Chapter 60
The HPNLMOD Procedure

Overview: HPNLMOD Procedure
The HPNLMOD procedure is a high-performance procedure that uses either nonlinear least squares or maximum likelihood to fit nonlinear regression models. PROC HPNLMOD enables you to specify the model by using SAS programming statements, which give you greater flexibility in modeling the relationship
between the response variable and independent (regressor) variables than do SAS procedures that use a more structured MODEL statement.

PROC HPNLMOD runs in either single-machine mode or distributed mode.

**NOTE:** Distributed mode requires SAS High-Performance Statistics.

---

**PROC HPNLMOD Features**

The HPNLMOD procedure does the following:

- reads input data in parallel and writes output data in parallel when the data source is the appliance database
- is highly multithreaded during all phases of analytic execution
- computes analytical derivatives of user-provided expressions for more robust parameter estimations
- evaluates user-provided expressions and their confidence limits by using the ESTIMATE and PREDICT statements
- estimates parameters without specifying a particular distribution function by using the least squares method
- estimates parameters by using the maximum likelihood method when either a built-in distribution function is specified or a likelihood function is provided

Because the HPNLMOD procedure is a high-performance analytical procedure, it also does the following:

- enables you to run in distributed mode on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on the server where SAS is installed
- exploits all the available cores and concurrent threads, regardless of execution mode

For more information, see the section “Processing Modes” (Chapter 2, *SAS/STAT User’s Guide: High-Performance Procedures*).

---

**PROC HPNLMOD Contrasted with the NLIN and NLMIXED Procedures**

Like the NLIN procedure, the HPNLMOD procedure estimates parameters by using least squares minimization for models that are specified by SAS programming statements. However, PROC HPNLMOD can also perform maximum likelihood estimation when information about the response variable’s distribution is provided. PROC HPNLMOD also has a RESTRICT statement for specifying restrictions on parameter estimates that are more general than those that are available in PROC NLIN. Because the HPNLMOD and
NLIN procedures use different optimization techniques, the available options that control the estimation process and resulting parameter estimates can differ between these procedures when equivalent models and data are analyzed.

Although it does not support the specification of random effects, PROC HPNLMOD is similar to PROC NLMIXED. Both procedures perform maximum likelihood estimation by using the same programming syntax and set of distributions to specify the model’s mean term. In addition, both PROC HPNLMOD and PROC NLMIXED use the same optimization techniques and options. However, PROC NLMIXED does not support least squares parameter estimation.

### Getting Started: HPNLMOD Procedure

The most common use of the HPNLMOD procedure is to estimate the parameters in a model in which the response variable is a nonlinear function of one or more of the parameters.

### Least Squares Model

The Michaelis-Menten model of enzyme kinetics (Ratkowsky 1990, p. 59) relates a substrate’s concentration to its catalyzed reaction rate. The Michaelis-Menten model can be analyzed using a least squares estimation because it does not specify how the reaction rate is distributed around its predicted value. The relationship between reaction rate and substrate concentration is

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

where \(x_i\) represents the concentration for \(n\) trials and \(f(x, \theta)\) is the reaction rate. The vector \(\theta\) contains the rate parameters.

For this model, which has experimental measurements of reaction rate and concentration stored in the enzyme data set, the following SAS statements estimate the parameters \(\theta_1\) and \(\theta_2\):

```sas
proc hpnlmod data=enzyme;
   parms theta1=0 theta2=0;
   model rate ~ residual(theta1*conc / (theta2 + conc));
run;
```

The least squares estimation performed by PROC HPNLMOD for this enzyme kinetics problem produces the analysis of variance table that is displayed in Figure 60.1. The table displays the degrees of freedom, sums of squares, and mean squares along with the model \(F\) test.
Figure 60.1: Nonlinear Least Squares Analysis of Variance

The HPNLMOD Procedure

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Approx Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>2</td>
<td>290116</td>
<td>145058</td>
<td>88537.2</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>12</td>
<td>19.6606</td>
<td>1.6384</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uncorrected Total</td>
<td>14</td>
<td>290135</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

An intercept was not specified for this model.

Finally, Figure 60.2 displays the parameter estimates, standard errors, t statistics, and 95% confidence intervals for \( \theta_1 \) and \( \theta_2 \).

Figure 60.2: Parameter Estimates and Approximate 95% Confidence Intervals

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>DF</th>
<th>t Value</th>
<th>Approx Pr &gt;</th>
<th>Approx 95% Confidence Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \theta_1 )</td>
<td>158.1</td>
<td>0.6737</td>
<td>1</td>
<td>234.67</td>
<td>&lt;.0001</td>
<td>156.6 159.6</td>
</tr>
<tr>
<td>( \theta_2 )</td>
<td>0.0741</td>
<td>0.00313</td>
<td>1</td>
<td>23.69</td>
<td>&lt;.0001</td>
<td>0.0673 0.0809</td>
</tr>
</tbody>
</table>

In the enzyme kinetics model, no information was supplied about the distribution of the reaction rate around the model’s mean value. Therefore, the residual model distribution was specified to perform a least squares parameter fit.

Binomial Model

In Example 85.3 cancer remission is modeled by expressing the maximum likelihood function for a binary distribution as a nonlinear least squares optimization. The following statements show an equivalent formulation of this model that uses PROC HPNLMOD and specifies the binary distribution explicitly:

```
proc hpnlmod data=remiss corr;
  parms int=-10 a=-2 b=-1 c=6;
  linp = int + a*cell + b*li + c*temp;
  p = probnorm(linp);
  model remiss ~ binary(1-p);
run;
```

This binary distribution model displays information about the quality of the estimation that is different from the information displayed in the section “Least Squares Model” on page 4797. No analysis of variance table is produced for this model; fit statistics that are based on the value of the likelihood function are displayed in Figure 60.3.
Figure 60.3  Nonlinear Likelihood Function Statistics

The HPNLMOD Procedure

<table>
<thead>
<tr>
<th>Fit Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2 Log Likelihood</td>
</tr>
<tr>
<td>AIC (smaller is better)</td>
</tr>
<tr>
<td>AICC (smaller is better)</td>
</tr>
<tr>
<td>BIC (smaller is better)</td>
</tr>
</tbody>
</table>

Parameter estimates for the binary distribution model that uses the same quantities as are used in the section “Least Squares Model” on page 4797 are displayed in Figure 60.4.

Figure 60.4  Parameter Estimates and Approximate 95% Confidence Intervals

<table>
<thead>
<tr>
<th>Parameter Estimates</th>
<th>Approximate 95% Confidence Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Estimate</td>
</tr>
<tr>
<td>int</td>
<td>-36.7548</td>
</tr>
<tr>
<td>a</td>
<td>-5.6298</td>
</tr>
<tr>
<td>b</td>
<td>-2.2513</td>
</tr>
<tr>
<td>c</td>
<td>45.1815</td>
</tr>
</tbody>
</table>

Syntax: HPNLMOD Procedure

The following statements are available in the HPNLMOD procedure:

```
PROC HPNLMOD <options> ;
    BOUNDS constraint < , ... , constraint> ;
    BY variables ;
    ESTIMATE 'label' expression <options> ;
    MODEL dependent-variable ~ distribution ;
    PARAMETERS <parameter-specification> < , ... , parameter-specification> </options> ;
    PERFORMANCE <performance-options> ;
    PREDICT 'label' expression <options> ;
    RESTRICT restriction1 < , restriction2 ... > ;
    Programming Statements ;
```

The PROC HPNLMOD statement and exactly one MODEL statement are required.
The PROC HPNLMOD statement invokes the procedure. Table 60.1 summarizes important options in the PROC HPNLMOD statement by function. These and other options in the PROC HPNLMOD statement are then described fully in alphabetical order.

