SAS/STAT 15.2® User's Guide
The NLMIXED Procedure

SAS® Documentation
November 06, 2020

This document is an individual chapter from *SAS/STAT® 15.2 User's Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2020. *SAS/STAT® 15.2 User's Guide*. Cary, NC: SAS Institute Inc.

SAS/STAT® 15.2 User's Guide

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

November 2020

Chapter 88
The NLMIXED Procedure

Contents

Overview: NLMIXED Procedure

Introduction

The NLMIXED procedure fits nonlinear mixed models—that is, models in which both fixed and random effects enter nonlinearly. These models have a wide variety of applications, two of the most common being pharmacokinetics and overdispersed binomial data. PROC NLMIXED enables you to specify a conditional distribution for your data (given the random effects) having either a standard form (normal, binomial, Poisson) or a general distribution that you code using SAS programming statements.

PROC NLMIXED fits nonlinear mixed models by maximizing an approximation to the likelihood integrated over the random effects. Different integral approximations are available, the principal ones being adaptive Gaussian quadrature and a first-order Taylor series approximation. A variety of alternative optimization techniques are available to carry out the maximization; the default is a dual quasi-Newton algorithm.

Successful convergence of the optimization problem results in parameter estimates along with their approximate standard errors based on the second derivative matrix of the likelihood function. PROC NLMIXED enables you to use the estimated model to construct predictions of arbitrary functions by using empirical Bayes estimates of the random effects. You can also estimate arbitrary functions of the nonrandom parameters, and PROC NLMIXED computes their approximate standard errors by using the delta method.

Literature on Nonlinear Mixed Models

Davidian and Giltinan (1995) and Vonesh and Chinchilli (1997) provide good overviews as well as general theoretical developments and examples of nonlinear mixed models. Pinheiro and Bates (1995) is a primary reference for the theory and computational techniques of PROC NLMIXED. They describe and compare several different integrated likelihood approximations and provide evidence that adaptive Gaussian quadrature is one of the best methods. Davidian and Gallant (1993) also use Gaussian quadrature for nonlinear mixed models, although the smooth nonparametric density they advocate for the random effects is currently not available in PROC NLMIXED.

Traditional approaches to fitting nonlinear mixed models involve Taylor series expansions, expanding around either zero or the empirical best linear unbiased predictions of the random effects. The former is the basis for the well-known first-order method (Beal and Sheiner 1982, 1988; Sheiner and Beal 1985), and it is optionally available in PROC NLMIXED. The latter is the basis for the estimation method of Lindstrom and Bates (1990), and it is not available in PROC NLMIXED. However, the closely related Laplacian approximation is an option; it is equivalent to adaptive Gaussian quadrature with only one quadrature point. The Laplacian approximation and its relationship to the Lindstrom-Bates method are discussed by: Beal and Sheiner (1992); Wolfinger (1993); Vonesh (1992, 1996); Vonesh and Chinchilli (1997); Wolfinger and Lin (1997).

A parallel literature exists in the area of generalized linear mixed models, in which random effects appear as a part of the linear predictor inside a link function. Taylor-series methods similar to those just described are discussed in articles such as: Harville and Mee (1984); Stiratelli, Laird, and Ware (1984); Gilmour, Anderson, and Rae (1985); Goldstein (1991); Schall (1991); Engel and Keen (1992); Breslow and Clayton (1993); Wolfinger and O'Connell (1993); McGilchrist (1994), but such methods have not been implemented in PROC NLMIXED because they can produce biased results in certain binary data situations (Rodriguez and Goldman 1995; Lin and Breslow 1996). Instead, a numerical quadrature approach is available in PROC NLMIXED, as discussed in: Pierce and Sands (1975); Anderson and Aitkin (1985); Hedeker and Gibbons (1994); Crouch and Spiegelman (1990); Longford (1994); McCulloch (1994); Liu and Pierce (1994); Diggle, Liang, and Zeger (1994).

Nonlinear mixed models have important applications in pharmacokinetics, and Roe (1997) provides a wide-ranging comparison of many popular techniques. Yuh et al. (1994) provide an extensive bibliography on nonlinear mixed models and their use in pharmacokinetics.

