat{\theta}_i^{(k)}| + 1\mathrm{E}-6}$$

where $\widehat{\theta}_i^{(k)}$ is the current value of the ith parameter and $\widehat{\theta}_i^{(k+1)}$ is the prospective value of this parameter after adding the change vector computed for the next iteration. These changes are measured before step length adjustments are made. The parameter with the maximum prospective relative change is displayed with the value of PPC, unless the PPC is nearly 0.

RPC is the retrospective parameter change measure. RPC measures the maximum relative change in the parameters from the previous iteration. At the kth iteration, RPC is the maximum over i of

$$\frac{|\widehat{\theta}_i^{(k)} - \widehat{\theta}_i^{(k-1)}|}{|\widehat{\theta}_i^{(k-1)}| + 1\text{E} - 6}$$

where $\widehat{\theta}_i^{(k)}$ is the current value of the ith parameter and $\widehat{\theta}_i^{k-1}$ is the previous value of this parameter. These changes are measured before step length adjustments are made. The name of the parameter with the maximum retrospective relative change is displayed with the value of RPC, unless the RPC is nearly 0.

OBJECT measures the relative change in the objective function value between iterations:

$$\frac{|O^{(k)} - O^{(k-1)}|}{|O^{(k-1)} + 1\text{E} - 6|}$$

where $O^{(k-1)}$ is the value of the objective function ($O^{(k)}$) from the previous iteration. This is the old CONVERGEOBJ= criterion.

Displayed Output

In addition to the output data sets, PROC NLIN also produces the following output objects:

- the residual sums of squares associated with all or some of the combinations of possible starting values of the parameters
- the estimates of the parameters and the residual sums of squares at each iteration
- the estimation summary table, which displays information about the estimation method, the number of observations in the analysis, the objective function, and convergence measures
- the analysis of variance table, including sums of squares for the “Model,” “Residual,” and “Total” sources of variation (“Corrected Total” or “Uncorrected Total”), and the model F test. Note that beginning in SAS® 9, only the uncorrected total SS is reported and the respective F test is based on the uncorrected total SS if PROC NLIN determines the model does not include an intercept. If PROC NLIN determines the model does include an intercept, only the corrected total SS is reported and the respective F test is based on the corrected total SS.
- the table of parameter estimates, which contains for each parameter in the model its estimate, the approximate standard error of the estimate, and a 95% confidence interval based on the approximate standard error. The confidence level can be changed with the ALPHA= option in the PROC NLIN statement. The HOUGAARD option in the PROC NLIN statement requests that Hougaard’s skewness measure be added for each parameter. The standard errors and confidence limits are labeled approximate because they are valid asymptotically as the number of observations grows. If your model is linear in the parameters, the standard errors and confidence intervals are not approximate.

- the approximate correlation matrix of the parameter estimates. This correlation matrix is labeled approximate because it is computed from the approximate covariance matrix of the parameter estimates. If your model is linear in the parameters, the correlation matrix is not approximate.

Incompatibilities with SAS 6.11 and Earlier Versions of PROC NLIN

The NLIN procedure now uses a compiler that is different from the DATA step compiler. The compiler was changed so that analytical derivatives could be computed automatically. For the most part, the syntax accepted by the old NLIN procedure can be used in the new NLIN procedure. However, there are several differences that should be noted:

- You cannot specify a character index variable in the DO statement, and you cannot specify a character test in the IF statement. Thus `do i=1,2,3;` is supported, but `do i='ONE','TWO','THREE';` is not supported. And `if 'THIS' < 'THAT' then ...;` is supported, but `if 'THIS' THEN ...;` is not supported.
- The PUT statement, which is used mostly for program debugging in PROC NLIN, supports only some of the features of the DATA step PUT statement, and it has some new features that the DATA step PUT statement does not.
  - The PUT statement does not support line pointers, factored lists, iteration factors, overprinting, the _INFILE_ option, the ':' format modifier, or the symbol '$'.
  - The PUT statement does support expressions inside of parentheses. For example, `put (sqrt(X));` produces the square root of X.
  - The PUT statement also supports the option _PDV_ to display a formatted listing of all the variables in the program. The statement `put _pdv_;` prints a much more readable listing of the variables than `put _all_;` does.
- You cannot use the '*' subscript, but you can specify an array name in a PUT statement without subscripts. Thus, `array a ...; put a;` is acceptable, but `put a[*];` is not. The statement `put a;` displays all the elements of the array a. The `put a=;` statement displays all the elements of A with each value labeled by the name of the element variable.
- You cannot specify arguments in the ABORT statement.
- You can specify more than one target statement in the WHEN and OTHERWISE statements. That is, DO/END groups are not necessary for multiple WHEN statements, such as `select; when(exp1); stmt1; stmt2; when(exp2); stmt3; stmt4; end;`.
- You can specify only the options LOG, PRINT, and LIST in the FILE statement.
- The RETAIN statement retains only values across one pass through the data set. If you need to retain values across iterations, use the CONTROL statement to make a control variable.

The ARRAY statement in PROC NLIN is similar to, but not the same as, the ARRAY statement in the DATA step. The ARRAY statement is used to associate a name (of no more than 8 characters) with a list of variables and constants. The array name can then be used with subscripts in the program to refer to the items in the list.

The ARRAY statement supported by PROC NLIN does not support all the features of the DATA step ARRAY statement. You cannot specify implicit indexing variables; all array references must have explicit subscript expressions. You can specify simple array dimensions; lower bound specifications are not supported. A maximum of six dimensions are accepted.

On the other hand, the ARRAY statement supported by PROC NLIN does accept both variables and constants as array elements. In the following statements, b is a constant array and c is a variable array. Note that the constant array elements cannot be changed with assignment statements.

```
proc nlin data=nld;
array b[4] 1 2 3 4;     /* Constant array */
array c[4] ( 1 2 3 4 ); /* Numeric array with initial values */

b[1] = 2;               /* This is an ERROR, b is a constant array*/
c[2] = 7.5;             /* This is allowed */
```

Both dimension specification and the list of elements are optional, but at least one must be specified. When the list of elements is not specified, or fewer elements than the size of the array are listed, array variables are created by suffixing element numbers to the array name to complete the element list.

If the array is used as a pure array in the program rather than a list of symbols (the individual symbols of the array are not referenced in the code), the array is converted to a numerical array. A pure array is literally a vector of numbers that are accessed only by index. Using these types of arrays results in faster derivatives and compiled code. The assignment to c1 in the following statements forces the array to be treated as a list of symbols:

```
proc nlin data=nld;
array c[4] ( 1 2 3 4 ); /* Numeric array with initial values */

c[2] = 7.5;             /* This is C used as a pure array  */
c1 = -92.5;             /* This forces C to be a list of symbols */
```

ODS Table Names

PROC NLIN 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. These names are listed in Table 60.3. For more information about ODS, see Chapter 20, “Using the Output Delivery System.”

Table 60.3 ODS Tables Produced by PROC NLIN

ODS Table Name	Description	Statement
ANOVA	Analysis of variance	default
CodeDependency	Variable cross reference	LISTDEP
CodeList	Listing of program statements	LISTCODE
ConvergenceStatus	Convergence status	default
CorrB	Correlation of the parameters	default
EstSummary	Summary of the estimation	default
FirstDerivatives	First derivative table	LISTDER
IterHistory	Iteration output	default
MissingValues	Missing values generated by the program	default
ParameterEstimates	Parameter estimates	default
ProgList	Listing of the compiled program	LIST

Convergence Status Table

The "Convergence Status" table can be used to programmatically check the status of an estimation. This table contains the Status variable that takes on the value 0, 1, 2, or 3. If Status takes on a value less than 3, the convergence criterion was met. Specifically, the values mean the following:

Status=0	indicates that the convergence criterion was met and no warning or error messages were issued during the PROC NLIN run. Also, no notes that could indicate a problem with the model were issued.
Status=1	indicates that the convergence criterion was met and notes were written to the log that might indicate a problem with the model.
Status=2	indicates that the convergence criterion was met and one or more warning messages were produced during the PROC NLIN run.
Status=3	indicates that the convergence criterion was not met.

The following sample program demonstrates how the "Convergence Status" table can be used:

```
ods output ConvergenceStatus=ConvStatus;
proc nlin data=YourData;
   parameters a=1 b=1 c=1;
   model wgt = a + x / (b*y+c*z);
run;

data _null_;
  set ConvStatus;
  if status > 0 then put "A problem occurred";
run;
```

Examples: NLIN Procedure

Example 60.1: Segmented Model

Suppose you are interested in fitting a model that consists of two segments that connect in a smooth fashion. For example, the following model states that for values of x less than x_0 the mean of Y is a quadratic function in x, and for values of x greater than x_0 the mean of Y is constant:

$$\mathrm{E}[Y|x] = \begin{cases} \alpha + \beta x + \gamma x^2 & \text{if } x < x_0 \\ c & \text{if } x \geq x_0 \end{cases}$$

In this model equation α, β, and γ are the coefficients of the quadratic segment, and c is the plateau of the mean function. The NLIN procedure can fit such a segmented model even when the join point, x_0, is unknown.

We also want to impose conditions on the two segments of the model. First, the curve should be continuous—that is, the quadratic and the plateau section need to meet at x_0. Second, the curve should be smooth—that is, the first derivative of the two segments with respect to x need to coincide at x_0.

The continuity condition requires that

$$p = \mathrm{E}[Y|x_0] = \alpha + \beta x_0 + \gamma x_0^2$$

The smoothness condition requires that

$$\frac{\partial \mathrm{E}[Y|x_0]}{\partial x} = \beta + 2\gamma x_0 \equiv 0$$

If you solve for x_0 and substitute into the expression for c, the two conditions jointly imply that

$$\begin{aligned} x_0 &= -\beta/2\gamma \\ c &= \alpha - \beta^2/4\gamma \end{aligned}$$

Although there are apparently four unknowns, the model contains only three parameters. The continuity and smoothness restrictions together completely determine one parameter given the other three.

The following DATA step creates the SAS data set for this example:

```
data a;
   input y x @@;
   datalines;
.46 1  .47  2 .57  3 .61  4 .62  5 .68  6 .69  7
.78 8  .70  9 .74 10 .77 11 .78 12 .74 13 .80 13
.80 15 .78 16
 ;
```

The following PROC NLIN statements fit this segmented model:

```
title 'Quadratic Model with Plateau';
proc nlin data=a;
   parms alpha=.45 beta=.05 gamma=-.0025;

   x0 = -.5*beta / gamma;

   if (x < x0) then
        mean = alpha + beta*x  + gamma*x*x;
   else mean = alpha + beta*x0 + gamma*x0*x0;
   model y = mean;

   if _obs_=1 and _iter_ =.  then do;
      plateau =alpha + beta*x0 + gamma*x0*x0;
      put /  x0= plateau=  ;
   end;
   output out=b predicted=yp;
run;
```

The parameters of the model are α, β, and γ, respectively. They are represented in the PROC NLIN statements by the variables alpha, beta, and gamma, respectively. In order to model the two segments, a conditional statement is used that assigns the appropriate expression to the mean function depending on the value of x_0. A PUT statement is used to print the constrained parameters every time the program is executed for the first observation. The OUTPUT statement computes predicted values for plotting and saves them to data set b.

Note that there are other ways in which you can write the conditional expressions for this model. For example, you could formulate a condition with two model statements, as follows:

```
proc nlin data=a;
   parms alpha=.45 beta=.05 gamma=-.0025;
   x0 = -.5*beta / gamma;
   if (x < x0) then
        model y = alpha+beta*x+gamma*x*x;
   else model y = alpha+beta*x0+gamma*x0*x0;
run;
```

Or you could use a single expression with a conditional evaluation, as in the following statements:

```
proc nlin data=a;
   parms alpha=.45 beta=.05 gamma=-.0025;
   x0 = -.5*beta / gamma;
   model y = (x <x0)*(alpha+beta*x +gamma*x*x) +
             (x>=x0)*(alpha+beta*x0+gamma*x0*x0);
run;
```

The results from fitting this model with PROC NLIN are shown in Output 60.1.1–Output 60.1.3. The iterative optimization converges after six iterations (Output 60.1.1). Output 60.1.1 indicates that the join point is 12.747 and the plateau value is 0.777.

