Chapter 45
The NLIN Procedure

Chapter Table of Contents

Chapter 45
The NLIN Procedure

Overview

The NLIN procedure produces least squares or weighted least squares estimates of the parameters of a nonlinear model. Nonlinear models are more difficult to specify and estimate than linear models. Instead of simply listing regressor variables, you must write the regression expression, declare parameter names, and supply initial parameter values. Some models are difficult to fit, and there is no guarantee that the procedure can fit the model successfully.

For each nonlinear model to be analyzed, you must specify the model (using a single dependent variable) and the names and starting values of the parameters to be estimated.

Using PROC NLIN, you can also

- 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, residuals, parameter estimates and SSE at each iteration, the covariance matrix of parameter estimates, and other statistics
- define your own objective function to be minimized

Estimation of a nonlinear model is an iterative process. To begin this process the NLIN procedure first examines the starting value specifications of the parameters. If a grid of values is specified, PROC NLIN evaluates the residual sum of squares at each combination of parameter values to determine the set of parameter values producing the lowest residual sum of squares. These parameter values are used for the initial step of the iteration.

Then PROC NLIN uses one of these five iterative methods:

- steepest-descent or gradient method
- Newton method
- modified Gauss-Newton method
- Marquardt method
- multivariate secant or false position (DUD) 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.

You can use the NLIN procedure for segmented models (see Example 45.1) or robust regression (see Example 45.2). You can also use it to compute maximum-likelihood estimates for certain models (refer to Jennrich and Moore 1975; Charnes, Frome, and Yu 1976).

Getting Started

The NLIN procedure performs univariate nonlinear regression using the least squares method. Nonlinear regression analysis is indicated when you have information specifying that the functional relationship between the predictor and response variables is nonlinear in the parameters. Such information might come from direct knowledge of the true model, theoretical developments, or previous studies. *Nonlinear*, in this sense, means that the mathematical relationship between the variables and parameters is not required to have a linear form. For example, consider the following two models:

$$Y = aX^2 + b$$

$$Y = \frac{1}{a}X + b$$

where a and b are parameters and X and Y are random variables. The first model is linear in the parameters; the second model is nonlinear.

Estimating 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.

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

where x_i represents the amount of substrate for n trials and $f(\boldsymbol{x}, \boldsymbol{\theta})$ is the velocity of the reaction. The vector $\boldsymbol{\theta}$ contains the rate parameters.

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. Your data set is as follows:

```
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 double trailing at sign (@@) in the INPUT statement specifies that observations are input from each line until all of the values are read.

The following statements request a nonlinear regression analysis:

```
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 MODEL statement specifies the enzymatic reaction model

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

where V represents the velocity or reaction rate and C represents the substrate concentration.

The PARMS statement declares the parameters and specifies their initial values. In this example, the initial estimates in the PARMS statement are obtained as follows. Since the model is a monotonic increasing function in C, and

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

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.

To obtain an initial value for the parameter $theta_2$, 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, which results in a search over the grid points from 0 to 0.07 in increments of 0.01. The output 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 45.1 displays the calculations resulting from the grid search.

The NLIN Procedure
Grid Search
Dependent Variable Velocity

theta1	theta2	Sum of 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 NLIN Procedure
Iterative Phase
Dependent Variable Velocity
Method: Marquardt

Iter	theta1	theta2	Sum of 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 45.1. Nonlinear Least Squares Grid Search from the NLIN Procedure

The parameter Theta1 is held constant at its specified initial value of 155, the grid is traversed, and the residual sums of squares are computed at each point. The "best"

starting value is the point that corresponds to the smallest value of the residual sum of squares. Figure 45.1 shows that the best starting value for Theta2 is 0.06. PROC NLIN uses this point as the initial value for Theta2 in the following iterative phase.

PROC NLIN determines convergence using the relative offset measure of Bates and Watts (1981). When this measure is less than 10^{-5}, convergence is declared. Figure 45.1 displays the iteration history.

The NLIN Procedure

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

Figure 45.2. Estimation Summary from the NLIN Procedure

Figure 45.2 displays a summary of the estimation including several convergence measures R, PPC, RPC, and Object.

The R measure 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 all the parameters) would change by that relative amount were PROC NLIN to take an additional iteration step. The RPC value indicates that Theta2 changed by 0.000078, relative to its value in the last iteration. The Object measure indicates that the objective function value changed 2.902E-7 in relative value from the last iteration.

The NLIN Procedure

NOTE: An intercept was not specified for this model.

Source	DF	Sum of Squares	Mean Square	F Value	Approx Pr > F
Regression	2	290116	145058	88537.2	<.0001
Residual	12	19.6606	1.6384		
Uncorrected Total	14	290135			
Corrected Total	13	1269.7			

Figure 45.3. Nonlinear Least Squares Summary from the NLIN Procedure

Figure 45.3 displays the least squares summary statistics for the model. The degrees of freedom, sums of squares, and mean squares are listed.

The NLIN Procedure

Parameter	Estimate	Approx Std Error	Approximate 95% 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 45.4. Parameter Estimates from the NLIN Procedure

Figure 45.4 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 parameters are nearly linear 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

$$\hat{\mathrm{V}} = \frac{158.105C}{0.0741 + C}$$

where V represents the velocity or rate of the reaction, and C represents the substrate concentration.

Syntax

PROC NLIN < *options* > **;**
MODEL *dependent=expression* **;**
PARAMETERS *parameter=values* <,. . ., *parameter=values*>**;**
other program statements
BOUNDS *inequality* < , . . . , *inequality* > **;**
BY *variables* **;**
DER.*parameter=expression* **;**
DER.*parameter.parameter=expression* **;**
ID *variables* **;**
OUTPUT OUT=*SAS-data-set keyword=names* <,. . .,*keyword=names*>**;**
CONTROL *variable* <=*values*> < . . . *variable* <=*values*>> **;**

A vertical bar (|) denotes a choice between two specifications. The *other program statements* 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. The NLIN procedure automatically creates several variables that are also available for use in the analysis. See the section "Special Variables" beginning on page 2395 for more information. The PROC NLIN, PARMS, and MODEL statements are required.

The statements used in PROC NLIN, in addition to the PROC 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
OUTPUT	creates an output data set containing statistics for each observation
PARMS	identifies parameters to be estimated and the starting values for each parameter
other program statements	includes assignment statements, ARRAY statements, DO loops, and program control statements

PROC NLIN Statement

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

The PROC NLIN statement invokes the procedure. The following table lists the options available with the PROC NLIN statement. Explanations follow in alphabetical order.