**Table 60.1 PROC HPNLMOD Statement Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic Options</strong></td>
<td></td>
</tr>
<tr>
<td>DATA=</td>
<td>Specifies the input data set</td>
</tr>
<tr>
<td>OUT=</td>
<td>Specifies the output data set</td>
</tr>
<tr>
<td><strong>Output Options</strong></td>
<td></td>
</tr>
<tr>
<td>CORR</td>
<td>Specifies the correlation matrix</td>
</tr>
<tr>
<td>COV</td>
<td>Specifies the covariance matrix</td>
</tr>
<tr>
<td>ECORR</td>
<td>Specifies the correlation matrix of additional estimates</td>
</tr>
<tr>
<td>ECOV</td>
<td>Specifies the covariance matrix of additional estimates</td>
</tr>
<tr>
<td>DF</td>
<td>Specifies the default degrees of freedom</td>
</tr>
<tr>
<td>NOPRINT</td>
<td>Suppresses ODS output</td>
</tr>
<tr>
<td>NOITPRINT</td>
<td>Suppresses output about iterations within the optimization process</td>
</tr>
<tr>
<td><strong>Optimization Options</strong></td>
<td></td>
</tr>
<tr>
<td>ABSCONV=</td>
<td>Tunes an absolute function convergence criterion</td>
</tr>
<tr>
<td>ABSFCONV=</td>
<td>Tunes an absolute difference function convergence criterion</td>
</tr>
<tr>
<td>ABSGCONV=</td>
<td>Tunes the absolute gradient convergence criterion</td>
</tr>
<tr>
<td>FCONV=</td>
<td>Tunes the relative function convergence criterion</td>
</tr>
<tr>
<td>GCONV=</td>
<td>Tunes the relative gradient convergence criterion</td>
</tr>
<tr>
<td>MAXITER=</td>
<td>Chooses the maximum number of iterations in any optimization</td>
</tr>
<tr>
<td>MAXFUNC=</td>
<td>Specifies the maximum number of function evaluations in any optimization</td>
</tr>
<tr>
<td>MAXTIME=</td>
<td>Specifies the upper limit seconds of CPU time for any optimization</td>
</tr>
<tr>
<td>MINITER=</td>
<td>Specifies the minimum number of iterations in any optimization</td>
</tr>
<tr>
<td>TECHNIQUE=</td>
<td>Selects the optimization technique</td>
</tr>
<tr>
<td><strong>Tolerance Options</strong></td>
<td></td>
</tr>
<tr>
<td>SINGULAR=</td>
<td>Tunes the general singularity criterion</td>
</tr>
<tr>
<td><strong>User-Defined Format Options</strong></td>
<td></td>
</tr>
<tr>
<td>FMTLIBXML=</td>
<td>Specifies a file reference for a format stream</td>
</tr>
<tr>
<td>XMLFORMAT=</td>
<td>Specifies a file name for a format stream</td>
</tr>
</tbody>
</table>
You can specify the following *options* in the PROC HPNLMOD statement.

**ABSCONV=**r

**ABSTOL=**r

specifies an absolute function convergence criterion. For minimization, termination requires \( f(\psi^{(k)}) \leq r \), where \( \psi \) is the vector of parameters in the optimization and \( f(\cdot) \) is the objective function. The default value of \( r \) is the negative square root of the largest double-precision value, which serves only as a protection against overflow.

**ABSFCONV=**r < n>

**ABSFTOL=**r < n>

specifies an absolute difference function convergence criterion. For all techniques except the Nelder-Mead simplex (NMSIMP) technique, termination requires a small change of the function value in successive iterations:

\[
| f(\psi^{(k-1)}) - f(\psi^{(k)}) | \leq r
\]

Here, \( \psi \) denotes the vector of parameters that participate in the optimization, and \( f(\cdot) \) is the objective function. The same formula is used for the NMSIMP technique, but \( \psi^{(k)} \) is defined as the vertex that has the lowest function value, and \( \psi^{(k-1)} \) is defined as the vertex that has the highest function value in the simplex. The default value is \( r = 0 \). The optional integer value \( n \) specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

**ABSGCONV=**r < n>

**ABSGTOL=**r < n>

specifies an absolute gradient convergence criterion. Termination requires the maximum absolute gradient element to be small:

\[
\max_j | g_j(\psi^{(k)}) | \leq r
\]

Here, \( \psi \) denotes the vector of parameters that participate in the optimization, and \( g_j(\cdot) \) is the gradient of the objective function with respect to the \( j \)th parameter. This criterion is not used by the NMSIMP technique. The default value is \( r = 1E-5 \). The optional integer value \( n \) specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

**ALPHA=**\( \alpha \)

specifies the level of significance \( \alpha \) that is 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 that are computed in the “Parameter Estimates” table and is used in the LOWER and UPPER options in the PREDICT statement.

**CORR**

requests the approximate correlation matrix for the parameter estimates.

**COV**

requests the approximate covariance matrix for the parameter estimates.
DATA=SAS-data-set

names the SAS data set to be used by PROC HPNLMOD. The default is the most recently created data set.

If PROC HPNLMOD executes in distributed mode, the input data are distributed to memory on the appliance nodes and analyzed in parallel, unless the data are already distributed in the appliance database. In the latter case, PROC HPNLMOD reads the data alongside the distributed database. For more information about the various execution modes, see the section “Processing Modes” (Chapter 2, SAS/STAT User’s Guide: High-Performance Procedures); for more information about the alongside-the-database model, see the section “Alongside-the-Database Execution” (Chapter 2, SAS/STAT User’s Guide: High-Performance Procedures).

DF=n

specifies the default number of degrees of freedom to use in the calculation of p-values and confidence limits for additional parameter estimates.

ECORR

requests the approximate correlation matrix for all expressions that are specified in ESTIMATE statements.

ECOV

requests the approximate covariance matrix for all expressions that are specified in ESTIMATE statements.

FCONV=r<n>

specifies a relative function convergence criterion. For all techniques except NMSIMP, termination requires a small relative change of the function value in successive iterations:

$$\frac{|f(\psi^{(k)}) - f(\psi^{(k-1)})|}{|f(\psi^{(k-1)})|} \leq r$$

Here, $\psi$ denotes the vector of parameters that participate in the optimization, and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\psi^{(k)}$ is defined as the vertex that has the lowest function value, and $\psi^{(k-1)}$ is defined as the vertex that has the highest function value in the simplex. The default is $r = 10^{-\text{FDIGITS}}$, where FDIGITS is by default $-\log_{10}\{\epsilon\}$ and $\epsilon$ is the machine precision. The optional integer value $n$ specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

FMTLIBXML=file-ref

specifies the file reference for the XML stream that contains the user-defined format definitions. User-defined formats are handled differently in a distributed computing environment than they are handled in other SAS products. For information about how to generate a XML stream for your formats, see the section “Working with Formats” (Chapter 2, SAS/STAT User’s Guide: High-Performance Procedures).

GCONV=r<n>

specifies a relative gradient convergence criterion. For all techniques except the conjugate gradient (CONGRA) and NMSIMP techniques, termination requires that the normalized predicted function
reduction be small:

\[
\frac{g(\psi^{(k)})' [H^{(k)}]^{-1} g(\psi^{(k)})}{|f(\psi^{(k)})|} \leq r
\]

Here, \(\psi\) denotes the vector of parameters that participate in the optimization, \(f(\cdot)\) is the objective function, and \(g(\cdot)\) is the gradient. For the CONGRA technique (where a reliable Hessian estimate \(H\) is not available), the following criterion is used:

\[
\frac{\|g(\psi^{(k)})\|^2}{\|g(\psi^{(k)}) - g(\psi^{(k-1)})\|^2 |f(\psi^{(k)})|} \leq r
\]

This criterion is not used by the NMSIMP technique. The default value is \(r = 1E-8\). The optional integer value \(n\) specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

**MAXFUNC=**\(n\)

**MAXFU=**\(n\)

specifies the maximum number of function calls in the optimization process. The default values are as follows, depending on the optimization technique (which you specify in the **TECHNIQUE=** option):

- **TRUREG**, **NRRIDG**, **NEWRAP**: \(n = 125\)
- **QUANEW**, **DBLDOG**: \(n = 500\)
- **CONGRA**: \(n = 1,000\)
- **NMSIMP**: \(n = 3,000\)

Optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed \(n\).

**MAXITER=**\(n\)

**MAXIT=**\(n\)

specifies the maximum number of iterations in the optimization process. The default values are as follows, depending on the optimization technique (which you specify in the **TECHNIQUE=** option):

- **TRUREG**, **NRRIDG**, **NEWRAP**: \(n = 50\)
- **QUANEW**, **DBLDOG**: \(n = 200\)
- **CONGRA**: \(n = 400\)
- **NMSIMP**: \(n = 1,000\)

These default values also apply when \(n\) is specified as a missing value.

**MAXTIME=**\(r\)

specifies an upper limit of \(r\) seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. This time that is specified by \(r\) is checked only once at the end of each iteration. Therefore, the actual running time can be longer than \(r\).
MINITER=n
MINIT=n
specifies the minimum number of iterations. The default value is 0. If you request more iterations than are actually needed for convergence to a stationary point, the optimization algorithms can behave strangely. For example, the effect of rounding errors can prevent the algorithm from continuing for the required number of iterations.

NOITPRINT
suppresses the display of the “Iteration History” table.

NOPRINT
suppresses the generation of ODS output.