Output 60.1.1 Nonlinear Least-Squares Iterative Phase

```
                           Quadratic Model with Plateau

                                The NLIN Procedure
                               Dependent Variable y
                               Method: Gauss-Newton

                                 Iterative Phase
                                                                 Sum of
               Iter        alpha          beta         gamma      Squares

                  0       0.4500        0.0500      -0.00250       0.0562
                  1       0.3881        0.0616      -0.00234       0.0118
                  2       0.3930        0.0601      -0.00234       0.0101
                  3       0.3922        0.0604      -0.00237       0.0101
                  4       0.3921        0.0605      -0.00237       0.0101
                  5       0.3921        0.0605      -0.00237       0.0101
                  6       0.3921        0.0605      -0.00237       0.0101


         NOTE: Convergence criterion met.
```

Output 60.1.2 Results from Put Statement

```
x0=12.747669162 plateau=0.7774974276
```

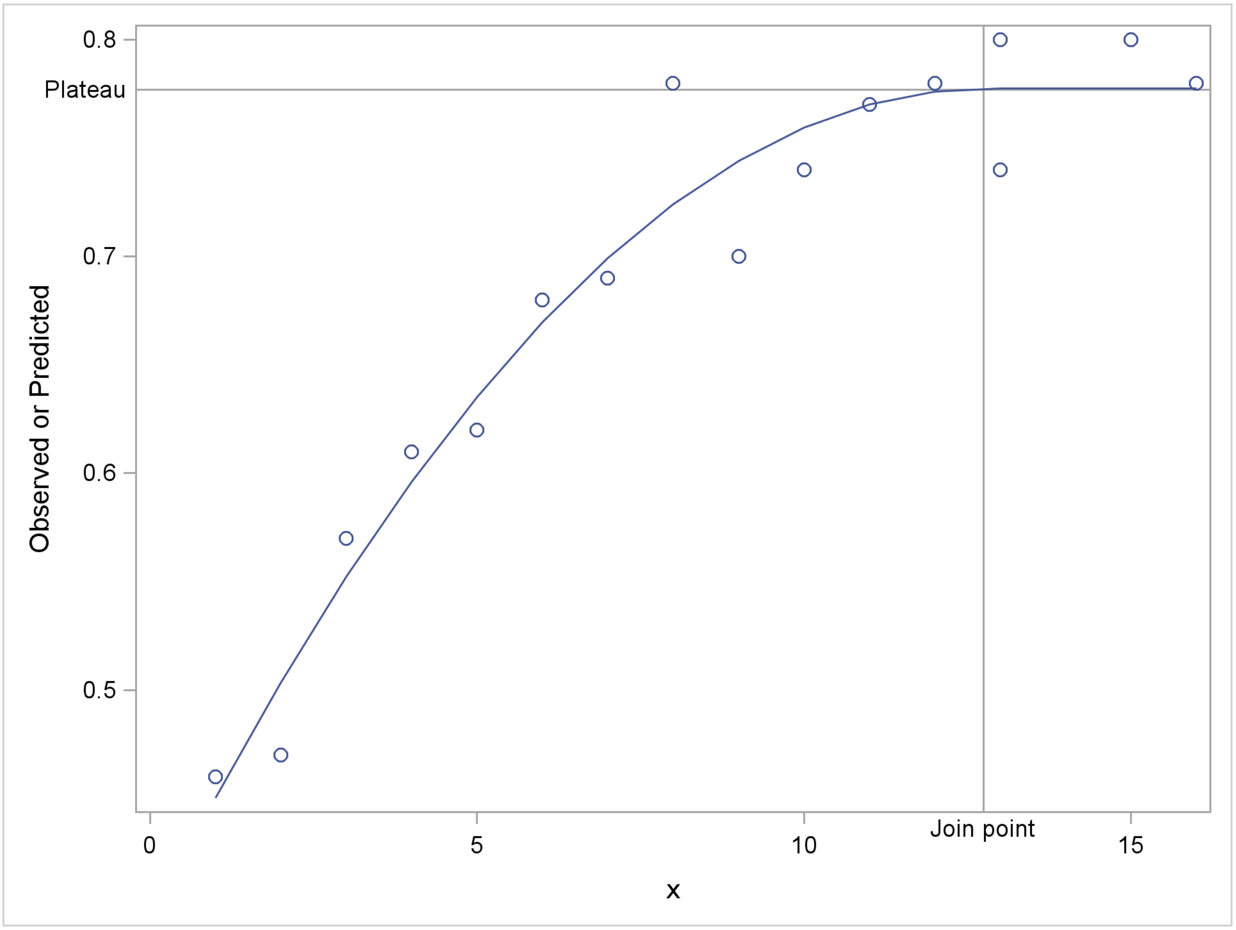
Output 60.1.3 Least-Squares Analysis for the Quadratic Model

```
                                          Sum of          Mean                   Approx
    Source                        DF     Squares        Square    F Value    Pr > F

    Model                          2      0.1769        0.0884     114.22    <.0001
    Error                         13      0.0101      0.000774
    Corrected Total               15      0.1869

                                             Approx      Approximate 95% Confidence
           Parameter      Estimate       Std Error                   Limits

           alpha            0.3921          0.0267        0.3345        0.4497
           beta             0.0605         0.00842        0.0423        0.0787
           gamma          -0.00237        0.000551      -0.00356      -0.00118
```

The following statements produce a graph of the observed and predicted values with reference lines for the join point and plateau estimates (Output 60.1.4):

```
proc sgplot data=b noautolegend;
   yaxis label='Observed or Predicted';
   refline 0.777  / axis=y label="Plateau"    labelpos=min;
   refline 12.747 / axis=x label="Join point" labelpos=min;
   scatter y=y  x=x;
   series  y=yp x=x;
run;
```

Output 60.1.4 Observed and Predicted Values for the Quadratic Model



If you want to estimate the join point directly, you can use the relationship between the parameters to change the parameterization of the model in such a way that the mean function depends directly on x_0. Using the smoothness condition that relates x_0 to γ,

$$x_0 = -\beta/2\gamma$$

you can express γ as a function of β and x_0:

$$\gamma = -\beta/(2x_0)$$

Substituting for γ in the model equation

$$\mathrm{E}[Y|x] = \begin{cases} \alpha + \beta x + \gamma x^2 & \text{if } x < x_0 \\ \alpha - \beta^2/(4\gamma) & \text{if } x \geq x_0 \end{cases}$$

yields the reparameterized model

$$\mathrm{E}[Y|x] = \begin{cases} \alpha + \beta x(1 - x/(2x_0)) & \text{if } x < x_0 \\ \alpha + \beta x_0/2 & \text{if } x \geq x_0 \end{cases}$$

This model is fit with the following PROC NLIN statements:

```
proc nlin data=a;
   parms alpha=.45 beta=.05 x0=10;
   if (x<x0) then
        mean = alpha + beta*x *(1-x/(2*x0));
   else mean = alpha + beta*x0/2;
   model y = mean;
run;
```

Output 60.1.5 Results from Reparameterized Model

```
                              The NLIN Procedure
                             Dependent Variable y
                             Method: Gauss-Newton

        NOTE: Convergence criterion met.

                                         Sum of        Mean                 Approx
    Source                     DF       Squares      Square    F Value    Pr > F

    Model                       2        0.1769      0.0884     114.22    <.0001
    Error                      13        0.0101    0.000774
    Corrected Total            15        0.1869

                                            Approx      Approximate 95% Confidence
          Parameter      Estimate       Std Error                  Limits

          alpha            0.3921          0.0267       0.3345       0.4497
          beta             0.0605         0.00842       0.0423       0.0787
          x0              12.7477          1.2781       9.9864      15.5089
```

The analysis of variance table in the reparameterized model is the same as in the earlier analysis (compare Output 60.1.5 and Output 60.1.3). Changing the parameterization of a model does not affect the fit. The “Parameter Estimates” table now shows x0 as a parameter in the model. The estimate agrees with the earlier result that uses the PUT statement (Output 60.1.2). Since x_0 is now a model parameter, the NLIN procedure also reports its asymptotic standard error and its approximate 95% confidence interval.

Example 60.2: Iteratively Reweighted Least Squares

With the NLIN procedure you can perform weighted nonlinear least-squares regression in situations where the weights are functions of the parameters. To minimize a weighted sum of squares, you assign an expression to the _WEIGHT_ variable in your PROC NLIN statements. When the _WEIGHT_ variable depends on the model parameters, the estimation technique is known as iteratively reweighted least squares (IRLS). In this situation you should employ the NOHALVE option in the PROC NLIN statement. Because the weights change from iteration to iteration, it is not reasonable to expect the *weighted* residual sum of squares to decrease between iterations. The NOHALVE option removes that restriction.

Examples where IRLS estimation is used include robust regression via M-estimation (Huber 1964, 1973), generalized linear models (McCullagh and Nelder 1989), and semivariogram fitting in spatial statistics (Schabenberger and Pierce 2002, Sect. 9.2). There are dedicated SAS/STAT procedures for robust regression (the ROBUSTREG procedure) and generalized linear models (the GENMOD and GLIMMIX procedures). Examples of weighted least-squares fitting of a semivariogram function can be found in Chapter 96, “The VARIOGRAM Procedure.”

In this example we show an application of PROC NLIN for M-estimation only to illustrate the connection between robust regression and weighted least squares. The ROBUSTREG procedure is the appropriate tool to fit these models with SAS/STAT software.

M-estimation was introduced by Huber (1964, 1973) to estimate location parameters robustly. Beaton and Tukey (1974) applied the idea of M-estimation in regression models and introduced the biweight (or bisquare) weight function. See Holland and Welsch (1977) for this and other robust methods. Consider a linear regression model of the form

$$\mathrm{E}[Y_i|x] = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

In weighted least-squares estimation you seek the parameters $\widehat{\boldsymbol{\beta}}$ that minimize

$$\sum_{i=1}^{n} w_i \left(y_i - \mathbf{x}_i'\boldsymbol{\beta}\right)^2 = \sum_{i=1}^{n} w_i e_i^2$$

where w_i is the weight associated with the ith observation. The *normal* equations of this minimization problem can be written as

$$\sum_{i=1}^{n} w_i e_i \mathbf{x}_i = \mathbf{0}$$

In M-estimation the corresponding equations take on the form

$$\sum_{i=1}^{n} \psi(e_i)\mathbf{x}_i = \mathbf{0}$$

where $\psi(\cdot)$ is a weighing function. The Beaton-Tukey biweight, for example, can be written as

$$\psi(e_i) = \begin{cases} e_i \left(1 - \left(\frac{e_i}{\sigma k}\right)^2\right)^2 & |e_i/\sigma| \le k \\ 0 & |e_i/\sigma| > k \end{cases}$$

Substitution into the estimating equation for M-estimation yields weighted least-squares equations

$$\sum_{i=1}^{n} \psi(e_i)\mathbf{x}_i = \sum_{i=1}^{n} w_i e_i \mathbf{x}_i = \mathbf{0}$$

$$w_i = \begin{cases} \left(1 - \left(\frac{e_i}{\sigma k}\right)^2\right)^2 & |e_i/\sigma| \le k \\ 0 & |e_i/\sigma| > k \end{cases}$$

The biweight function involves two constants, σ and k. The scale σ can be fixed or estimated from the fit in the previous iteration. If σ is estimated, a robust estimator of scale is typically used. In this example σ is fixed at 2. A common value for the constant k is $k = 4.685$.