Task	Options
Specify data sets	DATA= OUTEST= SAVE
Grid search	BEST=
Choose an iteration method	METHOD=
Control step size	MAXSUBIT= RHO= SMETHOD= TAU=
Specify details of iteration	G4 SIGSQ=
Minimization Tuning	CONVERGE= CONVERGEOBJ= CONVERGEPARM= SINGULAR= MAXITER=
Modify Amount of Output	HOUGAARD NOITPRINT NOPRINT
List Model Structure	LIST LISTALL LISTCODE LISTDEP LISTDER XREF
Trace Model Execution	FLOW PRINT TRACE

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 criteria for PROC NLIN. For all iterative methods except for METHOD=DUD, the relative offset convergence measure of Bates and Watts is used to determine convergence. This measure is labeled "R" in the Estimation Summary table. The iterations are said to have converged for CONVERGE=c if

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

where r is the residual vector and X is the Jacobian matrix. The default LOSS function is the sum of squared errors (SSE). 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= option, convergence is assumed.

CONVERGEOBJ=*c (DUD only)*

uses the change in the LOSS function as the convergence criterion. For more details on the LOSS function, see the section "Special Variable Used to Determine Convergence Criteria" on page 2396. The iterations are said to have converged for CONVERGEOBJ=c if

$$\frac{\text{LOSS}^{i-1} - \text{LOSS}^i}{\text{LOSS}^i + 10^{-6}} < c$$

where LOSS^i is the LOSS for the ith iteration. The default LOSS function is the sum of squared errors (SSE). The constant c should be a small positive number. See the "Computational Methods" section beginning on page 2398 for more details. By default, CONVERGE=10^{-8}.

CONVERGEPARM=*c (DUD only)*

uses the maximum change among parameter estimates as the convergence 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 β_j^i is the value of the jth parameter at the ith iteration.

The default convergence criterion for DUD is CONVERGEOBJ. If you specify CONVERGEOBJ=*c*, the specified c is used instead of the default of 10^{-8}. If you specify CONVERGEPARM=*c*, the maximum change in parameters is used as the convergence criterion (instead of LOSS). 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 SAS data set containing the data 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 (g_4) inverse in parameter estimation. Refer to Kennedy and Gentle (1980) for details.

HOUGAARD
adds Hougaard's measure of skewness to the parameter estimation table. Computation of the measure requires derivatives (see the section "Hougaard's Measure of Skewness" on page 2394). Thus, the HOUGAARD option is ignored if you specify the option METHOD=DUD, which is described later in this list.

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, since intermediate results for common subexpressions can be reused but are shown in expanded form by the LIST option. To see how the expression is actually evaluated, see the description for the LISTCODE option, which follows.

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

LISTCODE
displays the derivative tables and 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 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 the derivative expression.

MAXITER=*i*
limits the number of iterations PROC NLIN performs before it gives up trying to converge. The i value must be a positive integer. By default, MAXITER=100.

MAXSUBIT=*i*
places a limit on the number of step halvings. By default, MAXSUBIT=30. The value of MAXSUBIT must be a positive integer.

METHOD=GAUSS | MARQUARDT | NEWTON | GRADIENT | DUD
specifies the iterative method that PROC NLIN uses. The GAUSS, MARQUARDT and NEWTON methods are more robust than the GRADIENT and DUD methods. If you omit the METHOD= option, METHOD=GAUSS is used. See the "Computational Methods" section beginning on page 2398 for more information.

NOITPRINT
suppresses the display of the results of each iteration.

NOPRINT
suppresses the display of the output. Note that this option temporarily disables the Output Delivery System (ODS). For more information, see Chapter 15, "Using the Output Delivery System."

OUTEST=*SAS dataset*
specifies an output data set to contain the parameter estimates produced at each iteration. See the "Output Data Sets" section on page 2403 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 on permanent SAS data sets.

PRINT
displays the result of each statement in the program as it is executed. This option produces large amounts of output.

RHO=*value*
specifies a value to use in controlling the step-size search. By default, RHO=0.1 except when METHOD=MARQUARDT, in which case RHO=10. See the section "Computational Methods" beginning on page 2398 for more details.

SAVE
specifies that, when the iteration limit is exceeded, the parameter estimates from the final iteration are output to the OUTEST= data set. These parameter estimates are located in the observation with _TYPE_=FINAL. If you omit the SAVE option, the parameter estimates from the final iteration are not output to the data set unless convergence is attained.

SIGSQ=*value*
specifies a value to replace the mean square error for computing the standard errors of the estimates. The SIGSQ= option is used with 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 approximation to the Hessian. The default value is 1E-8.

SMETHOD=HALVE | GOLDEN | CUBIC
specifies the step-size search method that PROC NLIN uses. The default is SMETHOD=HALVE. See the section "Computational Methods" beginning on page 2398 for details.

TAU=*value*
specifies a value to use in controlling the step-size search. By default, TAU=1 except when METHOD=MARQUARDT, in which case TAU=0.01. See the section "Computational Methods" beginning on page 2398 for details.

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.

BOUNDS Statement

BOUNDS *inequality* $<, \ldots,$ *inequality* $>$ **;**

The BOUNDS statement restrains the parameter estimates within specified bounds. In each BOUNDS statement, you can specify a series of bounds separated by commas. The series of bounds is applied simultaneously. Each bound contains 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, $A + B < 1$), you can reparameterize the model so that the expression becomes a parameter.

For SAS versions 7.01 and later, lagrange multipliers are reported for all bounds that are enforced (active) when the estimation terminates. In the estimates table and in the OUTEST= data set, the Lagrange multiplier estimates are identified with names *Bound1*, *Bound2* ... . An active bound is treated as if a restriction was applied to the set of parameters so one parameter degree of freedom is deducted. The option NOCORRECTEDDF specifies that no degrees of freedom are lost when a bound is active. Lagrange multiplier based bounds are not used for METHOD=DUD.

BY Statement

BY *variables* **;**

You can specify a BY statement with PROC NLIN to obtain separate analyses on 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 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 using the DATASETS procedure.

For more information on the BY statement, refer to the discussion in *SAS Language Reference: Concepts*. For more information on the DATASETS procedure, refer to the discussion in the *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 2390) 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 on

automatic analytical derivatives, see the section "Automatic Derivatives" beginning on page 2391.

For most of the computational methods, you need only specify the first partial derivative for 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.