OUT=SAS-data-set
names the SAS data set to be created when one or more PREDICT statements are specified. A single OUT= data set is created to contain all predicted values when more than one PREDICT statement is specified. An error message is produced if a PREDICT statement is specified and an OUT= data set is not specified. For more information about output data sets in SAS high-performance analytical procedures, see the section “Output Data Sets” (Chapter 2, SAS/STAT User’s Guide: High-Performance Procedures).

SINGULAR=number
tunes the general singularity criterion that is applied in sweeps and inversions. The default is 1E4 times the machine epsilon; this product is approximately 1E−12 on most computers.

TECHNIQUE=keyword
TECH=keyword
specifies the optimization technique for obtaining maximum likelihood estimates. You can choose from the following techniques by specifying the appropriate keyword:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONGRA</td>
<td>performs a conjugate-gradient optimization.</td>
</tr>
<tr>
<td>DBLDOG</td>
<td>performs a version of double-dogleg optimization.</td>
</tr>
<tr>
<td>LEVMAR</td>
<td>performs a Levenberg-Marquardt optimization.</td>
</tr>
<tr>
<td>NEWRAP</td>
<td>performs a Newton-Raphson optimization that combines a line-search algorithm with ridging.</td>
</tr>
<tr>
<td>NMSIMP</td>
<td>performs a Nelder-Mead simplex optimization.</td>
</tr>
<tr>
<td>NONE</td>
<td>performs no optimization.</td>
</tr>
<tr>
<td>NRRIDG</td>
<td>performs a Newton-Raphson optimization with ridging.</td>
</tr>
<tr>
<td>QUANEW</td>
<td>performs a quasi-Newton optimization.</td>
</tr>
<tr>
<td>TRUREG</td>
<td>performs a trust-region optimization.</td>
</tr>
</tbody>
</table>

The default value is TECHNIQUE=LEVMAR for least squares regression models and TECHNIQUE=NRRIDG for models where the distribution is specified.
**XMLFORMAT=filename**

specifies the file name for the XML stream that contains the user-defined format definitions. User-defined formats are handled differently in a distributed computing environment than they are handled in other SAS products. For information about how to generate a XML stream for your formats, see the section “Working with Formats” (Chapter 2, **SAS/STAT User’s Guide: High-Performance Procedures**).

---

**BOUNDS Statement**

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

where **constraint** represents

< number operator > parameter-list < operator number >

Boundary constraints are specified in a BOUNDS statement. One- or two-sided boundary constraints are allowed. Elements in a list of boundary constraints are separated by commas. For example:

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

You can specify more than one BOUNDS statement. If you specify more than one lower (or upper) bound for the same parameter, the maximum (or minimum) of these is taken.

If the maximum $l_j$ of all lower bounds is larger than the minimum of all upper bounds $u_j$ for the same parameter $\theta_j$, the boundary constraint is replaced by $\theta_j := l_j := \min(u_j)$, which is defined by the minimum of all upper bounds specified for $\theta_j$.

---

**BY Statement**

**BY variables ;**

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

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

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

**BY statement processing is not supported when the HPNLMOD procedure runs alongside the database or alongside the Hadoop distributed file system (HDFS). These modes are used if the input data are stored in a database or HDFS and the grid host is the appliance that houses the data.**
For more information about BY-group processing, see the discussion in SAS Language Reference: Concepts. For more information about the DATASETS procedure, see the discussion in the Base SAS Procedures Guide.

**ESTIMATE Statement**

```
ESTIMATE 'label' expression < options > ;
```

The ESTIMATE statement enables you to compute an additional estimate that is a function of the parameter values. You must provide a quoted string to identify the estimate and then provide a valid SAS expression. Multiple ESTIMATE statements are permitted, and results from all ESTIMATE statements are listed in a common table. PROC HPNLMOD computes approximate standard errors for the estimates by using the delta method (Billingsley 1986). It uses these standard errors to compute corresponding $t$ statistics, $p$-values, and confidence limits.

The ECOV option in the PROC HPNLMOD statement produces a table that contains the approximate covariance matrix of all the additional estimates you specify. The ECORR option produces the corresponding correlation matrix.

You can specify the following options in the ESTIMATE statement:

- **ALPHA=** $\alpha$
  - specifies the alpha level to be used to compute confidence limits. The default value corresponds to the ALPHA= option in the PROC HPNLMOD statement.

- **DF=** $d$
  - specifies the degrees of freedom to be used to compute $p$-values and confidence limits. The default value corresponds to the DF= option in the PROC HPNLMOD statement.

**MODEL Statement**

```
MODEL dependent-variable ~ distribution ;
```

The MODEL statement is the mechanism for either using a distribution specification to specify the distribution of the data or using the RESIDUAL distribution to specify a predicted value. You must specify a single dependent variable from the input data set, a tilde ($\sim$), and then a distribution along with its parameters. You can specify the following values for distribution:

- **RESIDUAL**$(m)$ or **LS**$(m)$
  - specifies no particular distribution. Instead the sum of squares of the differences between $m$ and the dependent variable is minimized.

- **NORMAL**$(m, \nu)$
  - specifies a normal (Gaussian) distribution that has mean $m$ and variance $\nu$.

- **BINARY**$(p)$
  - specifies a binary (Bernoulli) distribution that has probability $p$.

- **BINOMIAL**$(n, p)$
  - specifies a binomial distribution that has count $n$ and probability $p$.

- **GAMMA**$(a, b)$
  - specifies a gamma distribution that has shape $a$ and scale $b$.

- **NEGBIN**$(n, p)$
  - specifies a negative binomial distribution that has count $n$ and probability $p$.

- **POISSON**$(m)$
  - specifies a Poisson distribution that has mean $m$. 

GENERAL (ll) specifies a general log-likelihood function that you construct by using SAS programming statements.

The MODEL statement must follow any SAS programming statements that you specify for computing parameters of the preceding distributions. For information about the built-in log-likelihood functions, see the section “Built-In Log-Likelihood Functions” on page 4814.

PARAMETERS Statement

PARAMETERS < parameter-specification > <, . . . , parameter-specification > < / options > ;
PARMS < parameter-specification > <, . . . , parameter-specification > < / options > ;

The purpose of the PARAMETERS statement is to provide starting values for the HPNLMOD procedure. You can provide values that define a single point in the parameter space or that define a set of points. For more information about parameter-specification, see the section “Assigning Starting Values by Using Parameter Specification” on page 4808.

You can specify the following options in the PARAMETERS statement after the slash (/).

BEST=i > 0

specifies the maximum number of parameter grid points and the corresponding objective function values to display in the “Parameters” table. If you specify this option, the parameter grid points are listed in ascending order of objective function value. By default, all parameter grid points are displayed.

PDATA=SAS-data-set
DATA=SAS-data-set
specifies the data set that provides parameter starting values.

START=value
DEFSTART=value
specifies a default starting value for all parameters.

There are four methods available for providing starting values to the optimization process. In descending order of precedence, the methods are as follows:

1. Specify values directly in the PARAMETERS statement.
2. Specify values in the PDATA= data set option.
3. Specify a single value for all parameters by using the START= option.
4. Use the default value 1.0.

The names that are 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 HPNLMOD statement). Parameters that are assigned starting values through the PARAMETERS statement can be omitted from the estimation if the expression in the MODEL statement does not depend on them.
Assigning Starting Values by Using Parameter Specification

A parameter-specification has the following general form, where name identifies the parameter and value-list provides the set of starting values for the parameter:

\[ \text{name} = \text{value-list} \]

Often the value-list contains only a single value, but more general and flexible list specifications such as the following are possible:

- \( m \) a single value
- \( m_1, m_2, \ldots, m_n \) several values
- \( m \text{ TO } n \) a sequence in which \( m \) equals the starting value, \( n \) equals the ending value, and the increment equals 1
- \( m \text{ TO } n \text{ BY } i \) a sequence in which \( m \) equals the starting value, \( n \) equals the ending value, and the increment is \( i \)
- \( m_1, m_2 \text{ TO } m_3 \) mixed values and sequences

When you specify more than one value for a parameter, PROC HPNLMOD 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 HPNLMOD 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 PARAMETERS statement specifies five parameters and sets their possible starting values as shown in the following table:

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

<table>
<thead>
<tr>
<th>Possible Starting Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

The objective function values are calculated for each of the \( 1 \times 5 \times 4 \times 3 \times 6 = 360 \) combinations of possible starting values. Each grid point’s objective function value is computed by using the execution mode that is specified in the PERFORMANCE statement.