The following DATA step creates a SAS data set of the population of the United States (in millions), recorded at 10-year intervals starting in 1790 and ending in 1990. The aim is to fit a quadratic linear model to the population over time.

```
title 'U.S. Population Growth';
data uspop;
   input pop :6.3 @@;
   retain year 1780;
   year   = year+10;
   yearsq = year*year;
   datalines;
3929 5308 7239 9638 12866 17069 23191 31443 39818 50155
62947 75994 91972 105710 122775 131669 151325 179323 203211
226542 248710
;
```

The PROC NLIN code that follows fits this linear model by M-estimation and IRLS. The weight function is set to a zero or nonzero value depending on the value of the scaled residual. The NOHALVE option removes the requirement that the (weighted) residual sum of squares must decrease between iterations.

```
title 'Beaton/Tukey Biweight Robust Regression using IRLS';
proc nlin data=uspop nohalve;
   parms b0=20450.43 b1=-22.7806 b2=.0063456;
   model pop=b0+b1*year+b2*year*year;
   resid = pop-model.pop;
   sigma = 2;
   k     = 4.685;
   if abs(resid/sigma)<=k then _weight_=(1-(resid / (sigma*k))**2)**2;
   else _weight_=0;
   output out=c r=rbi;
run;
```

Parameter estimates from this fit are shown in Output 60.2.1, and the computed weights at the final iteration are displayed in Output 60.2.2. The observations for 1940 and 1950 are highly discounted because of their large residuals.

Output 60.2.1 Nonlinear Least-Squares Analysis

```
                Beaton/Tukey Biweight Robust Regression using IRLS

                              The NLIN Procedure

                                     Sum of         Mean               Approx
Source                      DF      Squares       Square    F Value    Pr > F

Model                        2       113564      56782.0    49454.5    <.0001
Error                       18      20.6670       1.1482
Corrected Total             20       113585

                                        Approx      Approximate 95% Confidence
       Parameter       Estimate      Std Error                 Limits

       b0               20828.7          259.4       20283.8      21373.6
       b1              -23.2004         0.2746      -23.7773     -22.6235
       b2               0.00646       0.000073       0.00631      0.00661
```

Output 60.2.2 Listing of Computed Weights from PROC NLIN

```
 Obs       pop      year     yearsq        rbi        sigma       k       _weight_

   1      3.929     1790    3204100    -0.93711        2       4.685     0.98010
   2      5.308     1800    3240000     0.46091        2       4.685     0.99517
   3      7.239     1810    3276100     1.11853        2       4.685     0.97170
   4      9.638     1820    3312400     0.95176        2       4.685     0.97947
   5     12.866     1830    3348900     0.32159        2       4.685     0.99765
   6     17.069     1840    3385600    -0.62597        2       4.685     0.99109
   7     23.191     1850    3422500    -0.94692        2       4.685     0.97968
   8     31.443     1860    3459600    -0.43027        2       4.685     0.99579
   9     39.818     1870    3496900    -1.08302        2       4.685     0.97346
  10     50.155     1880    3534400    -1.06615        2       4.685     0.97427
  11     62.947     1890    3572100     0.11332        2       4.685     0.99971
  12     75.994     1900    3610000     0.25539        2       4.685     0.99851
  13     91.972     1910    3648100     2.03607        2       4.685     0.90779
  14    105.710     1920    3686400     0.28436        2       4.685     0.99816
  15    122.775     1930    3724900     0.56725        2       4.685     0.99268
  16    131.669     1940    3763600    -8.61325        2       4.685     0.02403
  17    151.325     1950    3802500    -8.32415        2       4.685     0.04443
  18    179.323     1960    3841600    -0.98543        2       4.685     0.97800
  19    203.211     1970    3880900     0.95088        2       4.685     0.97951
  20    226.542     1980    3920400     1.03780        2       4.685     0.97562
  21    248.710     1990    3960100    -1.33067        2       4.685     0.96007
```

You can obtain this analysis more conveniently with PROC ROBUSTREG. The procedure re-estimates the scale parameter robustly between iterations. To obtain an analysis with a fixed scale parameter as in this example, use the following PROC ROBUSTREG statements:

```
proc robustreg data=uspop method=m(scale=2);
   model pop = year year*year;
   output out=weights weight=w;
run;
```

```
proc print data=weights;
run;
```

Note that the computation of standard errors in the ROBUSTREG procedure is different from the calculations in the NLIN procedure.

Example 60.3: Probit Model with Likelihood Function

The data in this example, taken from Lee (1974), consist of patient characteristics and a variable indicating whether cancer remission occurred. This example demonstrates how to use PROC NLIN with a likelihood function. In this case, twice the negative of the log-likelihood function is to be minimized. This is the objective function for the analysis:

$$-2\log L = -2\sum_{i=1}^{n} \log\{\pi_i(y_i, \mathbf{x}_i)\}$$

In this expression, π_i denotes the success probability of the n Bernoulli trials, and $\log L$ is the log likelihood of n independent binary (Bernoulli) observations. The probability π_i depends on the observations through the linear predictor η_i,

$$\pi_i(y_i, \mathbf{x}_i) = \begin{cases} 1 - \Phi(\eta_i) & y_i = 1 \\ \Phi(\eta_i) & y_i = 0 \end{cases}$$

The linear predictor takes the form of a regression equation that is linear in the parameters,

$$\eta_i = \beta_0 + \beta_1 z_{1i} + \beta_2 z_{2i} + \cdots \beta_k z_{ki} = \mathbf{z}_i \boldsymbol{\beta}$$

Despite this linearity of η in the z variables, the probit model is nonlinear, because the linear predictor appears inside the nonlinear probit function.

In order to use the NLIN procedure to minimize the function, the estimation problem must be cast in terms of a nonlinear least-squares problem with objective function

$$\sum_{i=1}^{n}(y_i - f(\boldsymbol{\beta}, \mathbf{z}_i'))^2$$

This can be accomplished by setting $y_i = 0$ and $f(\boldsymbol{\beta}, \mathbf{z}_i') = \sqrt{(-2\log\{\pi_i\})}$. Because $0 \le \pi \le 1$, the function $-2\log\{\pi_i\}$ is strictly positive and the square root can be taken safely.

The following DATA step creates the data for the probit analysis. The variable like is created in the DATA step, and it contains the value 0 throughout. This variable serves as the "dummy" response variable in the PROC NLIN step. The variable remiss indicates whether cancer remission occurred. It is the binary outcome variable of interest and is used to determine the relevant probability for observation i as the success or failure probability of a Bernoulli experiment.

```
data remiss;
   input remiss cell smear infil li blast temp;
   label remiss = 'complete remission';
   like = 0;
   label like = 'dummy variable for nlin';
   datalines;
 1 0.8  .83 .66 1.9 1.10   .996
 1 0.9  .36 .32 1.4 0.74   .992
 0 0.8  .88 .70 0.8 0.176  .982
 0 1    .87 .87 0.7 1.053  .986
 1 0.9  .75 .68 1.3 0.519  .980
 0 1    .65 .65 0.6 0.519  .982
 1 0.95 .97 .92 1   1.23   .992
 0 0.95 .87 .83 1.9 1.354 1.020
 0 1    .45 .45 0.8 0.322  .999
 0 0.95 .36 .34 0.5 0     1.038
 0 0.85 .39 .33 0.7 0.279  .988
 0 0.7  .76 .53 1.2 0.146  .982
 0 0.8  .46 .37 0.4 0.38  1.006
 0 0.2  .39 .08 0.8 0.114  .990
 0 1    .90 .90 1.1 1.037  .990
 1 1    .84 .84 1.9 2.064 1.020
 0 0.65 .42 .27 0.5 0.114 1.014
 0 1    .75 .75 1   1.322 1.004
 0 0.5  .44 .22 0.6 0.114  .990
 1 1    .63 .63 1.1 1.072  .986
 0 1    .33 .33 0.4 0.176 1.010
 0 0.9  .93 .84 0.6 1.591 1.020
 1 1    .58 .58 1   0.531 1.002
 0 0.95 .32 .30 1.6 0.886  .988
 1 1    .60 .60 1.7 0.964  .990
 1 1    .69 .69 0.9 0.398  .986
 0 1    .73 .73 0.7 0.398  .986
 ;
```

The following NLIN statements fit the probit model:

```
proc nlin data=remiss method=newton sigsq=1;
   parms int=-10 a = -2 b = -1 c=6;

   linp = int + a*cell + b*li + c*temp;
   p    = probnorm(linp);

   if (remiss = 1) then pi = 1-p;
                   else pi = p;

   model.like = sqrt(- 2 * log(pi));
   output out=p p=predict;
run;
```

The assignment to the variable linp creates the linear predictor of the generalized linear model,

$$\eta = \beta_0 + \beta_1 \text{cell}_i + \beta_2 \text{li}_i + \beta_3 * \text{temp}_i$$

In this example, the variables cell, li, and temp are used as regressors.

By default, the NLIN procedure computes the covariance matrix of the parameter estimates based on the nonlinear least-squares assumption. That is, the procedure computes the estimate of the residual variance as the mean squared error and uses that to multiply the inverse crossproduct matrix or the inverse Hessian matrix. (See the section "Covariance Matrix of Parameter Estimates" on page 4931 for details.) In the probit model, there is no residual variance. In addition, standard errors in maximum likelihood estimation are based on the inverse Hessian matrix. The METHOD=NEWTON option in the PROC NLIN statement is used to employ the Hessian matrix in computing the covariance matrix of the parameter estimates. The SIGSQ=1 option replaces the residual variance estimate that PROC NLIN would use by default as a multiplier of the inverse Hessian with the value 1.0.

Output 60.3.1 shows the results of this analysis. The analysis of variance table shows an apparently strange result. The total sum of squares is zero, and the model sum of squares is negative. Recall that the values of the response variable were set to zero and the mean function was constructed as $-2\log\{\pi_i\}$ in order for the NLIN procedure to minimize a log-likelihood function in terms of a nonlinear least-squares problem. The value 21.9002 shown as the "Error" sum of squares is the value of the function $-2\log L$.

Output 60.3.1 Nonlinear Least-Squares Analysis from PROC NLIN

```
                          The NLIN Procedure

             NOTE: An intercept was not specified for this model.

                                  Sum of        Mean               Approx
 Source                   DF     Squares      Square    F Value    Pr > F

 Model                     4    -21.9002     -5.4750      -5.75         .
 Error                    23     21.9002      0.9522
 Uncorrected Total        27           0

                                         Approx      Approximate 95% Confidence
        Parameter      Estimate       Std Error                 Limits

        int            -36.7548         32.3607       -103.7     30.1885
        a               -5.6298          4.6376     -15.2235      3.9639
        b               -2.2513          0.9790      -4.2764     -0.2262
        c               45.1815         34.9095     -27.0343       117.4
```

The problem can be more simply solved using dedicated procedures for generalized linear models:

```
proc glimmix data=remiss;
   model remiss = cell li temp / dist=binary link=probit s;
run;

proc genmod data=remiss;
   model remiss = cell li temp / dist=bin link=probit;
run;
```

```
proc logistic data=remiss;
   model remiss = cell li temp / link=probit technique=newton;
run;
```

Example 60.4: Affecting Curvature through Parameterization

The work of Ratkowksy (1983, 1990) has brought into focus the importance of close-to-linear behavior of parameters in nonlinear regression models. The curvature in a nonlinear model consists of two components, the intrinsic curvature and the parameter-effects curvature. Intrinsic curvature expresses the degree to which the nonlinear model bends as values of the *parameters* change. This is not the same as the curviness of the model as a function of the covariates (the x variables). Intrinsic curvature is a function of the type of model you are fitting and the data. This curvature component cannot be affected by reparameterization of the model. According to Ratkowsky (1983), the intrinsic curvature component is typically smaller than the parameter-effects curvature, which can be affected by altering the parameterization of the model.

In models with low curvature, the nonlinear least-squares parameter estimators behave similarly to least-squares estimators in linear regression models, which have a number of desirable properties. If the model is correct, they are best linear unbiased estimators and are normally distributed if the model errors are normal (otherwise they are asymptotically normal). As you lower the curvature of a nonlinear model, you can expect that the parameter estimators approach the behavior of the linear regression model estimators: they behave "close to linear."

This example uses a simple data set and a commonly applied model for dose-response relationships to examine how the parameter-effects curvature can be reduced. The statistic by which an estimator's behavior is judged is Hougaard's measure of skewness (Hougaard 1982, 1985).