If you use the _LOSS_ variable, you can specify the derivative of _LOSS_ with respect to the parameters using the DER. syntax.For more information, see the "Special Variable Used to Determine Convergence Criteria" section on page 2396.

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 in the left-hand 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-hand 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 yielding a numeric result. The expression can include parameter names, variables in the data set, and variables created by program 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 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*>**;**

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 3, "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 all the variables in the data set to which PROC NLIN is applied. 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. However, with METHOD=DUD, the following statistics are not available: H, L95, L95M, STDP, STDR, STUDENT, U95, and U95M. These statistics are all calculated using H, the variable containing the leverage. For METHOD=DUD, the approximate Hessian is not available since no derivatives are specified with this method.

H=*name*
specifies a variable to contain the leverage, $x_i(\mathbf{X}'\mathbf{X})^{-1}x_i'$, where $\mathbf{X} = \partial\mathbf{F}/\partial\boldsymbol{\beta}$ and x_i is the ith row of $\mathbf{X}$. If you specify the special variable _WEIGHT_, the leverage is $w_i x_i(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}x_i'$.

L95M=*name*
specifies a variable to contain the lower bound of an approximate 95% confidence interval for the expected value (mean). See also the description for the U95M= option, which follows.

L95=*name*
specifies a variable to contain 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, which follows.

PARMS=*names*
specifies variables in the output data set to contain parameter estimates. These can be the same variable names as listed in the PARAMETERS statement; however, you can choose new names for the parameters identified in the sequence from the PARAMETERS statement. 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 to contain the predicted values of the dependent variable.

RESIDUAL=*name*
R=*name*
specifies a variable in the output data set to contain the residuals (actual values minus predicted values).

SSE=*name*
ESS=*name*
specifies a variable to include in the new data set. The values for the variable are the residual sums of squares finally determined by the procedure. The values of the variable are the same for every observation in the new data set.

STDI=*name*
specifies a variable to contain the standard error of the individual predicted value.

STDP=*name*
specifies a variable to contain the standard error of the mean predicted value.

STDR=*name*
specifies a variable to contain the standard error of the residual.

STUDENT=*name*
specifies a variable to contain the studentized residuals, which are residuals divided by their standard errors.

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

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

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

PARAMETERS Statement

PARAMETERS *parameter=values* . . . **;**
PARMS *parameter=values* . . . **;**

A PARAMETERS (or PARMS) statement must come before the RUN statement. Several parameter names and values can appear. The parameter names must all be valid SAS names and must not duplicate the names of any variables in the data set to which the NLIN procedure is applied. Any parameters specified but not used in the MODEL statement are dropped from the estimation.

In each *parameter=values* specification, the parameter name identifies a parameter to be estimated, both in subsequent procedure statements and in the output. *Values* specify the possible starting values of the parameter.

Usually, only one value is specified for each parameter. If you specify several values for each parameter, PROC NLIN evaluates the model at each point on the grid. The value specifications can take any of several forms:

m	a single value
$m1$, $m2$, . . . , 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

This PARMS statement specifies five parameters and sets their possible starting values as shown:

```
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. (This can take a long time.) See the "Special Variables" section beginning on page 2395 for information on programming parameter starting values.

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 2385) except that it is retained only *within* iterations. An iteration involves a single pass through the data set.

Other Program Statements with PROC NLIN

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 the section "Incompatibilities with 6.11 and Earlier Versions of PROC NLIN" beginning on page 2406.

Several SAS program statements can be used after the PROC NLIN statement. These statements can appear anywhere in the PROC NLIN statement, 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 they use it 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);
```

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);
```

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 program statements to create new SAS variables for the duration of the procedure. These variables are not permanently 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 program statements within PROC NLIN, and special

variables used by PROC NLIN. All of the following SAS program statements can be used in PROC NLIN:

- ARRAY
- assignment (`y = a*x + b;`)
- CALL
- DO
- iterative DO
- DO UNTIL
- DO WHILE
- END
- FILE
- GO TO
- IF-THEN/ELSE
- LINK-RETURN
- PUT (defaults to the list)
- RETURN
- SELECT
- sum (`y + 1;`)

These statements can use the special variables created by PROC NLIN. Consult the section "Special Variables" beginning on page 2395 for more information on special variables.

Details

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 program 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.

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

```
                        The NLIN Procedure

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

   1    471:74        MODEL.Velocity = x1 * EXP(x2
                      * Concentration);
   1    471:74        @MODEL.Velocity/@x1 = EXP(x2
                      * Concentration);
   1    471:74        @MODEL.Velocity/@x2 = x1 * Concentration
                      * EXP(x2 * Concentration);
   1    471:74        @@MODEL.Velocity/@x1/@x2 = Concentration
                      * EXP(x2 * Concentration);
   1    471:74        @@MODEL.Velocity/@x2/@x1 = Concentration
                      * EXP(x2 * Concentration);
   1    471:74        @@MODEL.Velocity/@x2/@x2 = x1
                      * Concentration * Concentration
                      * EXP(x2 * Concentration);
```

Figure 45.5. Model and Derivative Code Output

Note that all the derivatives require the evaluation of EXP(X2 * Concentration). If you specify the LISTCODE option in the PROC NLIN statement, the actual machine level code produced is as follows.