If you specify a starting value by using a parameter-specification, any starting values that are provided for this parameter through the PDATA= data set are ignored. The parameter-specification overrides the information in the PDATA= data set.
Assigning Starting Values from a SAS Data Set

The PDATA= option in the PARAMETERS statement enables you to assign starting values for parameters by using a SAS data set. The data set must contain at least two variables: a character variable named Parameter or Parm that identifies the parameter, and a numeric variable named Estimate or Est that contains the starting values. For example, the PDATA= option enables you to use the contents of the “ParameterEstimates” table from one PROC HPNLMOD run to supply starting values for a subsequent run, as follows:

```
proc hpnlmod data=d(obs=30);
   parameters alpha=100 beta=3 gamma=4;
   Switch = 1/(1+gamma*exp(beta*log(dose)));
   model y ~ residual(alpha*Switch);
   ods output ParameterEstimates=pest;
run;

proc hpnlmod data=d;
   parameters / pdata=pest;
   Switch = 1/(1+gamma*exp(beta*log(dose)));
   model y ~ residual(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
;
```

PERFORMANCE Statement

```
PERFORMANCE < performance-options > ;
```

The PERFORMANCE statement defines performance parameters for multithreaded and distributed computing, passes variables that describe the distributed computing environment, and requests detailed results about the performance characteristics of the procedure.

You can also use the PERFORMANCE statement to control whether PROC HPNLMOD executes in single-machine mode or distributed mode.

PREDICT Statement

PREDICT 'label' expression <options> ;

PREDICT 'label' MEAN <options> ;

The PREDICT statement enables you to construct predictions of an expression across all of the observations in the input data set. Multiple PREDICT statements are permitted. Results for all PREDICT statements are placed in the output data set that you specify in the OUT= option in the PROC HPNLMOD statement. For more information, see the section “Output Data Sets” (Chapter 2, SAS/STAT User’s Guide: High-Performance Procedures).

You must specify the following arguments:

'label'
identifies the predicted expression.

expression | MEAN
provides the predicted value. You can specify the predicted value either by using a SAS programming expression that involves the input data set variables and parameters or by using the keyword MEAN. If you specify the keyword MEAN, the predicted mean value for the distribution specified in the MODEL statement is used. Predicted values are computed using the final parameter estimates. Standard errors of prediction are computed using the delta method (Billingsley 1986; Cox 1998).

You can also specify the following options:

ALPHA=\(\alpha\)
specifies the alpha level to be used to compute confidence limits. The default value corresponds to the ALPHA= option in the PROC HPNLMOD statement.

DF=d
specifies the degrees of freedom to be used to compute confidence limits. The default value corresponds to the DF= option in the PROC HPNLMOD statement.

LOWER=name
specifies a variable that contains the lower confidence limit of the predicted value.

PRED=name
specifies a variable that contains the predicted value.

PROBT=name
specifies a variable that contains the \(p\)-value of the predicted value.

STDERR=name
specifies a variable that contains the standard error of the predicted value.
RESTRICT Statement

RESTRICT restriction1 <, restriction2 . . . > ;

The RESTRICT statement imposes linear restrictions on the model’s parameters estimates. You can specify any number of RESTRICT statements.

Each restriction is written as an expression, optionally followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression as follows:

expression < operator expression>

The operator can be =, <, >, <= or >=. The operator and second expression are optional. When they are omitted, the operator defaults to = and the second expression defaults to the value 0.

Restriction expressions can be composed of parameter names, arithmetic operators, functions, and constants. Comparison operators (such as = or <) and logical operators (such as &) cannot be used in RESTRICT statement expressions. Parameters that are named in restriction expressions must be among the parameters that are estimated by the model. Restriction expressions cannot refer to other variables that are defined in the program or the DATA= data set. The restriction expressions must be linear functions of the parameters.

The following example illustrates how to use the RESTRICT statement to impose a linear constraint on parameters:

```plaintext
proc hpnlmod;
  parms alpha beta;
  f = (x/alpha + beta)**2
  model y ~ residual(f);
  restrict beta < 2*(alpha + constant('pi'));
run;
```

The preceding RESTRICT statement represents the following model constraint:

\[ \beta < 2(\alpha + \pi) \]
Programming Statements

Programming statements define the arguments of the MODEL, ESTIMATE, and PREDICT statements in PROC HPNLMOD. Most of the programming statements that can be used in the SAS DATA step can also be used in the HPNLMOD procedure. See SAS Language Reference: Concepts for a description of SAS programming statements. The following are valid programming statements:

ABORT;
CALL name [ ( expression [, expression … ] ) ];
DELETE;
DO [variable = expression
   [ TO expression ] [ BY expression ]
   [, expression [ TO expression ] [ BY expression ] … ]
   ]
   [ WHILE expression ] [ UNTIL expression ];
END;
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)];
STOP;
SUBSTR( variable, index, length ) = expression;
WHEN (expression) program_statement;
   OTHERWISE program_statement;

For the most part, the SAS programming statements work the same as they do in the SAS DATA step, as documented in SAS Language Reference: Concepts. However, they differ as follows:

- The ABORT statement does not allow any arguments.
- The DO statement does not allow a character index variable. Thus, the first of the following statements is supported, but the second is not:

  do i = 1,2,3;

  do i = 'A','B','C';
In contrast to other procedures that share PROC HPNLMOD’s programming syntax, PROC HPNLMOD does not support the LAG function. Because observations are not processed sequentially when high-performance analytical procedures perform the parameter optimization, information for computing lagged values is not available.

The PUT statement, used mostly for program debugging in PROC HPNLMOD, 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 have:

- The PROC HPNLMOD PUT statement does not support line pointers, factored lists, iteration factors, overprinting, _INFILE_, the colon (:) format modifier, or “$”.

- The PROC HPNLMOD PUT statement supports expressions, but the expression must be enclosed in parentheses. For example, the following statement displays the square root of x:

```
put (sqrt(x));
```

- The PROC HPNLMOD PUT statement supports the item _PDV_, which displays a formatted listing of all variables in the program. For example, the following statement displays a much more readable listing of the variables than the _ALL_ print item:

```
put _pdv_;
```

The WHEN and OTHERWISE statements enable you to specify more than one programming statement. That is, DO/END groups are not necessary for multiple WHEN statements. For example, the following syntax is valid:

```
select;
  when (exp1) stmt1;
  stmt2;
  when (exp2) stmt3;
  stmt4;
end;
```

When you code your programming statements, avoid defining variables that begin with an underscore (_) because they might conflict with internal variables that are created by PROC HPNLMOD. The MODEL statement must come after any SAS programming statements that define or modify terms that are used to specify the model.
Details: HPNLMOD Procedure

Least Squares Estimation

Models that are estimated by PROC HPNLMOD can be represented by using the equations

\[ Y = f(\beta; z_1, \cdots, z_k) + \epsilon \]
\[ E[\epsilon] = 0 \]
\[ \text{Var}[\epsilon] = \sigma^2 I \]

where

- \(Y\) is the \((n \times 1)\) vector of observed responses
- \(f\) is the nonlinear prediction function of parameters and regressor variables
- \(\beta\) is the vector of model parameters to be estimated
- \(z_1, \cdots, z_k\) are the \((n \times 1)\) vectors for each of the \(k\) regressor variables
- \(\epsilon\) is the \((n \times 1)\) vector of residuals
- \(\sigma^2\) is the variance of the residuals

In these models, the distribution of the residuals is not specified and the model parameters are estimated using the least squares method. For the standard errors and confidence limits in the “ParameterEstimates” table to apply, the errors are assumed to be homoscedastic, uncorrelated, and have zero mean.