The log-logistic model

$$\mathrm{E}[Y|x] = \delta + \frac{\alpha - \delta}{1 + \gamma \exp\{\beta \ln(x)\}}$$

is a popular model to express the response Y as a function of dose x. The response is bounded between the asymptotes α and δ. The term in the denominator governs the transition between the asymptotes and depends on two parameters, γ and β. The log-logistic model can be viewed as a member of a broader class of dose-response functions, those relying on *switch-on* or *switch-off* mechanisms (see, for example, Schabenberger and Pierce 2002, Sect. 5.8.6). A switch function is usually a monotonic function $S(x, \boldsymbol{\theta})$ that takes values between 0 and 1. A switch-on function increases in x; a switch-off function decreases in x. In the log-logistic case, the function

$$S(x, [\beta, \gamma]) = \frac{1}{1 + \gamma \exp\{\beta \ln(x)\}}$$

is a switch-off function for $\beta > 0$ and a switch-on function for $\beta < 0$. You can write general dose-response functions with asymptotes simply as

$$\mathrm{E}[Y|x] = \mu_{\min} + (\mu_{\max} - \mu_{\min}) S(x, \boldsymbol{\theta})$$

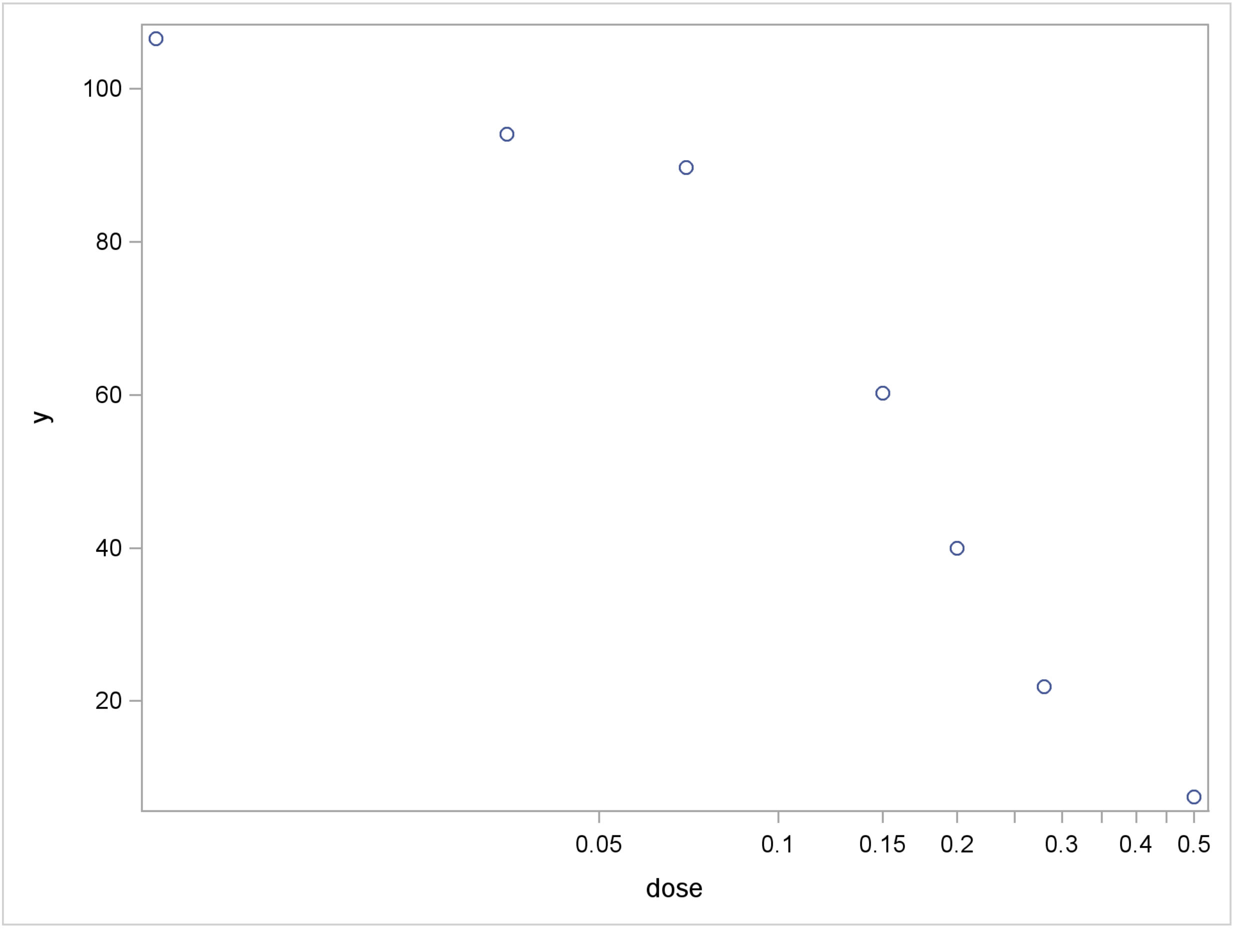
The following DATA step creates a small data set from a dose-response experiment with response y:

```
data logistic;
   input dose y;
   logdose = log(dose);
   datalines;
0.009  106.56
0.035   94.12
0.07    89.76
0.15    60.21
0.20    39.95
0.28    21.88
0.50     7.46
;
```

A graph of these data is produced with the following statements:

```
proc sgplot data=logistic;
   scatter y=y x=dose;
   xaxis type=log logstyle=linear;
run;
```

Output 60.4.1 Observed Data in Dose-Response Experiment



When dose is expressed on the log scale, the sigmoidal shape of the dose-response relationship is clearly visible (Output 60.4.1). The log-logistic switching model in the preceding parameterization is fit with the following statements in the NLIN procedure:

```
proc nlin data=logistic hougaard;
   parameters alpha=100 beta=3 gamma=300;
   delta = 0;
   Switch = 1/(1+gamma*exp(beta*log(dose)));
   model y = delta + (alpha - delta)*Switch;
run;
```

The lower asymptote δ was assumed to be 0 in this case. Since δ is not listed in the PARAMETERS statement and is assigned a value in the program, it is assumed to be constant. Note that the term Switch is the switch-off function in the log-logistic model. The HOUGAARD option in the PROC NLIN statement requests that Hougaard's skewness measure is added to the table of parameter estimates.

The NLIN procedure converges after 10 iteration and achieves a residual mean squared error of 15.1869 (Output 60.4.2). This value is not that important by itself, but it is worth noting since this model fit is compared to the fit with other parameterizations later on.

Output 60.4.2 Iteration History and Analysis of Variance

```
                         The NLIN Procedure
                        Dependent Variable y
                        Method: Gauss-Newton

                          Iterative Phase
                                                          Sum of
             Iter       alpha        beta       gamma    Squares

                0       100.0      3.0000       300.0      386.4
                1       100.4      2.8011       162.8      129.1
                2       100.8      2.6184       101.4    69.2710
                3       101.3      2.4266     69.7579    68.2167
                4       101.7      2.3790     69.0358    60.8223
                5       101.8      2.3621     67.3709    60.7516
                6       101.8      2.3582     67.0044    60.7477
                7       101.8      2.3573     66.9150    60.7475
                8       101.8      2.3571     66.8948    60.7475
                9       101.8      2.3570     66.8902    60.7475
               10       101.8      2.3570     66.8892    60.7475

        NOTE: Convergence criterion met.


             NOTE: An intercept was not specified for this model.

                                      Sum of        Mean                  Approx
    Source                    DF     Squares      Square    F Value    Pr > F

    Model                      3     33965.4     11321.8     745.50    <.0001
    Error                      4     60.7475     15.1869
    Uncorrected Total          7     34026.1
```

The table of parameter estimates displays the estimates of the three model parameters, their approximate standard errors, 95% confidence limits, and Hougaard's skewness measure (Output 60.4.3). Parameters for which this measure is less than 0.1 in absolute value exhibit very close-to-linear behavior, and values less than 0.25 in absolute value indicate reasonably close-to-linear behavior (Ratkowsky 1990). According to these rules, the estimators $\widehat{\beta}$ and $\widehat{\gamma}$ suffer from substantial curvature. The estimator $\widehat{\gamma}$ is especially "far from linear." Inferences involving $\widehat{\gamma}$ that rely on the reported standard errors and/or confidence limits for this parameter might be questionable.

Output 60.4.3 Parameter Estimates and Hougaard's Skewness

```
                                  Approx       Approximate 95%
   Parameter       Estimate    Std Error      Confidence Limits       Skewness

   alpha              101.8       3.0034     93.4751       110.2        0.1415
   beta              2.3570       0.2928      1.5440      3.1699        0.4987
   gamma            66.8892      31.6146    -20.8870       154.7        1.9200
```

One method of reducing the parameter-effects curvature is to replace a parameter with its expected-value parameterization. Schabenberger et al. (1999) and Schabenberger and Pierce (2002, Sect. 5.7.2) refer to this method as *reparameterization through defining relationships*. A defining relationship is obtained by equating the mean response at a chosen value of x (say, x^*) to the model:

$$\mathrm{E}[Y|x^*] = \delta + \frac{\alpha - \delta}{1 + \gamma \exp\{\beta \ln(x^*)\}}$$

This equation is then solved for a parameter that is subsequently replaced in the original equation. This method is particularly useful if x^* has an interesting interpretation. For example, let λ_K denote the value that reduces the response by $K \times 100\%$,

$$\mathrm{E}[Y|\lambda_K] = \delta + \left(\frac{100 - K}{100}\right)(\alpha - \delta)$$

Because γ exhibits large skewness, it is the target in the first round of reparameterization. Setting the expression for the conditional mean at λ_K equal to the mean function when $x = \lambda_K$ yields the following expression:

$$\delta + \left(\frac{100 - K}{100}\right)(\alpha - \delta) = \delta + \frac{\alpha - \delta}{1 + \gamma \exp\{\beta \ln(\lambda_K)\}}$$

This expression is solved for γ, and the result is substituted back into the model equation. This leads to a log-logistic model in which γ is replaced by the parameter λ_K, the dose at which the response was reduced by $K \times 100\%$. The new model equation is

$$\mathrm{E}[Y|x] = \delta + \frac{\alpha - \delta}{1 + K/(100 - K)\exp\{\beta \ln(x/\lambda_K)\}}$$

A particularly interesting choice is $K = 50$, since λ_{50} is the dose at which the response is halved. In studies of mortality, this concentration is also known as the LD50. For the special case of λ_{50} the model equation becomes

$$\mathrm{E}[Y|x] = \delta + \frac{\alpha - \delta}{1 + \exp\{\beta \ln(x/\lambda_{50})\}}$$

You can fit the model in the LD50 parameterization with the following statements:

```
proc nlin data=logistic hougaard;
   parameters alpha=100 beta=3 LD50=0.15;
   delta = 0;
   Switch = 1/(1+exp(beta*log(dose/LD50)));
   model y = delta + (alpha - delta)*Switch;
   output out=nlinout pred=p lcl=lcl ucl=ucl;
run;
```

Partial results from this NLIN run are shown in Output 60.4.4. The analysis of variance tables in Output 60.4.2 and Output 60.4.4 are identical. Changing the parameterization of a model does not affect the model fit. It does, however, affect the interpretation of the parameters and the statistical properties (close-to-linear behavior) of the parameter estimators. The skewness measure of the parameter LD50 is considerably reduced compared to the skewness of the parameter γ in the previous parameterization. Also, γ has been replaced by a parameter with a useful interpretation, the dose that yields a 50% reduction in mean response. Also notice that the skewness measures of α and β are not affected by the $\gamma \rightarrow$ LD50 reparameterization.