```
                          The NLIN Procedure

                            Code Listing


1 Stmt MODEL      line 328 column 78.
                  (1)
                  arg=MODEL.Velocity
                  argsave=MODEL.
                  Velocity
                  Source Text:          model Velocity = x1 * exp
                                        (x2 * Concentration);
    Oper *        at 328:108 (30,0,2).  * : #temp1 <- x2 Concentration
    Oper EXP      at 328:104            EXP : #temp2 <- #temp1
                  (103,0,1).
    Oper *        at 328:98 (30,0,2).   * : MODEL.Velocity <- x1 #temp2
    Oper eeocf    at 328:104 (18,0,1).  eeocf : _DER_ <- _DER_
    Oper *        at 328:104 (30,0,2).  * : @1dt1_1 <- Concentration #temp2
    Oper =        at 328:98 (1,0,1).    = : @MODEL.Velocity/@x1 <- #temp2
    Oper *        at 328:98 (30,0,2).   * : @MODEL.Velocity/@x2
                                        <- x1 @1dt1_1
    Oper *        at 328:104 (30,0,2).  * : @2dt1_1 <- Concentration
                                        @1dt1_1
    Oper =        at 328:98 (1,0,1).    = : @@MODEL.Velocity/@x1/@x2
                                        <- @1dt1_1
    Oper =        at 328:98 (1,0,1).    = : @@MODEL.Velocity/@x2/@x1
                                        <- @1dt1_1
    Oper *        at 328:98 (30,0,2).   * : @@MODEL.Velocity/@x2/@x2
                                        <- x1 @2dt1_1
```

Figure 45.6. LISTCODE Output

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

Hougaard's Measure of Skewness

A "close-to-linear" nonlinear regression model, first described by Ratkowsky (1990), is a model that produces parameters having properties similar to those produced by a linear regression model. That is, the least squares estimates of the parameters are close to being unbiased, normally distributed, minimum variance estimators.

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 parameters can render inferences using the reported standard errors and confidence limits invalid. You can often fix the problem with reparameterization, replacing the offending parameter by one with better estimation properties.

You can use Hougaard's measure of skewness, g_{1i}, to assess whether a parameter is close to linear or whether it contains considerable nonlinearity. Specify the HOUGAARD option in the PROC NLIN statement to compute Hougaard's measure of skewness.

According to Ratkowsky (1990), if $|g_{1i}| < 0.1$, the estimator $\hat{\theta}_i$ of parameter θ_i is very close-to-linear in behavior and, if $0.1 < |g_{1i}| < .25$, the estimator is reasonably close-to-linear. If $|g_{1i}| > .25$, the skewness is very apparent. For $|g_{1i}| > 1$, the nonlinear behavior is considerable.

Hougaard's measure is computed as follows

$$E[\hat{\theta}_i - E(\hat{\theta}_i)]^3 = -(mse)^2 \sum_{jkl}^{np} L^{ij} L^{ik} L^{il} (W_{jkl} + W_{kjl} + W_{lkj})$$

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

$$L = X' X^{-1}$$

$$W_{jkl} = \sum_{m=1}^{n} J_m^j H_m^{kl}$$

In the preceding equation, J_m is the Jacobian vector and H_m is the Hessian matrix evaluated at observation m. This third moment is normalized using the standard error as

$$g_{1i} = E[\hat{\theta}_i - E(\hat{\theta}_i)]^3 / (mse * L^{ii})^{3/2}$$

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, all the values of all the variables named in the OUTPUT statement are missing values.

Special Variables

Several special variables are created automatically and can be used in PROC NLIN program 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, 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 coding

```
if _model_ then return;
```

after your MODEL statement but before your derivative calculations. 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 convergence criteria:

LOSS is used to determine the criterion function for convergence and step shortening. PROC NLIN looks for the variable _LOSS_ in the program 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 where the residual sum of squares is not the basic criterion.

Weighted Regression with the Special Variable _WEIGHT_

To get weighted least squares estimates of parameters, the _WEIGHT_ variable can be given a value in an assignment statement:

```
_weight_ =  expression;
```

When this statement is included, the expression on the right-hand side of the assignment statement is evaluated for each observation in the data set to be analyzed. The values obtained are 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.

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 can occur in your analysis with PROC NLIN.

Excessive Time

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

The analysis may 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 may be singular, possibly indicating an over-parameterized 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.

You may have a term b2 in the exponent that is nonidentifiable 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 may 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 warning message