Built-In Log-Likelihood Functions

For models in which the distribution of model errors is specified, the HPNLMOD procedure estimates parameters by maximizing the value of a log-likelihood function for the specified distribution. The log-likelihood functions used by PROC HPNLMOD for the supported error distributions are as follows:

\( Y \sim \text{normal}(m, v) \)

\[ l(m, v; y) = -\frac{1}{2} \left( \log(2\pi) + \frac{(y - m)^2}{v} + \log(v) \right) \]
\[ E[Y] = m \]
\[ \text{Var}[Y] = v \]
\[ v > 0 \]
$Y \sim \text{binary}(p)$

\[
l_1(p; y) = \begin{cases} 
  y \log(p) & y > 0 \\
  0 & \text{otherwise} 
\end{cases} 
\]

\[
l_2(p; y) = \begin{cases} 
  (1 - y) \log(1 - p) & y < 1 \\
  0 & \text{otherwise} 
\end{cases} 
\]

\[l(p; y) = l_1(p; y) + l_2(p; y)\]

\[E[Y] = p\]

\[\text{Var}[Y] = p(1 - p)\]

$0 < p < 1$

$Y \sim \text{binomial}(n, p)$

\[
l_c = \log\{\Gamma(n + 1)\} - \log\{\Gamma(y + 1)\} - \log\{\Gamma(n - y + 1)\} 
\]

\[
l_1(n, p; y) = \begin{cases} 
  y \log(p) & y > 0 \\
  0 & \text{otherwise} 
\end{cases} 
\]

\[
l_2(n, p; y) = \begin{cases} 
  (n - y) \log(1 - p) & n - y > 0 \\
  0 & \text{otherwise} 
\end{cases} 
\]

\[l(n, p; y) = l_c + l_1(n, p; y) + l_2(n, p; y)\]

\[E[Y] = np\]

\[\text{Var}[Y] = np(1 - p)\]

$0 < p < 1$

$Y \sim \text{gamma}(a, b)$

\[
l(a, b; y) = -a \log(b) - \log(\Gamma(a)) + (a - 1) \log(y) - y/b 
\]

\[E[Y] = ab\]

\[\text{Var}[Y] = ab^2\]

$a > 0$

$b > 0$

This parameterization of the gamma distribution differs from the parameterization that the GLIMMIX and GENMOD procedures use. The scale parameter in PROC HPNLMOD is expressed as the inverse of the scale parameter that PROC GLIMMIX and PROC GENMOD use. The PROC HPNLMOD parameter represents the scale of the magnitude of the residuals. The scale parameter in PROC GLIMMIX can be estimated by using the following statements:

```plaintext
proc glimmix;
  model y = x / dist=gamma s;
run;
```

The following statements show how to use PROC HPNLMOD to estimate the equivalent scale parameter:
Chapter 60: The HPNLMOD Procedure

```sas
proc hpnlmod;
   parms b0=1 b1=0 scale=14;
   linp = b0 + b1*x;
   mu = exp(linp);
   b = mu*scale;
   model y ~ gamma(1/scale,b);
run;
```

\[ Y \sim \text{negbin}(n, p) \]

\[
I(n, p; y) = \log\{\Gamma(n + y)\} - \log\{\Gamma(n)\} - \log\{\Gamma(y + 1)\} \\
+ n \log\{p\} + y \log\{1 - p\}
\]

\[
E[Y] = n \left( \frac{1 - p}{p} \right)
\]

\[
\text{Var}[Y] = n \left( \frac{1 - p}{p^2} \right)
\]

\[ n \geq 0 \]
\[ 0 < p < 1 \]

The parameter \( n \) can be real-numbered; it does not have to be integer-valued.

\[ Y \sim \text{Poisson}(m) \]

\[
I(m; y) = y \log\{m\} - m - \log\{\Gamma(y + 1)\}
\]

\[ E[Y] = m \]

\[ \text{Var}[Y] = m \]

\[ m > 0 \]

---

**Computational Method**

**Multithreading**

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, substantial performance gains can be realized compared to sequential (single-threaded) execution.

The number of threads that the HPNLMOD procedure spawns is determined by the number of CPUs on a machine and can be controlled in the following ways:

- You can specify the CPU count by using the CPUCOUNT= SAS system option. For example, if you specify the following statement, the HPNLMOD procedure determines threading as if it executed on a system that has four CPUs, regardless of the actual CPU count:
You can specify the NTHREADS= option in the PERFORMANCE statement to determine the number of threads. This specification overrides the CPUCOUNT= system option. Specify NTHREADS=1 to force single-threaded execution.

The number of threads per machine is displayed in the “Performance Information” table, which is part of the default output. The HPNLMOD procedure allocates one thread per CPU.

The HPNLMOD procedure divides the data that are processed on a single machine among the threads—that is, the HPNLMOD procedure implements multithreading by distributing computations across the data. For example, if the input data set has 1,000 observations and PROC HPNLMOD is running with four threads, then 250 observations are associated with each thread. All operations that require access to the data are then multithreaded. These operations include the following:

- calculation of objective function values for the initial parameter grid
- objective function calculation
- gradient calculation
- Hessian calculation
- scoring of observations

In addition, operations on matrices such as sweeps might be multithreaded, provided that the matrices are of sufficient size to realize performance benefits from managing multiple threads for the particular matrix operation.

**Choosing an Optimization Algorithm**

**First- or Second-Order Algorithms**

The factors that affect how you choose a particular optimization technique for a particular problem are complex. Occasionally, you might benefit from trying several algorithms.

For many optimization problems, computing the gradient takes more computer time than computing the function value. Computing the Hessian sometimes takes much more computer time and memory than computing the gradient, especially when there are many decision variables. Unfortunately, optimization techniques that do not use some kind of Hessian approximation usually require many more iterations than techniques that do use a Hessian matrix; as a result, the total run time of these techniques is often longer. Techniques that do not use the Hessian also tend to be less reliable. For example, they can terminate more easily at stationary points than at global optima.

Table 60.3 shows which derivatives are required for each optimization technique.
The second-derivative methods (TRUREG, NEWRAP, and NRRIDG) are best for small problems for which the Hessian matrix is not expensive to compute. Sometimes the NRRIDG algorithm can be faster than the TRUREG algorithm, but TRUREG can be more stable. The NRRIDG algorithm requires only one matrix with \( p(p + 1)/2 \) double words; TRUREG and NEWRAP require two such matrices. Here, \( p \) denotes the number of parameters in the optimization.

The first-derivative methods QUANEW and DBLDOG are best for medium-sized problems for which the objective function and the gradient are much faster to evaluate than the Hessian. In general, the QUANEW and DBLDOG algorithms require more iterations than TRUREG, NRRIDG, and NEWRAP, but each iteration can be much faster. The QUANEW and DBLDOG algorithms require only the gradient to update an approximate Hessian, and they require slightly less memory than TRUREG or NEWRAP.

The first-derivative method CONGRA is best for large problems for which the objective function and the gradient can be computed much faster than the Hessian and for which too much memory is required to store the (approximate) Hessian. In general, the CONGRA algorithm requires more iterations than QUANEW or DBLDOG, but each iteration can be much faster. Because CONGRA requires only a factor of \( p \) double-word memory, many large applications can be solved only by CONGRA.

The no-derivative method NMSIMP is best for small problems for which derivatives are not continuous or are very difficult to compute.

The LEVMAR method is appropriate only for least squares optimization problems.

Each optimization method uses one or more convergence criteria that determine when it has converged. An algorithm is considered to have converged when any one of the convergence criteria is satisfied. For example, under the default settings, the QUANEW algorithm converges if \( \text{ABSGCONV} < 1\times 10^{-5}, \text{FCONV} < 2 \times \epsilon, \) or \( \text{GCONV} < 1\times 10^{-8}. \)

By default, the HPNLMOD procedure applies the NRRIDG algorithm because it can take advantage of multithreading in Hessian computations and inversions. If the number of parameters becomes large, specifying \( \text{TECHNIQUE} = \text{QUANEW} \) (which is a first-order method with good overall properties) is recommended.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>First-Order</th>
<th>Second-Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUREG</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>NEWRAP</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>NRRIDG</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>QUANEW</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>DBLDOG</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>CONGRA</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>LEVMAR</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>NMSIMP</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Algorithm Descriptions

The following subsections provide details about each optimization technique and follow the same order as Table 60.3.

**Trust Region Optimization (TRUREG)**

The trust region method uses the gradient $g(\psi^{(k)})$ and the Hessian matrix $H(\psi^{(k)})$; thus, it requires that the objective function $f(\psi)$ have continuous first- and second-order derivatives inside the feasible region.

The trust region method iteratively optimizes a quadratic approximation to the nonlinear objective function within a hyperelliptic trust region that has radius $\Delta$. The radius constrains the step size that corresponds to the quality of the quadratic approximation. The trust region method is implemented based on Dennis, Gay, and Welsch (1981); Gay (1983); More and Sorensen (1983).

The trust region method performs well for small- to medium-sized problems, and it does not need many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the quasi-Newton or conjugate gradient algorithms might be more efficient.

**Newton-Raphson Optimization with Line Search (NEWRAP)**

The NEWRAP technique uses the gradient $g(\psi^{(k)})$ and the Hessian matrix $H(\psi^{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region.

If second-order derivatives are computed efficiently and precisely, the NEWRAP method can perform well for medium-sized to large problems, and it does not need many function, gradient, and Hessian calls.