Output 60.4.4 ANOVA Table and Parameter Estimates in LD50 Parameterization

```
                              The NLIN Procedure
                             Dependent Variable y
                            Method: Gauss-Newton

         NOTE: Convergence criterion met.

               NOTE: An intercept was not specified for this model.

                                    Sum of        Mean               Approx
   Source                    DF    Squares      Square    F Value    Pr > F

   Model                      3    33965.4     11321.8     745.50    <.0001
   Error                      4    60.7475     15.1869
   Uncorrected Total          7    34026.1

                                 Approx       Approximate 95%
    Parameter      Estimate    Std Error     Confidence Limits      Skewness

    alpha             101.8       3.0034     93.4752       110.2      0.1415
    beta             2.3570       0.2928      1.5440      3.1699      0.4987
    LD50             0.1681      0.00915      0.1427      0.1935     -0.0605
```

To reduce the parameter-effects curvature of the β parameter, you can use the technique of defining relationships again. This can be done generically, by solving

$$\mu^* = \delta + \frac{\alpha - \delta}{1 + \exp\{\beta \ln(x/\lambda)\}}$$

for β, treating μ^* as the new parameter (in lieu of β), and choosing a value for x^* that leads to low skewness. This results in the expected-value parameterization of β. Solving for β yields

$$\beta = \frac{\log\left(\frac{\alpha-\mu^*}{\mu^*-\delta}\right)}{\log\left(x^*/\lambda\right)}$$

The interpretation of the parameter μ^* that replaces β in the model equation is simple: it is the mean dose response when the dose is x^*. Fixing $x^* = 0.3$, the following PROC NLIN statements fit this model:

```
proc nlin data=logistic hougaard;
   parameters alpha=100 mustar=20 LD50=0.15;
   delta   = 0;
   xstar   = 0.3;
   beta    = log((alpha - mustar)/(mustar - delta)) / log(xstar/LD50);
   Switch  = 1/(1+exp(beta*log(dose/LD50)));
   model y = delta + (alpha - delta)*Switch;
   output out=nlinout pred=p lcl=lcl ucl=ucl;
run;
```

Note that the switch-off function continues to be written in terms of β and the LD50. The only difference from the previous model is that β is now expressed as a function of the parameter μ^*. Using expected-value parameterizations is a simple mechanism to lower the curvature in a model and to arrive at starting values. The starting value for μ^* can be gleaned from Output 60.4.1 at $x = 0.3$.

Output 60.4.5 shows selected results from this NLIN run. The ANOVA table is again unaffected by the change in parameterization. The curvature for μ^* is greatly reduced in comparison to the curvature for the β parameter in the previous model (compare Output 60.4.5 and Output 60.4.4).

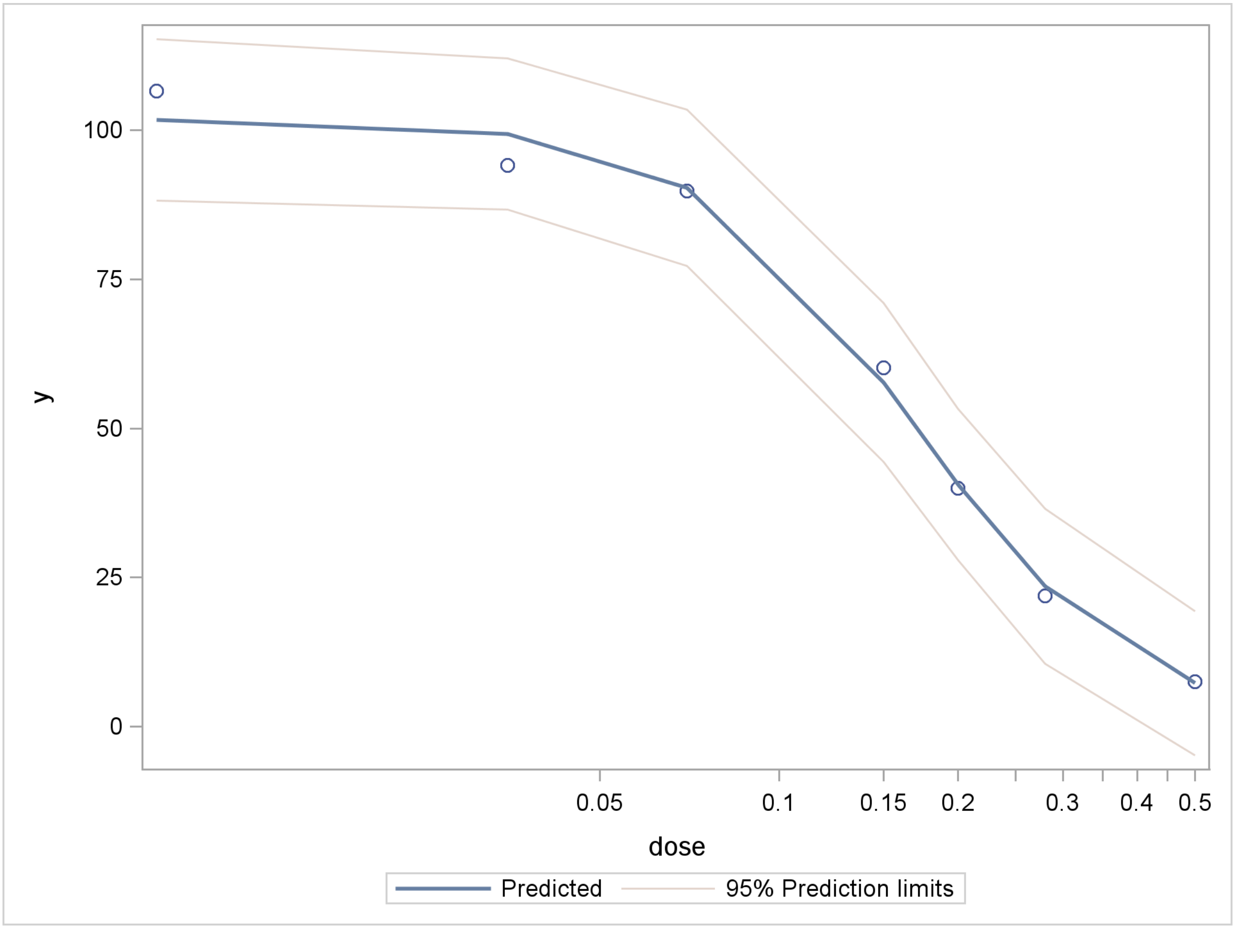
Output 60.4.5 ANOVA Table and Parameter Estimates in Expected-Value Parameterization

```
                              The NLIN Procedure
                             Dependent Variable y
                            Method: Gauss-Newton

         NOTE: Convergence criterion met.

               NOTE: An intercept was not specified for this model.

                                         Sum of        Mean               Approx
    Source                    DF        Squares      Square    F Value    Pr > F

    Model                      3        33965.4     11321.8     745.50    <.0001
    Error                      4        60.7475     15.1869
    Uncorrected Total          7        34026.1

                                      Approx       Approximate 95%
     Parameter      Estimate       Std Error      Confidence Limits       Skewness

     alpha             101.8          3.0034     93.4752        110.2       0.1415
     mustar          20.7073          2.6430     13.3693      28.0454      -0.0572
     LD50             0.1681         0.00915      0.1427       0.1935      -0.0605
```

The following statements produce a graph of the observed and fitted values along with the 95% prediction limits (Output 60.4.6). The graph is based on the data set created with the OUTPUT statement in the most recent PROC NLIN run.

```
proc sgplot data=nlinout;
   scatter y=y   x=dose;
   series  y=p   x=dose / name="fit"
                          lineattrs=GraphFit
                          LegendLabel="Predicted";
   series  y=lcl x=dose / lineattrs=GraphConfidence
                          name="lcl"
                          legendlabel="95% Prediction limits";
   series  y=ucl x=dose / lineattrs=GraphConfidence;
   keylegend "fit" "lcl";
   xaxis type=log logstyle=linear;
run;
```

Output 60.4.6 Fit Plot for Log-Logistic Model with 95% Prediction Limits



Example 60.5: Comparing Nonlinear Trends among Groups

When you model nonlinear trends in the presence of group (classification) variables, two questions often arise: whether the trends should be varied by group, and how to decide which parameters should be varied across groups. A large battery of tools is available on linear statistical models to test hypotheses involving the model parameters, especially to test linear hypotheses. To test similar hypotheses in nonlinear models, you can draw on analogous tools. Especially important in this regard are comparisons of nested models by contrasting their residual sums of squares.

In this example, a two-group model from a pharmacokinetic application is fit to data that are in part based on the theophylline data from Pinheiro and Bates (1995) and the first example in the documentation for the NLMIXED procedure. In a pharmacokinetic application you study how a drug is dispersed through a living organism. The following data represent concentrations of the drug theophylline over a 25-hour period following oral administration. The data are derived by collapsing and averaging the subject-specific data from Pinheiro and Bates (1995) in a particular, yet unimportant, way. The purpose of arranging the data in this way is purely to demonstrate the methodology.

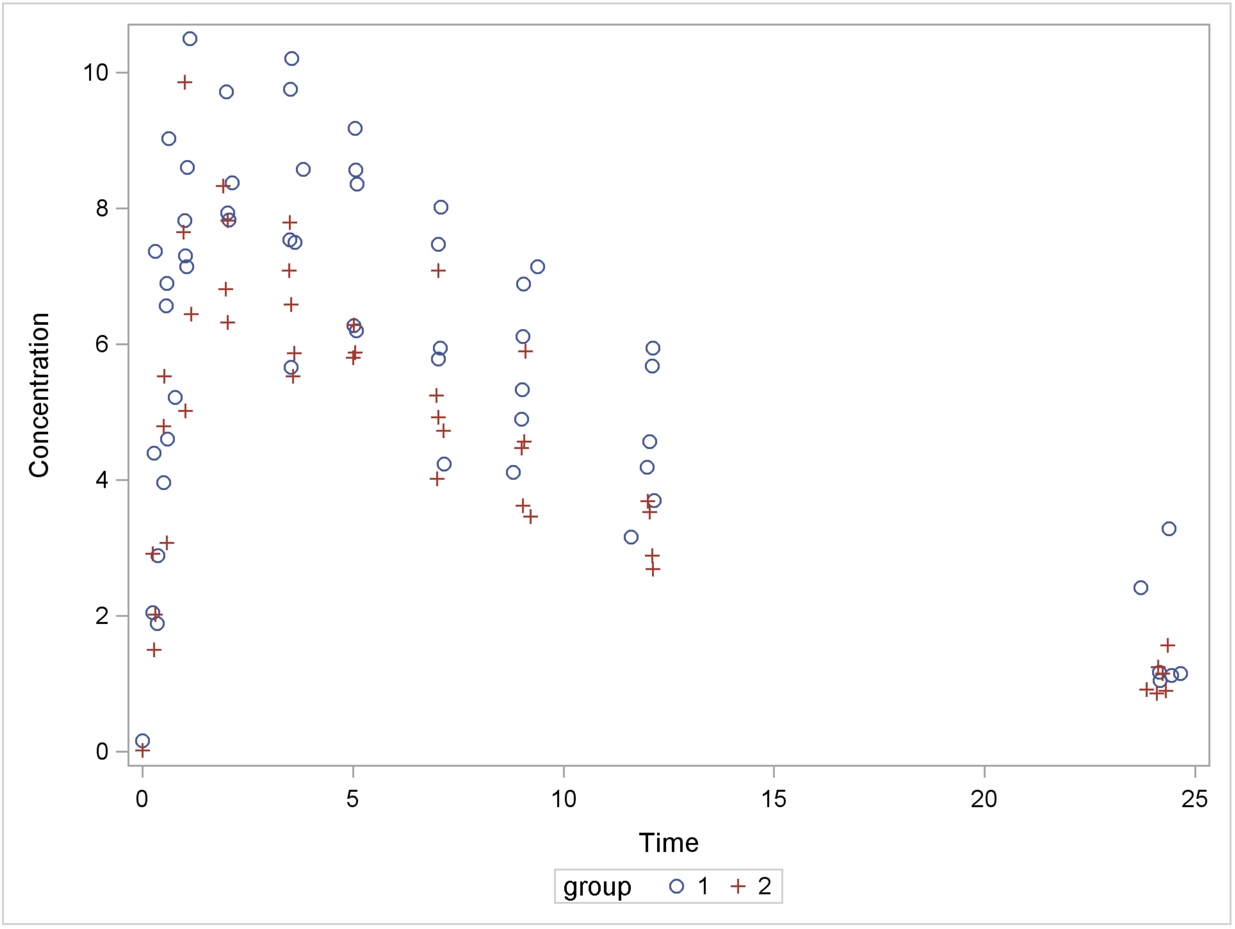
```
data theop;
  input time dose conc @@;
  if (dose = 4) then group=1; else group=2;
  datalines;
 0.00   4  0.1633  0.25   4   2.045
 0.27   4     4.4  0.30   4    7.37
 0.35   4    1.89  0.37   4    2.89
 0.50   4    3.96  0.57   4    6.57
 0.58   4     6.9  0.60   4     4.6
 0.63   4    9.03  0.77   4    5.22
 1.00   4    7.82  1.02   4   7.305
 1.05   4    7.14  1.07   4     8.6
 1.12   4    10.5  2.00   4    9.72
 2.02   4    7.93  2.05   4    7.83
 2.13   4    8.38  3.50   4    7.54
 3.52   4    9.75  3.53   4    5.66
 3.55   4   10.21  3.62   4     7.5
 3.82   4    8.58  5.02   4   6.275
 5.05   4    9.18  5.07   4    8.57
 5.08   4     6.2  5.10   4    8.36
 7.02   4    5.78  7.03   4    7.47
 7.07   4   5.945  7.08   4    8.02
 7.17   4    4.24  8.80   4    4.11
 9.00   4     4.9  9.02   4    5.33
 9.03   4    6.11  9.05   4    6.89
 9.38   4    7.14 11.60   4    3.16
11.98   4    4.19 12.05   4    4.57
12.10   4    5.68 12.12   4    5.94
12.15   4     3.7 23.70   4    2.42
24.15   4    1.17 24.17   4    1.05
24.37   4    3.28 24.43   4    1.12
24.65   4    1.15  0.00   5   0.025
```

```
 0.25   5    2.92  0.27   5   1.505
 0.30   5    2.02  0.50   5   4.795
 0.52   5    5.53  0.58   5    3.08
 0.98   5   7.655  1.00   5   9.855
 1.02   5    5.02  1.15   5    6.44
 1.92   5    8.33  1.98   5    6.81
 2.02   5  7.8233  2.03   5    6.32
 3.48   5    7.09  3.50   5   7.795
 3.53   5    6.59  3.57   5    5.53
 3.60   5    5.87  5.00   5     5.8
 5.02   5  6.2867  5.05   5    5.88
 6.98   5    5.25  7.00   5    4.02
 7.02   5    7.09  7.03   5   4.925
 7.15   5    4.73  9.00   5    4.47
 9.03   5    3.62  9.07   5    4.57
 9.10   5     5.9  9.22   5    3.46
12.00   5    3.69 12.05   5    3.53
12.10   5    2.89 12.12   5    2.69
23.85   5    0.92 24.08   5    0.86
24.12   5    1.25 24.22   5    1.15
24.30   5     0.9 24.35   5    1.57
;
```