```
PROC NLIN failed to converge
```

and displays the results. This often means that the procedure has not converged at all. If you provided the derivatives, check them very closely and then check the sum-of-squares error surface before proceeding. 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 may 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 illegal. For example, consider the following model:

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

Suppose that y happens to be all zero and x is nonzero. There is no least squares estimate for b since the SSE declines as b approaches infinity or minus infinity. The same model could be parameterized with no problem into y = a*x.

If you have divergence problems, try reparameterizing, 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 may 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 may require switching to a different iteration method or an inverse computation method. Specifying the option METHOD=MARQUARDT sometimes works when the default method (Gauss-Newton) does not work.

Computational Methods

For the system of equations represented by the nonlinear model

$$\mathbf{Y} = \mathbf{F}(\beta_0, \beta_1, \ldots, \beta_r, \mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_n) + \epsilon = \mathbf{F}(\boldsymbol{\beta}^*) + \epsilon$$

where $\mathbf{Z}$ is a matrix of the independent variables, $\boldsymbol{\beta}^*$ is a vector of the parameters, ϵ is the error vector, and $\mathbf{F}$ is a function of the independent variables and the parameters, there are two approaches to solving for the minimum. The first method is to minimize

$$L(\boldsymbol{\beta}) = 0.5(\mathbf{e}'\mathbf{e})$$

where $\mathbf{e} = \mathbf{Y} - \mathbf{F}(\boldsymbol{\beta})$ and $\boldsymbol{\beta}$ is an estimate of $\boldsymbol{\beta}^*$.

The second method is to solve the nonlinear "normal" equations

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

where

$$\mathbf{X} = \frac{\partial \mathbf{F}}{\partial \boldsymbol{\beta}}$$

In the nonlinear situation, both $\mathbf{X}$ and $\mathbf{F}(\boldsymbol{\beta})$ are functions of $\boldsymbol{\beta}$ and a closed-form solution generally does not exist. Thus, PROC NLIN uses an iterative process: a

starting value for $\boldsymbol{\beta}$ is chosen and continually improved until the error sum of squares $\epsilon'\epsilon$ is minimized.

The iterative techniques that PROC NLIN uses are similar to a series of linear regressions involving the matrix $\mathbf{X}$ evaluated for the current values of $\boldsymbol{\beta}$ and $\mathbf{e} = \mathbf{Y} - \mathbf{F}(\boldsymbol{\beta})$, the residuals evaluated for the current values of $\boldsymbol{\beta}$.

The iterative process begins at some point $\boldsymbol{\beta}_0$. Then $\mathbf{X}$ and $\mathbf{Y}$ are used to compute a $\boldsymbol{\Delta}$ such that

$$\text{SSE}(\boldsymbol{\beta}_0 + k\boldsymbol{\Delta}) < \text{SSE}(\boldsymbol{\beta}_0)$$

The four methods differ in how $\boldsymbol{\Delta}$ is computed to change the vector of parameters.

$$\begin{aligned}
\text{Steepest descent: } \boldsymbol{\Delta} &= \mathbf{X}'\mathbf{e} \\
\text{Gauss-Newton: } \boldsymbol{\Delta} &= (\mathbf{X}'\mathbf{X})^{-}\mathbf{X}'\mathbf{e} \\
\text{Newton: } \boldsymbol{\Delta} &= (\mathbf{G}^{-})\mathbf{X}'\mathbf{e} \\
\text{Marquardt: } \boldsymbol{\Delta} &= (\mathbf{X}'\mathbf{X} + \lambda\text{diag}(\mathbf{X}'\mathbf{X}))^{-}\mathbf{X}'\mathbf{e}
\end{aligned}$$

The default method used to compute $(\mathbf{X}'\mathbf{X})^{-}$ is the sweep operator producing a reflexive generalized (g_2) inverse. In some cases it would be preferable to use a Moore-Penrose (g_4) inverse. If the G4 option is specified in the PROC NLIN statement, a g_4 inverse is used to calculate $\boldsymbol{\Delta}$ on each iteration.

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. 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.

Steepest Descent (Gradient)

The steepest descent method is based on the gradient of $\epsilon'\epsilon$:

$$\frac{1}{2}\frac{\partial L(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -\mathbf{X}\mathbf{Y} + \mathbf{X}\mathbf{F}(\boldsymbol{\beta}) = -\mathbf{X}'\mathbf{e}$$

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

If the automatic variables _WEIGHT_ and _RESID_ are used, then

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

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

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \alpha\boldsymbol{\Delta}$$

where the scalar α is chosen such that

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

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

Newton

The Newton method uses the second derivatives and solves the equation

$$\boldsymbol{\Delta} = \mathbf{G}^{-}\mathbf{X}'\mathbf{e}$$

where

$$\mathbf{G} = (\mathbf{X}'\mathbf{X}) + \sum_{i=1}^{n} H_i(\boldsymbol{\beta})\mathbf{e}_i$$

and $H_i(\boldsymbol{\beta})$ is the Hessian of $\mathbf{e}$:

$$[H_i]_{jk} = \left[\frac{\partial^2 \mathbf{e}_i}{\partial\boldsymbol{\beta}_j \partial\boldsymbol{\beta}_k}\right]_{jk}$$

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

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

is the direction, where

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

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.

Gauss-Newton

The Gauss-Newton method uses the Taylor series

$$\mathbf{F}(\boldsymbol{\beta}) = \mathbf{F}(\boldsymbol{\beta}_0) + \mathbf{X}(\boldsymbol{\beta} - \boldsymbol{\beta}_0) + \cdots$$

where $\mathbf{X} = \partial\mathbf{F}/\partial\boldsymbol{\beta}$ is evaluated at $\boldsymbol{\beta} = \boldsymbol{\beta}_0$.

Substituting the first two terms of this series into the normal equations

$$\begin{aligned}
\mathbf{X}'\mathbf{F}(\boldsymbol{\beta}) &= \mathbf{X}'\mathbf{Y} \\
\mathbf{X}'(\mathbf{F}(\boldsymbol{\beta}_0) + \mathbf{X}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)) &= \mathbf{X}'\mathbf{Y} \\
(\mathbf{X}'\mathbf{X})(\boldsymbol{\beta} - \boldsymbol{\beta}_0) &= \mathbf{X}'\mathbf{Y} - \mathbf{X}'\mathbf{F}(\boldsymbol{\beta}_0) \\
(\mathbf{X}'\mathbf{X})\boldsymbol{\Delta} &= \mathbf{X}'\mathbf{e}
\end{aligned}$$

and therefore

$$\boldsymbol{\Delta} = (\mathbf{X}'\mathbf{X})^{-}\mathbf{X}'\mathbf{e}$$

Caution: If $\mathbf{X}'\mathbf{X}$ is singular or becomes singular, PROC NLIN computes $\boldsymbol{\Delta}$ 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.

Marquardt

The Marquardt updating formula is as follows:

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

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 $\lambda = 10^{-3}$ to start and computes a $\boldsymbol{\Delta}$. If $\text{SSE}(\boldsymbol{\beta}_0 + \boldsymbol{\Delta}) < \text{SSE}(\boldsymbol{\beta}_0)$, then $\lambda = \lambda/10$ for the next iteration. Each time $\text{SSE}(\boldsymbol{\beta}_0 + \boldsymbol{\Delta}) > \text{SSE}(\boldsymbol{\beta}_0)$, 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 is useful when the parameter estimates are highly correlated or the objective function is not well approximated by a quadratic.

Secant Method (DUD)

The multivariate secant method is like the Gauss-Newton method, except that the derivatives are estimated from the history of iterations rather than supplied analytically. The method is also called the *method of false position* or the Doesn't Use Derivatives (DUD) method (Ralston and Jennrich 1978). If only one parameter is being estimated, the derivative for iteration $i+1$ can be estimated from the previous two iterations:

$$der_{i+1} = \frac{\hat{\mathbf{Y}}_i - \hat{\mathbf{Y}}_{i-1}}{b_i - b_{i-1}}$$

When k parameters are to be estimated, the method uses the last k+1 iterations to estimate the derivatives.

Now that automatic analytic derivatives are available, DUD is not a recommended method but is retained for backward compatibility.

Step-Size Search

The default method of finding the step size k is step halving using SMETHOD=HALVE. If SSE($\boldsymbol{\beta}_0 + \boldsymbol{\Delta}$) $>$ SSE($\boldsymbol{\beta}_0$), compute SSE($\boldsymbol{\beta}_0 + 0.5\boldsymbol{\Delta}$), SSE($\boldsymbol{\beta}_0 + 0.25\boldsymbol{\Delta}$), . . . , 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 SSE($\boldsymbol{\beta}_0 + \boldsymbol{\Delta}$). 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 on the GOLDEN search, refer to 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 all the variables in 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:

$$\hat{\beta}_i \pm stderr_i * t(N-P, 0.05/2)$$

where $stderr_i$ is the standard error of the i^{th} parameter $\hat{\beta}_i$ and $t(N-P, 0.05/2)$ 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.

Model Confidence Intervals

Model confidence intervals are output when an OUT= data set is specified and one or more of the options L95M=, L95=, U95M=, or U95= is specified. The values of these terms are

$$\begin{aligned}
H &= w_i x_i (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1} x_i' \\
L95M &= f(\beta, \mathbf{z}_i) - \sqrt{MSE * H} * t(N-P, 0.95/2) \\
U95M &= f(\beta, \mathbf{z}_i) + \sqrt{MSE * H} * t(N-P, 0.95/2) \\
L95 &= f(\beta, \mathbf{z}_i) - \sqrt{MSE(H + 1/w_i)} * t(N-P, 0.95/2) \\
U95 &= f(\beta, \mathbf{z}_i) + \sqrt{MSE(H + 1/w_i)} * t(N-P, 0.95/2)
\end{aligned}$$

where $\mathbf{X} = \partial f / \partial \boldsymbol{\beta}$ and x_i is the ith row of $\mathbf{X}$. These results are derived for linear systems. The intervals are approximate for nonlinear models.

Parameter Covariance Matrix

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

$$(X'X)^{-1} * mse$$

for the gradient, Marquardt, and Gauss methods and

$$H^{-1} * mse$$

for Newton method. The mse is computed as

$$r'r/(nused - np)$$

where $nused$ is the number of non-missing observations and np is the number of estimable parameters. The standard error reported for the parameters is the sqrt of the corresponding diagonal element of this matrix.

Equality restrictions can be written as a vector function

$$h(\theta) = 0$$

Inequality restrictions are either active or inactive. When an inequality restriction is active, it is treated as an equality restriction.

For the following, assume the vector $h(\theta)$ contains all the current active restrictions. The constraint matrix A is

$$A(\hat{\theta}) = \frac{\partial h(\hat{\theta})}{\partial \hat{\theta}}$$

The covariance matrix for the restricted parameter estimates is computed as

$$Z(Z'HZ)^{-1}Z'$$

where H is Hessian or approximation to the Hessian, and Z is the last $(np - nc)$ columns of Q. Q is from an LQ factorization of the constraint matrix, *nc* is the number of active constraints, and *np* is the number of parameters. Refer to Gill, Murray, and Wright (1981) for more details on LQ factorization.

The covariance matrix for the Lagrange multipliers is computed as

$$(AH^{-1}A')^{-1}$$

Reported Convergence Measures

NLIN 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 Jacobian columns, and it approaches 0 as the gradient of the objective function becomes small. R is defined as

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

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 *k*th iteration, PPC is the maximum over the parameters

$$\frac{|\theta_i^{k+1} - \theta_i^k|}{|\theta|_i^k + 1E-6}$$

where θ_i^k is the current value of the *i*th parameter and θ_i^{k+1} is the prospective value of this parameter after adding the change vector computed for the next iteration. 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 *k*th iteration, RPC is the maximum over *i* of

$$\frac{|\theta_i^k - \theta_i^{k-1}|}{|\theta_i^{k-1} + 1E-6|}$$

where θ_i^k is the current value of the *i*th parameter and θ_i^{k-1} is the previous value of this parameter. 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:

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

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 items:

- the estimates of the parameters and the residual Sums of Squares determined in each iteration
- a list of the residual Sums of Squares associated with all or some of the combinations of possible starting values of parameters
- an analysis-of-variance table including as sources of variation Regression, Residual, Uncorrected Total, Corrected Total, and F test

If the convergence criterion is met, PROC NLIN produces

- Estimation Summary Table
- Parameter Estimates
- an asymptotically valid standard error of the estimate, Asymptotic Standard Error.
- an Asymptotic 95% Confidence Interval for the estimate of the parameter
- an Asymptotic Correlation Matrix of the parameters

Incompatibilities with 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 any 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, for example, 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 SAS 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. (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.