This algorithm uses a pure Newton step when the Hessian is positive-definite and when the Newton step reduces the value of the objective function successfully. Otherwise, a combination of ridging and line search is performed to compute successful steps. If the Hessian is not positive-definite, a multiple of the identity matrix is added to the Hessian matrix to make it positive-definite (Eskow and Schnabel 1991).

In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The default line-search method uses quadratic interpolation and cubic extrapolation (LIS=2).

**Newton-Raphson Ridge Optimization (NRRIDG)**

The NRRIDG technique uses the gradient $g(\psi^{(k)})$ and the Hessian matrix $H(\psi^{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region.

This algorithm uses a pure Newton step when the Hessian is positive-definite and when the Newton step reduces the value of the objective function successfully. If at least one of these two conditions is not satisfied, a multiple of the identity matrix is added to the Hessian matrix.

The NRRIDG method performs well for small- to medium-sized problems, and it does not require many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the quasi-Newton or conjugate gradient algorithms might be more efficient.

Because the NRRIDG technique uses an orthogonal decomposition of the approximate Hessian, each iteration of NRRIDG can be slower than an iteration of the NEWRAP technique, which works with a Cholesky decomposition. However, NRRIDG usually requires fewer iterations than NEWRAP.
**Quasi-Newton Optimization (QUANEW)**

The (dual) quasi-Newton method uses the gradient $g(\psi^{(k)})$, and it does not need to compute second-order derivatives because they are approximated. It works well for medium-sized to moderately large optimization problems, where the objective function and the gradient can be computed much faster than the Hessian. However, in general it requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques, which compute second-order derivatives. QUANEW is the default optimization algorithm because it provides an appropriate balance between the speed and stability that are required for most nonlinear mixed model applications.

The QUANEW technique that is implemented by the HPNLMOD procedure is the dual quasi-Newton algorithm, which updates the Cholesky factor of an approximate Hessian.

In each iteration, a line search is performed along the search direction to find an approximate optimum. The line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size $\alpha$ that satisfies the Goldstein conditions (Fletcher 1987). One of the Goldstein conditions can be violated if the feasible region defines an upper limit of the step size. Violating the left-side Goldstein condition can affect the positive-definiteness of the quasi-Newton update. In that case, either the update is skipped or the iterations are restarted by using an identity matrix, resulting in the steepest descent or ascent search direction.

The QUANEW algorithm uses its own line-search technique.

**Double-Dogleg Optimization (DBLDOG)**

The double-dogleg optimization method combines the ideas of the quasi-Newton and trust region methods. In each iteration, the double-dogleg algorithm computes the step $s^{(k)}$ as the linear combination of the steepest descent or ascent search direction $s_1^{(k)}$ and a quasi-Newton search direction $s_2^{(k)}$:

$$s^{(k)} = \alpha_1 s_1^{(k)} + \alpha_2 s_2^{(k)}$$

The step is requested to remain within a prespecified trust region radius (Fletcher 1987, p. 107). Thus, the DBLDOG subroutine uses the dual quasi-Newton update but does not perform a line search.

The double-dogleg optimization technique works well for medium-sized to moderately large optimization problems, where the objective function and the gradient are much faster to compute than the Hessian. The implementation is based on Dennis and Mei (1979); Gay (1983), but it is extended for dealing with boundary and linear constraints. The DBLDOG technique generally requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques, which require second-order derivatives; however, each of the DBLDOG iterations is computationally cheap. Furthermore, the DBLDOG technique requires only gradient calls for the update of the Cholesky factor of an approximate Hessian.

**Conjugate Gradient Optimization (CONGRA)**

Second-order derivatives are not required by the CONGRA algorithm and are not even approximated. The CONGRA algorithm can be expensive in function and gradient calls, but it requires only $O(p)$ memory for unconstrained optimization. In general, the algorithm must perform many iterations to obtain a precise solution, but each of the CONGRA iterations is computationally cheap.

The CONGRA algorithm should be used for optimization problems that have large $p$. For the unconstrained or boundary-constrained case, the CONGRA algorithm requires only $O(p)$ bytes of working memory, whereas all other optimization methods require order $O(p^2)$ bytes of working memory. During $p$ successive iterations, uninterrupted by restarts or changes in the working set, the CONGRA algorithm computes a cycle of $p$ conjugate search directions. In each iteration, a line search is performed along the search direction to find an
approximate optimum of the objective function. The default line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size $\alpha$ that satisfies the Goldstein conditions. One of the Goldstein conditions can be violated if the feasible region defines an upper limit for the step size. Other line-search algorithms can be specified with the LIS= option.

**Levenberg-Marquardt Optimization (LEVMAR)**

The LEVMAR algorithm performs a highly stable optimization; however, for large problems, it consumes more memory and takes longer than the other techniques. The Levenberg-Marquardt optimization technique is a slightly improved variant of the Moré (1978) implementation.

**Nelder-Mead Simplex Optimization (NMSIMP)**

The Nelder-Mead simplex method does not use any derivatives and does not assume that the objective function has continuous derivatives. The objective function itself needs to be continuous. This technique is quite expensive in the number of function calls, and it might be unable to generate precise results for $p \gg 40$.

The original Nelder-Mead simplex algorithm is implemented and extended to boundary constraints. This algorithm does not compute the objective for infeasible points, but it changes the shape of the simplex by adapting to the nonlinearities of the objective function. This adaptation contributes to an increased speed of convergence. NMSIMP uses a special termination criterion.

---

**Displayed Output**

The following sections describe the output that PROC HPNLMOD produces by default. The output is organized into various tables, which are discussed in the order of their appearance.

**Performance Information**

The “Performance Information” table is produced by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node.

If you specify the DETAILS option in the PERFORMANCE statement, the procedure also produces a “Timing” table in which elapsed times (absolute and relative) for the main tasks of the procedure are displayed.

**Specifications**

The “Specifications” table displays basic information about the model such as the data source, the dependent variable, the distribution being modeled, and the optimization technique.

**Number of Observations**

The “Number of Observations” table displays the number of observations that are read from the input data set and the number of observations that are used in the analysis.
Dimensions

The “Dimensions” table displays the number of parameters that are estimated in the model and the number of upper and lower bounds that are imposed on the parameters.

Parameters

The “Parameters” table displays the initial values of parameters that are used to start the estimation process. You can limit this information by specifying the BEST= option in the PARAMETERS statement when you specify a large number of initial parameter value combinations. The parameter combinations and their corresponding objective function values are listed in increasing order of objective function value.

Iteration History

For each iteration of the optimization, the “Iteration History” table displays the number of function evaluations (including gradient and Hessian evaluations), the value of the objective function, the change in the objective function from the previous iteration, and the absolute value of the largest (projected) gradient element.

Convergence Status

The convergence status table is a small ODS table that follows the “Iteration History” table in the default output. In the listing it appears as a message that identifies whether the optimization succeeded and which convergence criterion was met. If the optimization fails, the message indicates the reason for the failure. If you save the convergence status table to an output data set, a numeric Status variable is added that enables you to programmatically assess convergence. The values of the Status variable encode the following:

0 Convergence was achieved or an optimization was not performed because TECHNIQUE=NONE.
1 The objective function could not be improved.
2 Convergence was not achieved because of a user interrupt or because a limit (such as the maximum number of iterations or the maximum number of function evaluations) was reached. To modify these limits, see the MAXITER=, MAXFUNC=, and MAXTIME= options in the PROC HPNLMOD statement.
3 Optimization failed to converge because function or derivative evaluations failed at the starting values or during the iterations or because a feasible point that satisfies the parameter constraints could not be found in the parameter space.

Linear Constraints

The “Linear Constraints” table summarizes the linear constraints that are applied to the model by using the RESTRICT statements. All the constraints that are specified in the model are listed in the “Linear Constraints” table, together with information about whether each constraint represents an inequality or equality condition and whether that constraint is active for the final parameter estimates.

Fit Statistics

The “Fit Statistics” table displays a variety of measures of fit, depending on whether the model was estimated using least squares or maximum likelihood. In both cases, smaller values of the fit statistics indicate better fit.
For least squares estimations, the “Fit Statistics” table displays the sum of squares of errors and the variance of errors.

For maximum likelihood estimations, the table uses the following formulas to display information criteria, where \( p \) denotes the number of effective parameters, \( n \) denotes the number of observations used, and \( l \) is the log likelihood that is evaluated at the converged estimates:

\[
AIC = -2l + 2p \\
AICC = \begin{cases} 
-2l + 2pn/(n - p - 1), & f > p + 2 \\
-2l + 2p(p + 2), & \text{otherwise}
\end{cases} \\
BIC = -2l + p \log(f)
\]

The information criteria values that are displayed in the “Fit Statistics” table are not based on a normalized log-likelihood function.