PROC NLMIXED Compared with Other SAS Procedures and Macros

The models fit by PROC NLMIXED can be viewed as generalizations of the random coefficient models fit by the MIXED procedure. This generalization allows the random coefficients to enter the model nonlinearly, whereas in PROC MIXED they enter linearly. With PROC MIXED you can perform both maximum likelihood and restricted maximum likelihood (REML) estimation, whereas PROC NLMIXED implements only maximum likelihood. This is because the analog to the REML method in PROC NLMIXED would involve a high-dimensional integral over all of the fixed-effects parameters, and this integral is typically not available in closed form. Finally, PROC MIXED assumes the data to be normally distributed, whereas PROC NLMIXED enables you to analyze data that are normal, binomial, or Poisson or that have any likelihood programmable with SAS statements.

PROC NLMIXED does not implement the same estimation techniques available with the NLINMIX macro or the default estimation method of the GLIMMIX procedure. These are based on the estimation methods

of: Lindstrom and Bates (1990); Breslow and Clayton (1993); Wolfinger and O'Connell (1993), and they iteratively fit a set of generalized estimating equations (see Chapters 14 and 15 of Littell et al. 2006; Wolfinger 1997). In contrast, PROC NLMIXED directly maximizes an approximate integrated likelihood. This remark also applies to the SAS/IML macros MIXNLIN (Vonesh and Chinchilli 1997) and NLMEM (Galecki 1998).

The GLIMMIX procedure also fits mixed models for nonnormal data with nonlinearity in the conditional mean function. In contrast to the NLMIXED procedure, PROC GLIMMIX assumes that the model contains a linear predictor that links covariates to the conditional mean of the response. The NLMIXED procedure is designed to handle general conditional mean functions, whether they contain a linear component or not. As mentioned earlier, the GLIMMIX procedure by default estimates parameters in generalized linear mixed models by pseudo-likelihood techniques, whereas PROC NLMIXED by default performs maximum likelihood estimation by adaptive Gauss-Hermite quadrature. This estimation method is also available with the GLIMMIX procedure (METHOD=QUAD in the PROC GLIMMIX statement).

PROC NLMIXED has close ties with the NLP procedure in SAS/OR software. PROC NLMIXED uses a subset of the optimization code underlying PROC NLP and has many of the same optimization-based options. Also, the programming statement functionality used by PROC NLMIXED is the same as that used by PROC NLP and the MODEL procedure in SAS/ETS software.

Getting Started: NLMIXED Procedure

Nonlinear Growth Curves with Gaussian Data

As an introductory example, consider the orange tree data of Draper and Smith (1981). These data consist of seven measurements of the trunk circumference (in millimeters) on each of five orange trees. You can input these data into a SAS data set as follows:

```
data tree;
   input tree day y;
   datalines;
1  118   30
1  484   58
1  664   87

   ... more lines ...

5 1582  177
;
```

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

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

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

The statements to fit this nonlinear mixed model are as follows:

```
proc nlmixed data=tree;
   parms b1=190 b2=700 b3=350 s2u=1000 s2e=60;
   num = b1+u1;
   ex  = exp(-(day-b2)/b3);
   den = 1 + ex;
   model y ~ normal(num/den,s2e);
   random u1 ~ normal(0,s2u) subject=tree;
run;
```

The PROC NLMIXED statement invokes the procedure and inputs the tree data set. The PARMS statement identifies the unknown parameters and their starting values. Here there are three fixed-effects parameters (b1, b2, b3) and two variance components (s2u, s2e).

The next three statements are SAS programming statements specifying the logistic mixed model. A new variable u1 is included to identify the random effect. These statements are evaluated for every observation in the data set when the NLMIXED procedure computes the log likelihood function and its derivatives.

The MODEL statement defines the dependent variable and its conditional distribution given the random effects. Here a normal (Gaussian) conditional distribution is specified with mean num/den and variance s2e.