```
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 the following table. For more information on ODS, see Chapter 15, "Using the Output Delivery System."

Table 45.1. ODS Tables Produced in 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
DerList	Derivative variables	LISTDER
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
SecondDerivatives	Second derivative table	LISTDER

Examples

Example 45.1. Segmented Model

From theoretical considerations, you can hypothesize that

$$\begin{aligned} y &= a + b\,x + c\,x^2 \qquad \text{if } x < x_0 \\ y &= p \qquad\qquad\qquad\quad \text{if } x >= x_0 \end{aligned}$$

That is, for values of x less than x_0, the equation relating y and x is quadratic (a parabola); and, for values of x greater than x_0, the equation is constant (a horizontal line). PROC NLIN can fit such a segmented model even when the joint point, x_0, is unknown.

The curve must be continuous (the two sections must meet at x_0), and the curve must be smooth (the first derivatives with respect to x are the same at x_0).

These conditions imply that

$$\begin{aligned} x_0 &= -b/2c \\ p &= a - b^2/4c \end{aligned}$$

The segmented equation includes only three parameters; however, the equation is nonlinear with respect to these parameters.

You can write program statements with PROC NLIN to conditionally execute different sections of code for the two parts of the model, depending on whether x is less than x_0 .

A PUT statement is used to print the constrained parameters every time the program is executed for the first observation (where x =1). The following statements perform the analysis.

```
*---------FITTING A SEGMENTED MODEL USING NLIN-----*
|      |                                            |
|   Y  | QUADRATIC          PLATEAU                 |
|      | Y=A+B*X+C*X*X      Y=P                     |
|      |                     ..................     |
|      |                .    :                      |
|      |           .         :                      |
|      |       .             :                      |
|      |     .               :                      |
|      |   .                 :                      |
|      +-----------------------------------------X  |
|                          X0                       |
|                                                   |
| CONTINUITY RESTRICTION: P=A+B*X0+C*X0**2          |
| SMOOTHNESS RESTRICTION: 0=B+2*C*X0 SO X0=-B/(2*C)|
*---------------------------------------------------*;

title 'Quadratic Model with Plateau';
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
;
proc nlin;
   parms a=.45 b=.05 c=-.0025;

   x0=-.5*b / c;                * Estimate join point;
   if x<x0 then                 * Quadratic part of Model;
      model y=a+b*x+c*x*x;
   else                         * Plateau part of Model;
      model y=a+b*x0+c*x0*x0;

   if _obs_=1 and _iter_ =.  then do;
      plateau=a+b*x0+c*x0*x0;
      put /  x0= plateau=  ;
      end;
   output out=b predicted=yp;
run;

/* Setup for creating the graph */
legend1 frame cframe=ligr label=none cborder=black
        position=center value=(justify=center);
axis1 label=(angle=90 rotate=0) minor=none;
axis2 minor=none;

proc gplot;
   plot y*x yp*x/frame cframe=ligr legend=legend1
   vaxis=axis1 haxis=axis2 overlay ;
run;
```