**ANOVA**

The “Analysis of Variance” table is displayed only for least squares estimations. The ANOVA table displays the number of degrees of freedom and the sum of squares that are attributed to the model, the error, and the total. The ANOVA table also reports the variance of the model and the errors, the \( F \) statistic, and its probability for the model.

**Parameter Estimates**

The “Parameter Estimates” table displays the parameter estimates, their estimated (asymptotic) standard errors \( t \) statistics, and associated \( p \)-values for the hypothesis that the parameter is 0. Confidence limits are displayed for each parameter and are based on the value of the ALPHA= option in the PROC HPNLMOD statement.

**Additional Estimates**

The “Additional Estimates” table displays the same information as the “Parameter Estimates” table for the expressions that appear in the optional ESTIMATE statements. The table is generated when one or more ESTIMATE statements are specified. Because a separate ALPHA= option can be specified for each ESTIMATE statement, the “Additional Estimates” table also includes a column that indicates each confidence interval’s corresponding significance level.

**Covariance**

The “Covariance” table appears when the COV option is specified in the PROC HPNLMOD statement. The “Covariance” table displays a matrix of covariances between each pair of estimated parameters.

**Correlation**

The “Correlation” table appears when the CORR option is specified in the PROC HPNLMOD statement. The “Correlation” table displays the correlation matrix for the estimated parameters.
Additional Estimates Covariance

The “Covariance of Additional Estimates” table appears when the ECOV option is specified in the PROC HPNLMOD statement. The “Covariance of Additional Estimates” table displays a matrix of covariances between each pair of expressions that are specified in ESTIMATE statements.

Additional Estimates Correlation

The “Correlation of Additional Estimates” table appears when the ECORR option is specified in the PROC HPNLMOD statement. The “Correlation of Additional Estimates” table displays the correlation matrix for the expressions that are specified in ESTIMATE statements.

Procedure Task Timing

If you specify the DETAILS option in the PERFORMANCE statement, the procedure also produces a “Procedure Task Timing” table in which elapsed times (absolute and relative) for the main tasks of the procedure are displayed.

ODS Table Names

Each table that is created by the HPNLMOD procedure has a name associated with it, and you must use this name to refer to the table when you use ODS statements. These names are listed in Table 60.4.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Required Statement and Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdditionalEstimates</td>
<td>Functions of estimated parameters and their associated statistics</td>
<td>ESTIMATE statement</td>
</tr>
<tr>
<td>ANOVA</td>
<td>Least squares analysis of variance information</td>
<td>RESIDUAL option in the MODEL statement</td>
</tr>
<tr>
<td>Constraints</td>
<td>Information about the model’s linear constraints</td>
<td>RESTRICT statement</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Optimization success and convergence information</td>
<td>Default output</td>
</tr>
<tr>
<td>CorrB</td>
<td>Parameter correlation matrix</td>
<td>CORR option in the PROC HPNLMOD statement</td>
</tr>
<tr>
<td>CovB</td>
<td>Parameter covariance matrix</td>
<td>COV option in the PROC HPNLMOD statement</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Number of parameters and their bounds</td>
<td>Default output</td>
</tr>
<tr>
<td>ECorrB</td>
<td>Additional estimates’ correlation matrix</td>
<td>ECORR option in the PROC HPNLMOD statement</td>
</tr>
<tr>
<td>ECovB</td>
<td>Additional estimates’ covariance matrix</td>
<td>ECOV option in the PROC HPNLMOD statement</td>
</tr>
</tbody>
</table>
Table 60.4  continued

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Required Statement and Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FitStatistics</td>
<td>Statistics about the quality of the fit</td>
<td>Default output</td>
</tr>
<tr>
<td>IterHistory</td>
<td>Optimizer iteration information</td>
<td>Default output</td>
</tr>
<tr>
<td>NObs</td>
<td>Number of observations read and used</td>
<td>Default output</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates and associated statistics</td>
<td>Default output</td>
</tr>
<tr>
<td>Parameters</td>
<td>Initial parameter values</td>
<td>Default output</td>
</tr>
<tr>
<td>PerformanceInfo</td>
<td>Information about high-performance computing environment</td>
<td>Default output</td>
</tr>
<tr>
<td>Specifications</td>
<td>Basic model characteristics</td>
<td>Default output</td>
</tr>
</tbody>
</table>

Examples: HPNLMOD 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 the mean of $Y$ is a quadratic function in $x$ for values of $x$ less than $x_0$ and that the mean of $Y$ is constant for values of $x$ greater than $x_0$:

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

In this model equation, $\alpha$, $\beta$, and $\gamma$ are the coefficients of the quadratic segment, and $\gamma$ is the plateau of the mean function. The HPNLMOD procedure can fit such a segmented model even when the join point, $x_0$, is unknown.

Suppose you 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 $x$ needs to coincide at $x_0$.

The continuity condition requires that

$$c = E[Y|x_0] = \alpha + \beta x_0 + \gamma x_0^2$$

The smoothness condition requires that

$$\frac{\partial E[Y|x_0]}{\partial x} = \beta + 2\gamma x_0 \equiv 0$$
If you solve for \( x_0 \) and substitute into the expression for \( c \), the two conditions jointly imply that

\[
\begin{align*}
  x_0 &= -\beta / 2\gamma \\
  c &= \alpha - \beta^2 / 4\gamma
\end{align*}
\]

Although there are five unknowns, the model contains only three independent parameters. The continuity and smoothness restrictions together completely determine two parameters, given the other three.

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

```sas
data a;
  input y x @@;
datelines;
  .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 HPNLMOD statements fit this segmented model:

```sas
proc hpnlmod data=a out=b;
  parms alpha=.45 beta=.05 gamma=-.0025;
  x0 = -.5*beta / gamma;
  if (x < x0) then
    yp = alpha + beta*x + gamma*x*x;
  else
    yp = alpha + beta*x0 + gamma*x0*x0;
  model y ~ residual(yp);
  estimate 'join point' -beta/2/gamma;
  estimate 'plateau value c' alpha - beta**2/(4*gamma);
  predict 'predicted' yp pred=yp;
  predict 'response' y pred=y;
  predict 'x' x pred=x;
run;
```

The parameters of the model are \( \alpha \), \( \beta \), and \( \gamma \). They are represented in the PROC HPNLMOD statements by the variables `alpha`, `beta`, and `gamma`, respectively. In order to model the two segments, a conditional statement assigns the appropriate expression to the mean function, depending on the value of \( x_0 \). The ESTIMATE statements compute the values of \( x_0 \) and \( c \). The PREDICT statement computes predicted values for plotting and saves them to data set b.

The results from fitting this model are shown in Output 60.1.1 through Output 60.1.3. The iterative optimization converges after six iterations (Output 60.1.1). Output 60.1.2 shows the estimated parameters. Output 60.1.3 indicates that the join point is 12.7477 and the plateau value is 0.7775.
Output 60.1.1 Nonlinear Least Squares Iterative Phase

Quadratic Model with Plateau

The HPNLMOD Procedure

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Evaluations</th>
<th>Objective Function</th>
<th>Change</th>
<th>Max Gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>0.0035144531</td>
<td>7.184063</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.0007352716</td>
<td>0.00277918</td>
<td>2.145337</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.0006292751</td>
<td>0.00010600</td>
<td>0.032551</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.0006291261</td>
<td>0.00000015</td>
<td>0.002952</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0.0006291244</td>
<td>0.00000000</td>
<td>0.000238</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0.0006291244</td>
<td>0.00000000</td>
<td>0.000023</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0.0006291244</td>
<td>0.00000000</td>
<td>2.313E-6</td>
</tr>
</tbody>
</table>

Convergence criterion (GCONV=1E-8) satisfied.

Output 60.1.2 Least Squares Analysis for the Quadratic Model

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Approx Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>2</td>
<td>0.1769</td>
<td>0.0884</td>
<td>114.22</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>13</td>
<td>0.0101</td>
<td>0.000774</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>15</td>
<td>0.1869</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Output 60.1.3 Additional Estimates for the Quadratic Model

<table>
<thead>
<tr>
<th>Label</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>DF</th>
<th>t Value</th>
<th>Approx Pr &gt;</th>
<th>Approximate 95% Confidence Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>join point</td>
<td>12.7477</td>
<td>1.2781</td>
<td>1</td>
<td>9.97</td>
<td>&lt;.0001</td>
<td>0.05 9.9864 15.5089</td>
</tr>
<tr>
<td>plateau value c</td>
<td>0.7775</td>
<td>0.0123</td>
<td>1</td>
<td>63.11</td>
<td>&lt;.0001</td>
<td>0.05 0.7509 0.8041</td>
</tr>
</tbody>
</table>

The following statements produce a graph of the observed and predicted values along with reference lines for the join point and plateau estimates (Output 60.1.4):
proc sgplot data=b noautolegend;
  yaxis label='Observed or Predicted';
  repline 0.7775 / axis=y label="Plateau"  labelpos=min;
  repline 12.7477 / 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

References


Subject Index

Bernoulli distribution  
  HPNLMOD procedure, 4806
binary distribution  
  HPNLMOD procedure, 4806
binomial distribution  
  HPNLMOD procedure, 4806
boundary constraints, 4805
bounds  
  HPNLMOD procedure, 4805
computational method  
  HPNLMOD procedure, 4816
convergence criterion  
  HPNLMOD procedure, 4801, 4802
displayed output  
  HPNLMOD procedure, 4821
distribution functions  
  HPNLMOD procedure, 4814
empirical Bayes estimate  
  HPNLMOD procedure, 4810
examples, HPNLMOD procedure  
  boundary specification, 4805
  conditional model expression, 4826
  enzyme data, 4797
  gamma distribution, 4815
  join point, 4826
  plateau model, 4826
  predicted values, 4826
  segmented model, 4826
  starting values, data set, 4809
  starting values, grid, 4808
gamma distribution  
  HPNLMOD procedure, 4806
Gaussian distribution  
  HPNLMOD procedure, 4806
general distribution  
  HPNLMOD procedure, 4807
HPNLMOD procedure, 4795  
  additional estimates, 4806, 4823
  additional estimates correlation, 4824
  additional estimates covariance, 4824
  ANOVA, 4823
  Bernoulli distribution, 4806
  binary distribution, 4806
  binomial distribution, 4806
  bounds, 4805
  computational method, 4816
  convergence criterion, 4801, 4802
  convergence status, 4822
  correlation, 4823
  covariance, 4823
  dimensions, 4822
  displayed output, 4821
  distribution functions, 4814
  empirical Bayes estimate, 4810
  fit statistics, 4822
  function-based convergence criteria, 4801, 4802
  gamma distribution, 4806
  general distribution, 4807
  gradient-based convergence criteria, 4801, 4802
  initial values, 4807
  input data sets, 4802
  iteration history, 4822
  lag functionality, 4813
  least squares, 4814
  least squares distribution, 4806
  linear constraints, 4822
  log-likelihood functions, 4814
  multithreading, 4809, 4816
  negative binomial distribution, 4806
  normal distribution, 4806
  number of observations, 4821
  ODS table names, 4824
  optimization technique, 4817
  output data sets, 4804
  parameter estimates, 4823
  parameters, 4822
  performance information, 4821
  Poisson distribution, 4806
  prediction, 4810
  procedure task timing, 4824
  programming statements, 4812
  residual distribution, 4806
  restrictions, 4811
  segmented model example, 4825
  specifications, 4821
  starting values, 4807
  user-defined formats, 4802, 4805
  XML input stream, 4802, 4805
initial values  
  HPNLMOD procedure, 4807
lag functionality
  HPNLMOD procedure, 4813
least squares
  HPNLMOD procedure, 4814
least squares distribution
  HPNLMOD procedure, 4806
log-likelihood functions
  HPNLMOD procedure, 4814

model
  specification (HPNLMOD procedure, 4806
multithreading
  HPNLMOD procedure, 4809, 4816

negative binomial distribution
  HPNLMOD procedure, 4806
normal distribution
  HPNLMOD procedure, 4806

optimization technique
  HPNLMOD procedure, 4817
options summary
  PROC HPNLMOD statement, 4800

Poisson distribution
  HPNLMOD procedure, 4806
prediction
  HPNLMOD procedure, 4810
programming statements
  HPNLMOD procedure, 4812

residual distribution
  HPNLMOD procedure, 4806
restrictions
  HPNLMOD procedure, 4811

starting values
  HPNLMOD procedure, 4807
Syntax Index

ABSCONV option
   PROC HPNLMOD statement, 4801

ABSFCONV option
   PROC HPNLMOD statement, 4801

ABSGCONV option
   PROC HPNLMOD statement, 4801

ABSGTOL option
   PROC HPNLMOD statement, 4801

ABSTOL option
   PROC HPNLMOD statement, 4801

ALPHA= option
   ESTIMATE statement (HPNLMOD), 4806
   PREDICT statement (HPNLMOD), 4810
   PROC HPNLMOD statement, 4801

BOUNDS statement
   HPNLMOD procedure, 4805

BY statement
   HPNLMOD procedure, 4805

CORR option
   PROC HPNLMOD statement, 4801

COV option
   PROC HPNLMOD statement, 4801

DATA= option
   PROC HPNLMOD statement, 4802

DF= option
   ESTIMATE statement (HPNLMOD), 4806
   PREDICT statement (HPNLMOD), 4810
   PROC HPNLMOD statement, 4802

ECORR option
   PROC HPNLMOD statement, 4802

ECOV option
   PROC HPNLMOD statement, 4802

ESTIMATE statement
   HPNLMOD procedure, 4806

FCONV option
   PROC HPNLMOD statement, 4802

FMTLIBXML= option
   PROC HPNLMOD statement, 4802

FTOL option
   PROC HPNLMOD statement, 4802

GCONV option
   PROC HPNLMOD statement, 4802

GTOL option
   PROC HPNLMOD statement, 4802

PROC HPNLMOD statement, 4802

HPNLMOD procedure, 4799
   PERFORMANCE statement, 4809
   PROC HPNLMOD statement, 4800
   syntax, 4799

HPNLMOD procedure, BOUNDS statement, 4805

HPNLMOD procedure, BY statement, 4805

HPNLMOD procedure, ESTIMATE statement, 4806
   ALPHA= option, 4806
   DF= option, 4806

HPNLMOD procedure, MODEL statement, 4806

HPNLMOD procedure, PARAMETERS statement, 4807

HPNLMOD procedure, PERFORMANCE statement, 4809

HPNLMOD procedure, PREDICT statement, 4810
   ALPHA= option, 4810
   DF= option, 4810
   LOWER= option, 4810
   PRED= option, 4810
   PROBT= option, 4810
   STDERR= option, 4810
   TVALUEN= option, 4811
   UPPER= option, 4811

HPNLMOD procedure, PROC HPNLMOD statement, 4800

   ABSCONV option, 4801
   ABSFCONV option, 4801
   ABSFTOL option, 4801
   ABSGCONV option, 4801
   ABSGTOL option, 4801
   ABSTOL option, 4801
   ALPHA= option, 4801
   CORR option, 4801
   COV option, 4801
   DATA= option, 4802
   DF= option, 4802
   ECORR option, 4802
   ECOV option, 4802
   ESTIMATE statement (HPNLMOD), 4806
   PREDICT statement (HPNLMOD), 4810
   PROC HPNLMOD statement, 4802

   MAXFUCN= option, 4803
   MAXITER= option, 4803, 4804
   MAXTIME= option, 4803
NOITPRINT option, 4804
NOPRINT option, 4804
OUT= option, 4804
SINGULAR= option, 4804
TECHNIQUE= option, 4804
XMLFORMAT= option, 4805
HPNLMOD procedure, RESTRICT statement, 4811

LOWER= option
  PREDICT statement (HPNLMOD), 4810

MAXFUNC= option
  PROC HPNLMOD statement, 4803
MAXITER= option
  PROC HPNLMOD statement, 4803, 4804
MAXTIME= option
  PROC HPNLMOD statement, 4803
MODEL statement
  HPNLMOD procedure, 4806

NOITPRINT option
  PROC HPNLMOD statement, 4804
NOPRINT option
  PROC HPNLMOD statement, 4804

OUT= option
  PROC HPNLMOD statement, 4804

PARAMETERS statement
  HPNLMOD procedure, 4807
PERFORMANCE statement
  HPNLMOD procedure, 4809
PRED= option
  PREDICT statement (HPNLMOD), 4810
PREDICT statement
  HPNLMOD procedure, 4810
PROBT= option
  PREDICT statement (HPNLMOD), 4810
PROC HPNLMOD statement, see HPNLMOD procedure
  HPNLMOD procedure, 4800

RESTRICT statement
  HPNLMOD procedure, 4811

SINGULAR= option
  PROC HPNLMOD statement, 4804
STDErr= option
  PREDICT statement (HPNLMOD), 4810
syntax
  HPNLMOD procedure, 4799

TECHNIQUE= option
  PROC HPNLMOD statement, 4804
TVALUE= option