The following code plots the theophylline concentration data over time for the two groups (Output 60.5.1). In each group the concentration tends to rise sharply right after the drug is administered, followed by a prolonged tapering of the concentration.

```
proc sgplot data=theop;
   scatter x=time y=conc / group=group;
   yaxis label='Concentration';
   xaxis label='Time';
run;
```

Output 60.5.1 Observed Responses in Two Groups



In the context of nonlinear mixed models, Pinheiro and Bates (1995) consider a first-order compartment model for these data. In terms of two fixed treatment groups, the model can be written as

$$C_{it} = \frac{D k_{e_i} k_{a_i}}{Cl_i (k_{a_i} - k_{e_i})} [\exp(-k_{e_i} t) - \exp(-k_{a_i} t)] + \epsilon_{it}$$

where C_{it} is the observed concentration in group i at time t, D is the dose of theophylline, k_{e_i} is the elimination rate in group i, k_{a_i} is the absorption rate in group i, Cl_i is the clearance in group i, and ϵ_{it} denotes the model error. Because the rates and the clearance must be positive, you can parameterize the model in terms of log rates and the log clearance:

$$\begin{aligned} Cl_i &= \exp\{\beta_{1i}\} \\ k_{a_i} &= \exp\{\beta_{2i}\} \\ k_{e_i} &= \exp\{\beta_{3i}\} \end{aligned}$$

In this parameterization the model contains six parameters, and the rates and clearance vary by group. This produces two separate response profiles, one for each group. On the other extreme, you could model the trends as if there were no differences among the groups:

$$\begin{aligned} Cl_i &= \exp\{\beta_1\} \\ k_{a_i} &= \exp\{\beta_2\} \\ k_{e_i} &= \exp\{\beta_3\} \end{aligned}$$

In between these two extremes lie other models, such as a model where both groups have the same absorption and elimination rate, but different clearances. The question then becomes how to go about building a model in an organized manner.

To test hypotheses about nested nonlinear models, you can apply the idea of a "Sum of Squares Reduction Test." A reduced model is nested within a full model if you can impose q constraints on the full model to obtain the reduced model. Then, if SSE_r and SSE_f denote the residual sum of squares in the reduced and the full model, respectively, the test statistic is

$$F_R = \frac{\left(\mathrm{SSE}_r - \mathrm{SSE}_f\right)/q}{\mathrm{SSE}_f/(n-p)} = \frac{\left(\mathrm{SSE}_r - \mathrm{SSE}_f\right)/q}{\mathrm{MSE}_f}$$

where n are the number of observations used and p are the number of parameters in the full model. The numerator of the F_R statistic is the average reduction in residual sum of squares per constraint. The mean squared error of the full model is used to scale this average because it is less likely to be a biased estimator of the residual variance than the variance estimate from a constrained (reduced) model. The F_R statistic is then compared against quantiles from an F distribution with q numerator and $n-p$ denominator degrees of freedom. Schabenberger and Pierce (2002) discuss the justification for this test and compare it to other tests in nonlinear models.

In the present application we might phrase the initial question akin to the overall F test for a factor in a linear model: Should any parameters be varied between the two groups? The corresponding null hypothesis is

$$H\colon \begin{cases} \beta_{11} = \beta_{12} \\ \beta_{21} = \beta_{22} \\ \beta_{31} = \beta_{32} \end{cases}$$

where the first subscript identifies the type of the parameter and the second subscript identifies the group. Note that this hypothesis implies

$$H\colon \begin{cases} Cl_1 = Cl_2 \\ k_{a_1} = k_{a_2} \\ k_{e_1} = k_{e_2} \end{cases}$$

If you fail to reject this hypothesis, there is no need to further examine individual parameter differences.

The reduced model—the model subject to the null hypothesis—is fit with the following PROC NLIN statements:

```
proc nlin data=theop;
   parms beta1=-3.22 beta2=0.47 beta3=-2.45;
   cl   = exp(beta1);
   ka   = exp(beta2);
   ke   = exp(beta3);
   mean = dose*ke*ka*(exp(-ke*time)-exp(-ka*time))/cl/(ka-ke);
   model conc = mean;
   ods output Anova=aovred(rename=(ss=ssred ms=msred df=dfred));
run;
```

The clearance, the rates, and the mean function are formed independently of the group membership. The analysis of variance table is saved to the data set aovred, and some of its variables are renamed. This is done so that the data set can be merged easily with the analysis of variance table for the full model (see following).

The converged model has a residual sum of square of $\text{SSE}_r = 286.4$ and a mean squared error of 3.0142 (Output 60.5.2). The table of parameter estimates gives the values for the estimated log clearance ($\widehat{\beta}_1 = -3.2991$), the estimated log absorption rate ($\widehat{\beta}_2 = 0.4769$), and the estimated log elimination rate ($\widehat{\beta}_3 = -2.5555$).

Output 60.5.2 Fit Results for the Reduced Model

```
                          The NLIN Procedure
                       Dependent Variable conc
                        Method: Gauss-Newton

         NOTE: Convergence criterion met.

               NOTE: An intercept was not specified for this model.

                                     Sum of        Mean               Approx
    Source                    DF    Squares      Square    F Value    Pr > F

    Model                      3     3100.5      1033.5     342.87    <.0001
    Error                     95      286.4      3.0142
    Uncorrected Total         98     3386.8

                                        Approx      Approximate 95% Confidence
           Parameter      Estimate    Std Error               Limits

           beta1           -3.2991       0.0956     -3.4888     -3.1094
           beta2            0.4769       0.1640      0.1512      0.8025
           beta3           -2.5555       0.1410     -2.8354     -2.2755
```

The full model, in which all three parameters are varied by group, can be fit with the following statements in the NLIN procedure:

```
proc nlin data=theop;
   parms beta1_1=-3.22 beta2_1=0.47 beta3_1=-2.45
         beta1_2=-3.22 beta2_2=0.47 beta3_2=-2.45;
   if (group=1) then do;
      cl   = exp(beta1_1);
      ka   = exp(beta2_1);
      ke   = exp(beta3_1);
   end; else do;
      cl   = exp(beta1_2);
      ka   = exp(beta2_2);
      ke   = exp(beta3_2);
   end;
   mean = dose*ke*ka*(exp(-ke*time)-exp(-ka*time))/cl/(ka-ke);
   model conc = mean;
   ods output Anova=aovfull;
run;
```

Separate parameters for the groups are now specified in the PARMS statement, and the value of the model variables cl, ka, and ke is assigned conditional on the group membership of an observation. Notice that the same expression as in the previous run can be used to model the mean function.

The results from this PROC NLIN run are shown in Output 60.5.3. The residual sum of squares in the full model is only $\text{SSE}_f = 138.9$, compared to $\text{SSE}_r = 286.4$ in the reduced model (Output 60.5.3).

Output 60.5.3 Fit Results for the Full Model

```
                              The NLIN Procedure
                            Dependent Variable conc
                             Method: Gauss-Newton

        NOTE: Convergence criterion met.

              NOTE: An intercept was not specified for this model.

                                       Sum of        Mean                Approx
    Source                     DF     Squares      Square    F Value    Pr > F

    Model                       6      3247.9       541.3     358.56    <.0001
    Error                      92       138.9      1.5097
    Uncorrected Total          98      3386.8

                                          Approx      Approximate 95% Confidence
           Parameter      Estimate     Std Error                Limits

           beta1_1         -3.5671        0.0864     -3.7387      -3.3956
           beta2_1          0.4421        0.1349      0.1742       0.7101
           beta3_1         -2.6230        0.1265     -2.8742      -2.3718
           beta1_2         -3.0111        0.1061     -3.2219      -2.8003
           beta2_2          0.3977        0.1987     0.00305       0.7924
           beta3_2         -2.4442        0.1618     -2.7655      -2.1229
```

Whether this reduction in sum of squares is sufficient to declare that the full model provides a significantly better fit than the reduced model depends on the number of constraints imposed on the full model and on the variability in the data. In other words, before drawing any conclusions, you have to take into account how many parameters have been dropped from the model and how much variation around the regression trends the data exhibit. The F_R statistic sets these quantities into relation. The following macro merges the analysis of variance tables from the full and reduced model, and computes F_R and its p-value:

```
%macro SSReductionTest;
   data aov; merge aovred aovfull;
     if (Source='Error') then do;
        Fstat  = ((SSred-SS)/(dfred-df))/ms;
        pvalue = 1-Probf(Fstat,dfred-df,df);
        output;
     end;
   run;
   proc print data=aov label noobs;
      label Fstat  = 'F Value'
            pValue = 'Prob > F';
      format pvalue pvalue8.;
      var Fstat pValue;
   run;
%mend;
%SSReductionTest;
```

Output 60.5.4 F Statistic and P-value for Hypothesis of Equal Trends

```
                                    F Value    Prob > F

                                    32.5589    <.000001
```

There is clear evidence that the model with separate trends fits these data significantly better (Output 60.5.4). To decide whether all parameters should be varied between the groups or only one or two of them, we first refit the model in a slightly different parameterization:

```
proc nlin data=theop;
   parms beta1_1=-3.22 beta2_1=0.47 beta3_1=-2.45
         beta1_diff=0 beta2_diff=0 beta3_diff=0;
   if (group=1) then do;
      cl   = exp(beta1_1);
      ka   = exp(beta2_1);
      ke   = exp(beta3_1);
   end; else do;
      cl   = exp(beta1_1 + beta1_diff);
      ka   = exp(beta2_1 + beta2_diff);
      ke   = exp(beta3_1 + beta3_diff);
   end;
   mean = dose*ke*ka*(exp(-ke*time)-exp(-ka*time))/cl/(ka-ke);
   model conc = mean;
run;
```

In the preceding statements, the parameters in the second group were expressed using offsets from parameters in the first group. For example, the parameter beta1_diff measures the change in log clearance between group 2 and group 1.