Output 45.1.1. Nonlinear Least Squares Iterative Phase

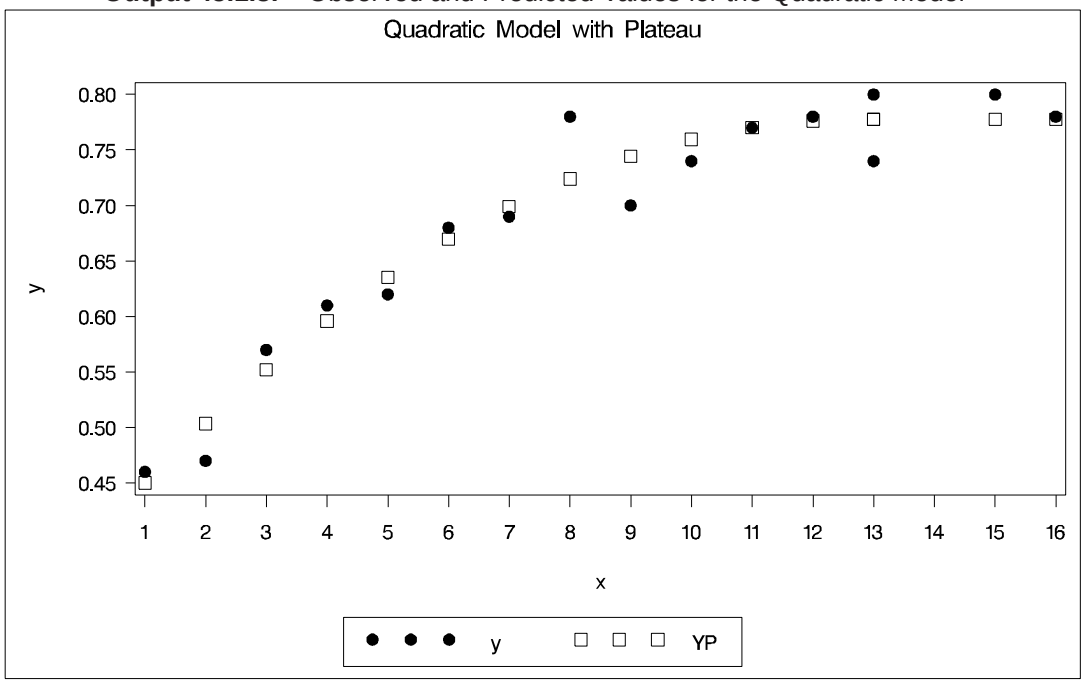
```
                    Quadratic Model with Plateau

                         The NLIN Procedure
                          Iterative Phase
                        Dependent Variable y
                        Method: Gauss-Newton

                                                        Sum of
          Iter           a           b           c     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.
x0=12.747669162 plateau=0.7774974276
```

Output 45.1.2. Least Squares Analysis for the Quadratic Model

```
                          The NLIN Procedure

x0=12.747669162 plateau=0.7774974276 Sum of        Mean                Approx
   Source                     DF     Squares      Square    F Value    Pr > F

   Regression                  3      7.7256      2.5752     114.22    <.0001
   Residual                   13      0.0101    0.000774
   Uncorrected Total          16      7.7357

   Corrected Total            15      0.1869


                                     Approx      Approximate 95% Confidence
         Parameter      Estimate    Std Error                Limits

         a                0.3921       0.0267      0.3345      0.4497
         b                0.0605      0.00842      0.0423      0.0787
         c               -0.00237     0.000551    -0.00356    -0.00118


                      Approximate Correlation Matrix
                             a              b              c

              a       1.0000000      -0.9020250       0.8124327
              b      -0.9020250       1.0000000      -0.9787952
              c       0.8124327      -0.9787952       1.0000000
x0=12.747669162 plateau=0.7774974276
```

Output 45.1.1 indicates that the join point is 12.75 and the plateau value is 0.78. As displayed in the following plot of the predicted values (YP) and the actual values, the selected join point and plateau value is reasonable. The predicted values for the estimation are written to the data set b with the OUTPUT statement.

Output 45.1.3. Observed and Predicted Values for the Quadratic Model



Example 45.2. Iteratively Reweighted Least Squares

The NLIN procedure is suited to methods that make the weight a function of the parameters in each iteration since the _WEIGHT_ variable can be computed with program statements.

The NOHALVE option is used because the SSE definition is modified at each iteration and the step-shortening criteria is thus circumvented.

Iteratively reweighted least squares (IRLS) can produce estimates for many of the robust regression criteria suggested in the literature. These methods act like automatic outlier rejectors since large residual values lead to very small weights. Holland and Welsch (1977) outline several of these robust methods. For example, the biweight criterion suggested by Beaton and Tukey (1974) tries to minimize

$$S_{biweight} = \Sigma\rho(r)$$

where

$$\rho(r) = (B^2/2)(1-(1-(r/B)^2)^2) \quad \text{if } |r| \leq B$$

or

$$\rho(r) = (B^2/2) \qquad \text{otherwise}$$

where r is $|residual|/\sigma$, σ is a measure of scale of the error, and B is a tuning constant.

The weighting function for the biweight is

$$w_i = (1 - (r_i/B)^2)^2 \qquad \text{if } |r_i| \leq B$$

or

$$w_i = 0 \qquad \text{if } |r_i| > B$$

The biweight estimator depends on both a measure of scale (like the standard deviation) and a tuning constant; results vary if these values are changed.

The data are the population of the United States (in millions), recorded at ten-year intervals starting in 1790 and ending in 1990.

```
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
;

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;
   b=4.685;
   r=abs(resid / sigma);
   if r<=b then _weight_=(1-(r / b)**2)**2;
   else _weight_=0;
   output out=c r=rbi;
run;

data c;
set c;
   sigma=2;
   b=4.685;
   r=abs(rbi / sigma);
   if r<=b then _weight_=(1-(r / b)**2)**2;
   else _weight_=0;
proc print;
run;
```

Output 45.2.1. Nonlinear Least Squares Analysis

Beaton/Tukey Biweight Robust Regression using IRLS

The NLIN Procedure

Source	DF	Sum of Squares	Mean Square	F Value	Approx Pr > F
Regression	3	232436	77478.8	49454.5	<.0001
Residual	18	20.6670	1.1482		
Uncorrected Total	21	232457			
Corrected Total	20	113585			

Parameter	Estimate	Approx Std Error	Approximate 95% Confidence 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 45.2.2. Listing of Computed Weights from PROC NLIN