The RANDOM statement defines the single random effect to be u1, and specifies that it follow a normal distribution with mean 0 and variance s2u. The SUBJECT= argument in the RANDOM statement defines a variable indicating when the random effect obtains new realizations; in this case, it changes according to the values of the tree variable. PROC NLMIXED assumes that the input data set is clustered according to the levels of the tree variable; that is, all observations from the same tree occur sequentially in the input data set.

The output from this analysis is as follows.

Figure 88.1 Model Specifications

The NLMIXED Procedure

Specifications	
Data Set	WORK.TREE
Dependent Variable	y
Distribution for Dependent Variable	Normal
Random Effects	u1
Distribution for Random Effects	Normal
Subject Variable	tree
Optimization Technique	Dual Quasi-Newton
Integration Method	Adaptive Gaussian Quadrature

The "Specifications" table lists basic information about the nonlinear mixed model you have specified (Figure 88.1). Included are the input data set, the dependent and subject variables, the random effects, the relevant distributions, and the type of optimization. The "Dimensions" table lists various counts related to the model, including the number of observations, subjects, and parameters (Figure 88.2). These quantities are useful for checking that you have specified your data set and model correctly. Also listed is the number of quadrature points that PROC NLMIXED has selected based on the evaluation of the log likelihood at the starting values of the parameters. Here, only one quadrature point is necessary because the random-effect parameters u_{i1} enter the model linearly. (The Gauss-Hermite quadrature with a single quadrature point results in the Laplace approximation of the log likelihood.)

Figure 88.2 Dimensions Table for Growth Curve Model

Dimensions	
Observations Used	35
Observations Not Used	0
Total Observations	35
Subjects	5
Max Obs per Subject	7
Parameters	5
Quadrature Points	1

Figure 88.3 Starting Values of Parameter Estimates and Negative Log Likelihood

Initial Parameters					
b1	b2	b3	s2u	s2e	Negative Log Likelihood
190	700	350	1000	60	132.491787

The "Parameters" table lists the parameters to be estimated, their starting values, and the negative log likelihood evaluated at the starting values (Figure 88.3).

Figure 88.4 Iteration History for Growth Curve Model

Iteration History					
Iteration	Calls	Negative Log Likelihood	Difference	Maximum Gradient	Slope
1	8	131.6867	0.805045	0.010269	-0.63300
2	12	131.6447	0.042082	0.014783	-0.01820
3	16	131.6141	0.030583	0.009809	-0.02796
4	20	131.5725	0.041555	0.001186	-0.01344
5	22	131.5719	0.000627	0.000200	-0.00121
6	25	131.5719	5.549E-6	0.000092	-7.68E-6
7	28	131.5719	1.096E-6	6.097E-6	-1.29E-6

NOTE: GCONV convergence criterion satisfied.

The "Iteration History" table records the history of the minimization of the negative log likelihood (Figure 88.4). For each iteration of the quasi-Newton optimization, values are listed for the number of function calls, the value of the negative log likelihood, the difference from the previous iteration, the absolute value of the largest gradient, and the slope of the search direction. The note at the bottom of the table indicates that the algorithm has converged successfully according to the GCONV convergence criterion, a standard criterion computed using a quadratic form in the gradient and the inverse Hessian.

The final maximized value of the log likelihood as well as the information criterion of Akaike (AIC), its small sample bias corrected version (AICC), and the Bayesian information criterion (BIC) in the "smaller is better" form appear in the "Fit Statistics" table (Figure 88.5). These statistics can be used to compare different nonlinear mixed models.

Figure 88.5 Fit Statistics for Growth Curve Model

Fit Statistics	
-2 Log Likelihood	263.1
AIC (smaller is better)	273.1
AICC (smaller is better)	275.2
BIC (smaller is better)	271.2

Figure 88.6 Parameter Estimates at Convergence

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|	95% Confidence Limits		Gradient
b1	192.05	15.6473	4	12.27	0.0003	148.61	235.50	1.154E-6
b2	727.90	35.2474	4	20.65	<.0001	630.04	825.76	5.289E-6
b3	348.07	27.0793	4	12.85	0.0002	272.88	423.25	-6.1E-6
s2u	999.88	647.44	4	1.54	0.1974	-797.71	2797.46	-3.84E-6
s2e	61.5139	15.8832	4	3.87	0.0179	17.4150	105.61	2.892E-6