This simple reparameterization does not affect the model fit. The analysis of variance tables in Output 60.5.5 and Output 60.5.3 are identical. It does, however, affect the interpretation of the estimated quantities. Since the parameter beta1_diff measures the change in the log clearance rates between the groups, you can use the approximate 95% confidence limits in Output 60.5.5 to assess whether that quantity in the pharmacokinetic equation varies between groups. Only the confidence interval for the difference in the log clearances excludes 0. The intervals for beta2_diff and beta3_diff include 0.

Output 60.5.5 Fit Results for the Full Model in Difference Parameterization

```
                          The NLIN Procedure
                       Dependent Variable conc
                         Method: Gauss-Newton

        NOTE: Convergence criterion met.

              NOTE: An intercept was not specified for this model.

                                     Sum of        Mean               Approx
  Source                    DF      Squares      Square    F Value    Pr > F

  Model                      6       3247.9       541.3     358.56    <.0001
  Error                     92        138.9      1.5097
  Uncorrected Total         98       3386.8

                                        Approx      Approximate 95% Confidence
         Parameter      Estimate    Std Error                Limits

         beta1_1         -3.5671       0.0864     -3.7387     -3.3956
         beta2_1          0.4421       0.1349      0.1742      0.7101
         beta3_1         -2.6230       0.1265     -2.8742     -2.3718
         beta1_diff       0.5560       0.1368      0.2842      0.8278
         beta2_diff      -0.0444       0.2402     -0.5214      0.4326
         beta3_diff       0.1788       0.2054     -0.2291      0.5866
```

This suggests as the final model one where the absorption and elimination rates are the same for both groups and only the clearances are varied. The following statements fit this model and perform the sum of squares reduction test:

```
proc nlin data=theop;
   parms beta1_1=-3.22 beta2_1=0.47 beta3_1=-2.45
         beta1_diff=0;
   ka = exp(beta2_1);
   ke = exp(beta3_1);
   if (group=1) then do;
      cl = exp(beta1_1);
   end; else do;
      cl = exp(beta1_1 + beta1_diff);
   end;
   mean = dose*ke*ka*(exp(-ke*time)-exp(-ka*time))/cl/(ka-ke);
   model conc = mean;
   ods output Anova=aovred(rename=(ss=ssred ms=msred df=dfred));
   output out=predvals predicted=p;
run;
%SSReductionTest;
```

The results for this model with common absorption and elimination rates are shown in Output 60.5.6. The sum-of-squares reduction test comparing this model against the full model with six parameters shows—as expected—that the full model does not fit the data significantly better ($p = 0.6193$, Output 60.5.7).

Output 60.5.6 Fit Results for Model with Common Rates

```
                          The NLIN Procedure
                        Dependent Variable conc
                         Method: Gauss-Newton

         NOTE: Convergence criterion met.

               NOTE: An intercept was not specified for this model.

                                       Sum of        Mean                Approx
  Source                        DF    Squares      Square    F Value    Pr > F

  Model                          4     3246.5       811.6     543.60    <.0001
  Error                         94      140.3      1.4930
  Uncorrected Total             98     3386.8

                                            Approx      Approximate 95% Confidence
         Parameter      Estimate       Std Error                Limits

         beta1_1         -3.5218          0.0681      -3.6570      -3.3867
         beta2_1          0.4226          0.1107       0.2028       0.6424
         beta3_1         -2.5571          0.0988      -2.7532      -2.3610
         beta1_diff       0.4346          0.0454       0.3444       0.5248
```

Output 60.5.7 F Statistic and P-value for Hypothesis of Common Rates

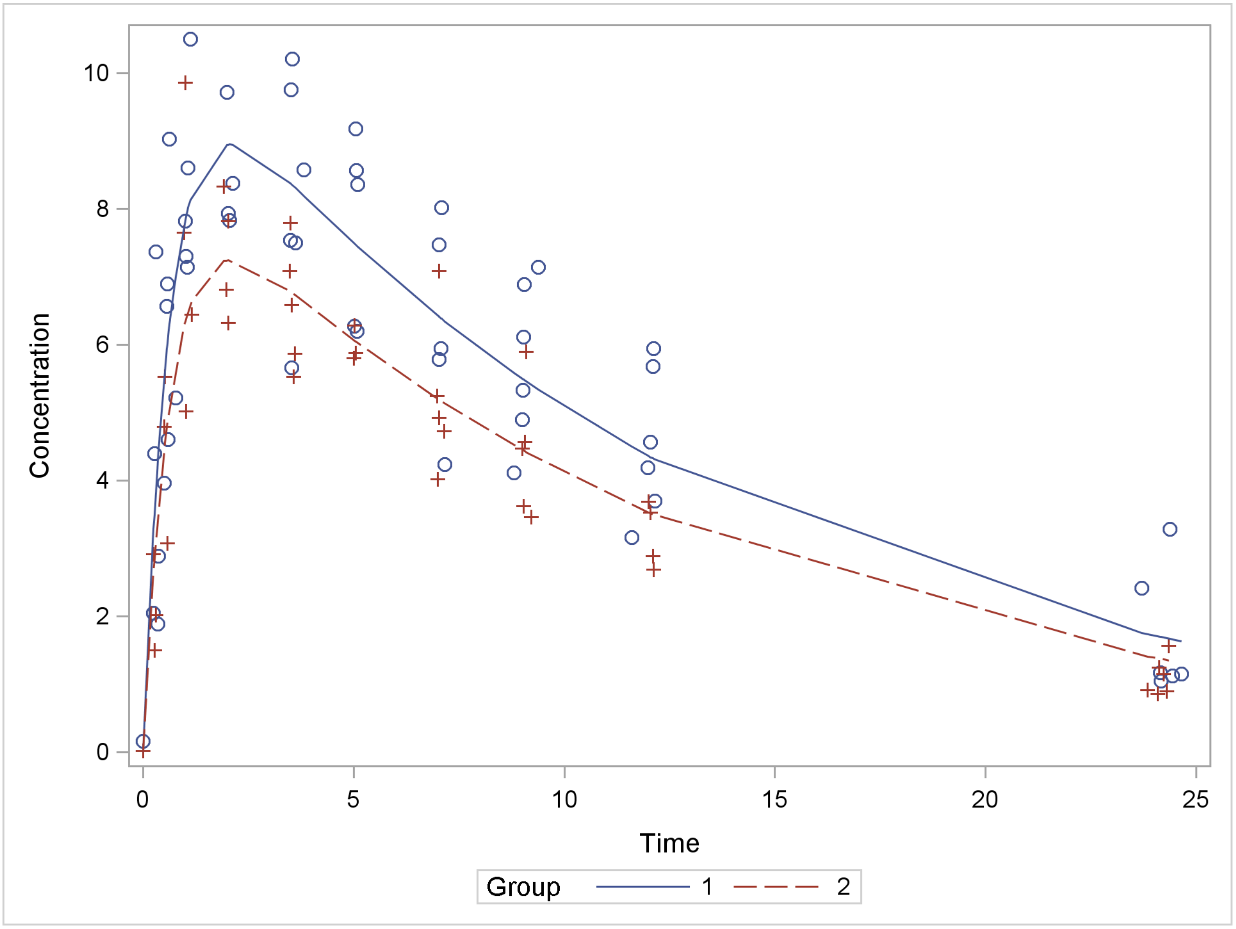
```
                         F Value    Prob > F

                         0.48151    0.619398
```

A plot of the observed and predicted values for this final model is obtained with the following statements:

```
proc sgplot data=predvals;
   scatter x=time y=conc / group=group;
   series  x=time y=p    / group=group name='fit';
   keylegend 'fit'       / across=2 title='Group';
   yaxis label='Concentration';
   xaxis label='Time';
run;
```

The plot is shown in Output 60.5.8.

Output 60.5.8 Observed and Fitted Values for Theophylline Data



The sum-of-squares reduction test is not the only possibility of performing linear-model style hypothesis testing in nonlinear models. You can also perform Wald-type testing of linear hypotheses about the parameter estimates. See Example 38.17 in Chapter 38, “The GLIMMIX Procedure,” for an application of this example that uses the NLIN and GLIMMIX procedures to compare the parameters across groups and adjusts *p*-values for multiplicity.

References

Bard, J. (1970), "Comparison of Gradient Methods for the Solution of the Nonlinear Parameter Estimation Problem," *SIAM Journal of Numerical Analysis*, 7, 157–186.

Bard, J. (1974), *Nonlinear Parameter Estimation*, New York: Academic Press.

Bates, D. M., and Watts, D. L. (1981), "A Relative Offset Orthogonality Convergence Criterion for Nonlinear Least Squares" *Technometrics*, 123, 179–183.

Beaton, A. E. and Tukey, J. W. (1974), "The Fitting of Power Series, Meaning Polynomials, Illustrated on Band-Spectroscopic Data," *Technometrics*, 16, 147–185.

Charnes, A., Frome, E. L., and Yu, P. L. (1976), "The Equivalence of Generalized Least Squares and Maximum Likelihood Estimation in the Exponential Family," *Journal of the American Statistical Association*, 71, 169–172.

Cox, D. R. (1970), *Analysis of Binary Data*, London: Chapman & Hall.

Finney, D. J. (1971), *Probit Analysis*, Third Edition, Cambridge: Cambridge University Press.

Gallant, A. R. (1975), "Nonlinear Regression," *American Statistician*, 29, 73–81.

Gill, P. E., Murray, W., and Wright, M. H. (1981), *Practical Optimization*, New York: Academic Press.

Goodnight, J. H. (1979), "A Tutorial on the Sweep Operator," *American Statistician*, 33, 149–158.

Hartley, H. O. (1961), "The Modified Gauss-Newton Method for the Fitting of Non-linear Regression Functions by Least Squares," *Technometrics*, 3, 269–280.

Holland, P. H. and Welsch, R. E. (1977), "Robust Regression Using Iteratively Reweighted Least-Squares," *Communications Statistics: Theory and Methods*, 6, 813–827.

Hougaard, P. (1982), "Parameterizations of Non-linear Models," *Journal of the Royal Statistical Society, Series B*, 244–252.

Hougaard, P. (1985), "The Appropriateness of the Asymptotic Distribution in a Nonlinear Regression Model in Relation to Curvature," *Journal of the Royal Statistical Society, Series B*, 103–114.

Huber, P. J. (1964), "Robust Estimation of a Location Parameter," *Annals of Mathematical Statistics*, 35, 73–101.

Huber, P. J. (1973), "Robust Regression: Asymptotics, Conjectures, and Monte Carlo," *Annals of Statistics*, 1, 799–821.

Jennrich, R. I. (1969), "Asymptotic Properties of Nonlinear Least Squares Estimators," *Annals of Mathematical Statistics*, 40, 633–643.

Jennrich, R. I. and Moore, R. H. (1975), "Maximum Likelihood Estimation by Means of Nonlinear

Least Squares," *American Statistical Association*, *1975 Proceedings of the Statistical Computing Section*, 57–65.

Jennrich, R. I. and Sampson, P. F. (1968), "Application of Stepwise Regression to Non-linear Estimation," *Technometrics*, 10, 63–72.

Judge, G. G., Griffiths, W. E., Hill, R. C., and Lee, T.-C. (1980), *The Theory and Practice of Econometrics*, New York: John Wiley & Sons.

Kennedy, W. J. and Gentle, J. E. (1980), *Statistical Computing*, New York: Marcel Dekker.

Lee, E. T. (1974), "A Computer Program for Linear Logistic Regression Analysis," *Computer Programs in Biomedicine*, 80–92.

Marquardt, D. W. (1963), "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *Journal for the Society of Industrial and Applied Mathematics*, 11, 431–441.

McCullagh, P. and Nelder, J. A. (1989), *Generalized Linear Models*, Second Edition, New York: Chapman & Hall.