Beaton/Tukey Biweight Robust Regression using IRLS

Obs	pop	year	yearsq	RBI	sigma	b	r	_weight_
1	3.929	1790	3204100	-0.93711	2	4.685	0.46855	0.98010
2	5.308	1800	3240000	0.46091	2	4.685	0.23045	0.99517
3	7.239	1810	3276100	1.11853	2	4.685	0.55926	0.97170
4	9.638	1820	3312400	0.95176	2	4.685	0.47588	0.97947
5	12.866	1830	3348900	0.32159	2	4.685	0.16080	0.99765
6	17.069	1840	3385600	-0.62597	2	4.685	0.31298	0.99109
7	23.191	1850	3422500	-0.94692	2	4.685	0.47346	0.97968
8	31.443	1860	3459600	-0.43027	2	4.685	0.21514	0.99579
9	39.818	1870	3496900	-1.08302	2	4.685	0.54151	0.97346
10	50.155	1880	3534400	-1.06615	2	4.685	0.53308	0.97427
11	62.947	1890	3572100	0.11332	2	4.685	0.05666	0.99971
12	75.994	1900	3610000	0.25539	2	4.685	0.12770	0.99851
13	91.972	1910	3648100	2.03607	2	4.685	1.01804	0.90779
14	105.710	1920	3686400	0.28436	2	4.685	0.14218	0.99816
15	122.775	1930	3724900	0.56725	2	4.685	0.28363	0.99268
16	131.669	1940	3763600	-8.61325	2	4.685	4.30662	0.02403
17	151.325	1950	3802500	-8.32415	2	4.685	4.16207	0.04443
18	179.323	1960	3841600	-0.98543	2	4.685	0.49272	0.97800
19	203.211	1970	3880900	0.95088	2	4.685	0.47544	0.97951
20	226.542	1980	3920400	1.03780	2	4.685	0.51890	0.97562
21	248.710	1990	3960100	-1.33067	2	4.685	0.66533	0.96007

Output 45.2.2 displays the computed weights. The observations for 1940 and 1950 are highly discounted because of their large residuals.

Example 45.3. Probit Model with Likelihood function

The data, taken from Lee (1974), consist of patient characteristics and a variable indicating whether cancer remission occured. This example demonstrates how to use PROC NLIN with a likelihood function. In this case, the likelihood function to minimize is

$$-2\log L \quad = \quad -2\sum_{i=1}^{N} wght_i \log(\hat{p}_i(y_i, \mathbf{x}_i))$$

where

$$\hat{p}_i(y_i, \mathbf{x}_i) = \begin{cases} \Phi(\alpha + \beta'\mathbf{x}_i) & y_i = 0 \\ 1 - \Phi(\alpha + \beta' x_i) & y_i = 1 \end{cases}$$

and Φ is the normal probability function. This is the likelihood function for a binary probit model. This likelihood is strictly positive so that you can take a square root of $\log(\hat{p}_i(y_i, \mathbf{x}_i))$ and use this as your residual in PROC NLIN. The DATA step also creates a zero-valued dummy variable, like, that is used as the dependent variable.

```
Data remiss;
   input remiss cell smear infil li blast temp;
   label remiss = 'complete remission';
   like = 0;
   label like = 'dummy variable for nlin';
   datalines;
1 .8 .83 .66 1.9 1.1 .996
1 .9 .36 .32 1.4 .74 .992
0 .8 .88 .7 .8 .176 .982
0 1 .87 .87 .7 1.053 .986
1 .9 .75 .68 1.3 .519 .98
0 1 .65 .65 .6 .519 .982
1 .95 .97 .92 1 1.23 .992
0 .95 .87 .83 1.9 1.354 1.02
0 1 .45 .45 .8 .322 .999
0 .95 .36 .34 .5 0 1.038
0 .85 .39 .33 .7 .279 .988
0 .7 .76 .53 1.2 .146 .982
0 .8 .46 .37 .4 .38 1.006
0 .2 .39 .08 .8 .114 .99
0 1 .9 .9 1.1 1.037 .99
1 1 .84 .84 1.9 2.064 1.02
0 .65 .42 .27 .5 .114 1.014
0 1 .75 .75 1 1.322 1.004
0 .5 .44 .22 .6 .114 .99
1 1 .63 .63 1.1 1.072 .986
0 1 .33 .33 .4 .176 1.01 0
0 .9 .93 .84 .6 1.591 1.02
```

```
1 1 .58 .58 1 .531 1.002
0 .95 .32 .3 1.6 .886 .988
1 1 .6 .6 1.7 .964 .99
1 1 .69 .69 .9 .398 .986
0 1 .73 .73 .7 .398 .986
;
run;

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

        /* Linear portion of model ------*/
   eq1 = a*cell + b*li + c*temp +int;

          /* probit */
   p = probnorm(eq1);

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

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

Note that the asymptotic standard errors of the parameters are computed under the least squares assumptions. The SIGSQ=1 option on the PROC NLIN statement forces PROC NLIN to replace the usual mean square error with 1. Also, METHOD=NEWTON is selected so the true Hessian of the likelihood function is used to calculate parameter standard errors rather than the crossproducts approximation to the Hessian.

Output 45.3.1. Nonlinear Least Squares Analysis from PROC NLIN

Beaton/Tukey Biweight Robust Regression using IRLS

The NLIN Procedure

NOTE: An intercept was not specified for this model.

Source	DF	Sum of Squares	Mean Square	F Value	Approx Pr > F
Regression	4	-21.9002	-5.4750	-5.75	.
Residual	23	21.9002	0.9522		
Uncorrected Total	27	0			
Corrected Total	26	0			

Parameter	Estimate	Approx Std Error	Approximate 95% Confidence Limits	
a	-5.6298	4.6376	-15.2234	3.9638
b	-2.2513	0.9790	-4.2764	-0.2262
c	45.1815	34.9095	-27.0337	117.4
int	-36.7548	32.3607	-103.7	30.1879

The problem can be more simply solved using the following SAS statements.

```
proc probit data=remiss ;
   class remiss;
   model remiss=cell li temp ;
run;
```

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, Inc.

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 and 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, Philip E., Murray, Walter, and Wright, Margaret H. (1981), "Practical Optimization," New York: Academic Press Inc.

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.

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, Tsoung-Chao (1980), *The Theory and Practice of Econometrics*, New York: John Wiley & Sons, Inc.

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

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.

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

Ralston, M.L. and Jennrich, R.I. (1978), "DUD, A Derivative-Free Algorithm for Nonlinear Least Squares," *Technometrics*, 20, 7–14.

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

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/STAT® User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS/STAT® User's Guide, Version 8

ISBN 1–58025–494–2

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

1st printing, October 1999