Nelder, J. A. and Wedderburn, R. W. M. (1972), "Generalized Linear Models," *Journal of the Royal Statistical Society, Series A*, 135, 370–384.

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.

Pringle, R. M. and Rayner, A. A. (1971), *Generalized Inverse Matrices with Applications to Statistics*, New York: Hafner Publishing Co.

Ratkowsky, D. (1983), *Nonlinear Regression Modeling*, New York and Basel: Marcel Dekker.

Ratkowsky, D. (1990), *Handbook of Nonlinear Regression Models*, New York and Basel: Marcel Dekker.

Schabenberger, O. and Pierce, F. J. (2002), *Contemporary Statistical Models for the Plant and Soil Sciences*, Boca Raton, FL: CRC Press.

Schabenberger, O., Tharp, B. E., Kells, J. J., and Penner, D. (1999), "Statistical Tests for Hormesis and Effective Dosages in Herbicide Dose Response," *Agronomy Journal*, 91, 713–721.

Seber, G. A. F. and Wild, C. J. (1989) *Nonlinear Regression*, New York: John Wiley & Sons.

Subject Index

g2 inverse
    NLIN procedure, 4926
g4 inverse
    NLIN procedure, 4927

alpha level
    NLIN procedure, 4913

comparing trends
    NLIN procedure, 4955
confidence intervals
    model confidence interval (NLIN), 4931
    parameter confidence interval (NLIN), 4930

derivatives
    NLIN procedure, 4913

examples, NLIN
    array, constant, 4935
    array, variable, 4935
    biweight, 4943
    boundary specifications, 4907
    cancer remission data, 4946
    conditional model expression, 4938
    constant array, 4935
    contrasts among parameters, 4961
    convergence status, 4936
    derivative code, 4917
    derivatives, output, 4913
    derivatives, zero, 4923
    discontinuity, 4924
    divergence, 4924
    dose-response data, 4949
    enzyme data, 4897
    expected value parameterization, 4953
    GENMOD procedure, 4947
    getting started, 4896
    GLIMMIX procedure, 4947
    Hougaard's skewness measure, 4897, 4950, 4952, 4953
    iteratively reweighted least squares, 4943
    join point, 4938
    join point, estimated, 4941
    LD50 parameterization, 4952
    local minimum, 4924
    log-logistic model, 4950
    LOGISTIC procedure, 4947
    machine-level code, 4918
    maximum likelihood estimation, 4946
    model code, 4917
    Newton-Raphson algorithm, 4946
    non-identifiable parameter, 4923
    one-compartment model, 4959
    output, derivatives, 4913
    output, predicted, 4953
    output, predicted values, 4938
    output, prediction limits, 4953
    parameter differences, 4961
    pharmacokinetic model, 4959
    plateau model, 4938
    plot, observed and predicted, 4940, 4954
    predicted values, 4938
    probit model, 4946
    programming statements, 4916
    programming statements, efficiency, 4921
    PUT statement, 4938
    reparameterization, 4941, 4952, 4953, 4961
    reparameterizing constraint, 4908
    robust regression, 4943
    ROBUSTREG procedure, 4944
    segmented model, 4938
    SGPLOT procedure, 4940, 4949, 4950, 4954, 4956, 4964
    SIGSQ= option, 4946
    starting values, data set, 4915
    starting values, grid, 4914
    starting values, multiple, 4897
    sum-of-squares reduction test, 4961, 4963
    switching function, 4950
    theophylline data, 4955
    U.S. population growth data, 4943
    variable array, 4935
    varying parameters by groups, 4960
    weight variable, 4943
    weighted least-squares, 4922

generalized inverse
    NLIN procedure, 4926
group comparisons
    NLIN procedure, 4955

initial values
    NLIN procedure, 4913

Jacobian
    NLIN procedure, 4903

Lagrange multiplier

covariance matrix, 4932
NLIN procedure, 4908
log-logistic model
NLIN procedure, 4948

Moore-Penrose inverse
NLIN procedure, 4927

NLIN procedure
g2 inverse, 4926
g4 inverse, 4927
alpha level, 4913
analytic derivatives, 4909, 4917
automatic derivatives, 4917
close-to-linear, 4919
comparing trends, 4955
confidence interval, 4911, 4912, 4930, 4931
convergence, 4923
convergence criterion, 4903
covariance matrix, 4931
cross-referencing variables, 4907
debugging execution, 4904
derivatives, 4909, 4913, 4917
G4 inverse, 4904
Gauss iterative method, 4905
Gauss-Newton iterative method, 4927, 4928
Gauss-Newton method, 4926
generalized inverse, 4926
gradient method, 4926, 4929
group comparisons, 4955
Hessian, 4932
Hougaard's measure, 4904, 4919
incompatibilities, 4934, 4935
initial values, 4913
iteratively reweighted least squares example, 4942
Jacobian, 4903
Lagrange multiplier, 4908
Lagrange multipliers, covariance matrix, 4932
log-logistic model, 4948
Marquardt iterative method, 4905, 4926, 4929
maximum iterations, 4905
maximum subiterations, 4905
mean square error specification, 4906
missing values, 4921
model confidence interval, 4931
model.variable syntax, 4910
Moore-Penrose inverse, 4927
Newton iterative method, 4905, 4926–4928
object convergence measure, 4899
options summary (PROC statement), 4902
output table names, 4935
parameter confidence interval, 4930
PPC convergence measure, 4899
predicted values, output, 4911
prediction interval, 4911, 4912
R convergence measure, 4899
residual values, output, 4911
retaining variables, 4909, 4916
RPC convergence measure, 4899
segmented model example, 4937
singularity criterion, 4906
skewness, 4900, 4904, 4919
SMETHOD=GOLDEN step size search, 4930
special variables, 4921
specifying bounds, 4907
standard error, 4912
starting values, 4913
steepest descent method, 4926, 4929
step size search, 4930
switching model, 4948
trend comparisons, 4955
troubleshooting, 4923
tuning display of iteration computation, 4906
weighted regression, 4922

options summary (PROC statement)
NLIN procedure, 4902

PPC convergence measure, 4932
predicted values
NLIN procedure, 4911

R convergence measure, 4932
residuals
NLIN procedure, 4911
RPC convergence measure, 4932

S convergence measure, 4932
singularity criterion
NLIN procedure, 4906
skewness
NLIN procedure, 4904
starting values
NLIN procedure, 4913
switching model
NLIN procedure, 4948

theophylline data
examples, NLIN, 4955
trend comparisons
NLIN procedure, 4955

Syntax Index

ALPHA= option
    OUTPUT statement (NLIN), 4913
    PROC NLIN statement, 4903

BEST= option
    PROC NLIN statement, 4903
BOUNDS statement
    NLIN procedure, 4907
BY statement
    NLIN procedure, 4908

CONTROL statement
    NLIN procedure, 4909
CONVERGE= option
    PROC NLIN statement, 4903
CONVERGEOBJ= option
    PROC NLIN statement, 4903
CONVERGEPARM= option
    PROC NLIN statement, 4904

DATA= option
    PROC NLIN statement, 4904
DER option
    OUTPUT statement (NLIN), 4913
DER statement
    NLIN procedure, 4909

FLOW option
    PROC NLIN statement, 4904

G4 option
    PROC NLIN statement, 4904

H= option
    OUTPUT statement (NLIN), 4911
HOUGAARD option
    PROC NLIN statement, 4904

ID statement
    NLIN procedure, 4909

L95= option
    OUTPUT statement (NLIN), 4911
L95M= option
    OUTPUT statement (NLIN), 4911
LCL= option
    OUTPUT statement (NLIN), 4911
LCLM= option
    OUTPUT statement (NLIN), 4911
LIST option
    PROC NLIN statement, 4905
LISTALL option
    PROC NLIN statement, 4905
LISTCODE option
    PROC NLIN statement, 4905
LISTDEP option
    PROC NLIN statement, 4905
LISTDER option
    PROC NLIN statement, 4905

MAXITER= option
    PROC NLIN statement, 4905
MAXSUBIT= option
    PROC NLIN statement, 4905
METHOD= option
    PROC NLIN statement, 4905
MODEL statement
    NLIN procedure, 4910

NLIN procedure
    syntax, 4901
NLIN procedure, BOUNDS statement, 4907
NLIN procedure, BY statement, 4908
NLIN procedure, CONTROL statement, 4909
NLIN procedure, DER statement, 4909
NLIN procedure, ID statement, 4909
NLIN procedure, MODEL statement, 4910
NLIN procedure, OUTPUT statement, 4910
    ALPHA= option, 4913
    DER option, 4913
    H= option, 4911
    L95= option, 4911
    L95M= option, 4911
    LCL= option, 4911
    LCLM= option, 4911
    OUT= option, 4910
    PARMS= option, 4911
    PREDICTED= option, 4911
    RESIDUAL= option, 4911
    SSE= option, 4912
    STDI= option, 4912
    STDP= option, 4912
    STDR= option, 4912
    STUDENT= option, 4912
    U95= option, 4912
    U95M= option, 4912
    UCL= option, 4912
    UCLM= option, 4912
    WEIGHT= option, 4912

NLIN procedure, PARAMETERS statement, 4913
NLIN procedure, PROC NLIN statement, 4902
    ALPHA= option, 4903
    BEST= option, 4903
    CONVERGE= option, 4903
    CONVERGEOBJ= option, 4903
    CONVERGEPARM= option, 4904
    DATA= option, 4904
    FLOW option, 4904
    G4 option, 4904
    HOUGAARD option, 4904
    LIST option, 4905
    LISTALL option, 4905
    LISTCODE option, 4905
    LISTDEP option, 4905
    LISTDER option, 4905
    MAXITER= option, 4905
    MAXSUBIT= option, 4905
    METHOD= option, 4905
    NOHALVE option, 4906
    NOITPRINT option, 4906
    NOPRINT option, 4906
    OUTEST= option, 4906
    PRINT option, 4906
    RHO= option, 4906
    SAVE option, 4906
    SIGSQ= option, 4906
    SINGULAR= option, 4906
    SMETHOD= option, 4907
    TAU= option, 4907
    TOTALSS option, 4907
    TRACE option, 4907
    XREF option, 4907
NLIN procedure, program statements, 4916
NLIN procedure, programming statements, 4916
NLIN procedure, RETAIN statement, 4916
NOHALVE option
    PROC NLIN statement, 4906
NOITPRINT option
    PROC NLIN statement, 4906
NOPRINT option
    PROC NLIN statement, 4906

OUT= option
    OUTPUT statement (NLIN), 4910
OUTEST= option
    PROC NLIN statement, 4906
OUTPUT statement
    NLIN procedure, 4910

PARAMETERS statement
    NLIN procedure, 4913
PARMS= option
    OUTPUT statement (NLIN), 4911
PREDICTED= option
    OUTPUT statement (NLIN), 4911
PRINT option
    PROC NLIN statement, 4906
PROC NLIN statement, *see* NLIN procedure

RESIDUAL= option
    OUTPUT statement (NLIN), 4911
RETAIN statement
    NLIN procedure, 4916
RHO= option
    PROC NLIN statement, 4906

SAVE option
    PROC NLIN statement, 4906
SIGSQ= option
    PROC NLIN statement, 4906
SINGULAR= option
    PROC NLIN statement, 4906
SMETHOD= option
    PROC NLIN statement, 4907
SSE= option
    OUTPUT statement (NLIN), 4912
STDI= option
    OUTPUT statement (NLIN), 4912
STDP= option
    OUTPUT statement (NLIN), 4912
STDR= option
    OUTPUT statement (NLIN), 4912
STUDENT= option
    OUTPUT statement (NLIN), 4912

TAU= option
    PROC NLIN statement, 4907
TRACE option
    PROC NLIN statement, 4907
TTOTALSS option
    PROC NLIN statement, 4907

U95= option
    OUTPUT statement (NLIN), 4912
U95M= option
    OUTPUT statement (NLIN), 4912
UCL= option
    OUTPUT statement (NLIN), 4912
UCLM= option
    OUTPUT statement (NLIN), 4912

WEIGHT= option
    OUTPUT statement (NLIN), 4912

XREF option
    PROC NLIN statement, 4907

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`**.