The maximum likelihood estimates of the five parameters and their approximate standard errors computed using the final Hessian matrix are displayed in the "Parameter Estimates" table (Figure 88.6). Approximate t-values and Wald-type confidence limits are also provided, with degrees of freedom equal to the number of subjects minus the number of random effects. You should interpret these statistics cautiously for variance parameters like s2u and s2e. The final column in the output shows the gradient vector at the optimization solution. Each element appears to be sufficiently small to indicate a stationary point.

Since the random-effect parameters u_{i1} enter the model linearly, you can obtain equivalent results by using the first-order method (specify METHOD=FIRO in the PROC NLMIXED statement).

Logistic-Normal Model with Binomial Data

This example analyzes the data from Beitler and Landis (1985), which represent results from a multi-center clinical trial investigating the effectiveness of two topical cream treatments (active drug, control) in curing an infection. For each of eight clinics, the number of trials and favorable cures are recorded for each treatment. The SAS data set is as follows.

```
data infection;
   input clinic t x n;
   datalines;
1 1 11 36
1 0 10 37
2 1 16 20
2 0 22 32
3 1 14 19
3 0  7 19
4 1  2 16
4 0  1 17
5 1  6 17
```

```
5 0  0 12
6 1  1 11
6 0  0 10
7 1  1  5
7 0  1  9
8 1  4  6
8 0  6  7
;
```

Suppose n_{ij} denotes the number of trials for the ith clinic and the jth treatment ($i = 1, \ldots, 8; j = 0, 1$), and x_{ij} denotes the corresponding number of favorable cures. Then a reasonable model for the preceding data is the following logistic model with random effects:

$$x_{ij}|u_i \sim \text{Binomial}(n_{ij}, p_{ij})$$

and

$$\eta_{ij} = \log\left(\frac{p_{ij}}{1-p_{ij}}\right) = \beta_0 + \beta_1 t_j + u_i$$

The notation t_j indicates the jth treatment, and the u_i are assumed to be iid $N(0, \sigma_u^2)$.

The PROC NLMIXED statements to fit this model are as follows:

```
proc nlmixed data=infection;
   parms beta0=-1 beta1=1 s2u=2;
   eta    = beta0 + beta1*t + u;
   expeta = exp(eta);
   p      = expeta/(1+expeta);
   model x ~ binomial(n,p);
   random u ~ normal(0,s2u) subject=clinic;
   predict eta out=eta;
   estimate '1/beta1' 1/beta1;
run;
```

The PROC NLMIXED statement invokes the procedure, and the PARMS statement defines the parameters and their starting values. The next three statements define p_{ij}, and the MODEL statement defines the conditional distribution of x_{ij} to be binomial. The RANDOM statement defines u to be the random effect with subjects defined by the clinic variable.

The PREDICT statement constructs predictions for each observation in the input data set. For this example, predictions of η_{ij} and approximate standard errors of prediction are output to a data set named eta. These predictions include empirical Bayes estimates of the random effects u_i.

The ESTIMATE statement requests an estimate of the reciprocal of β_1.

The output for this model is as follows.

Figure 88.7 Model Information and Dimensions for Logistic-Normal Model

The NLMIXED Procedure

Specifications	
Data Set	WORK.INFECTION
Dependent Variable	x
Distribution for Dependent Variable	Binomial
Random Effects	u
Distribution for Random Effects	Normal
Subject Variable	clinic
Optimization Technique	Dual Quasi-Newton
Integration Method	Adaptive Gaussian Quadrature

Dimensions	
Observations Used	16
Observations Not Used	0
Total Observations	16
Subjects	8
Max Obs per Subject	2
Parameters	3
Quadrature Points	5

The "Specifications" table provides basic information about the nonlinear mixed model (Figure 88.7). For example, the distribution of the response variable, conditional on normally distributed random effects, is binomial. The "Dimensions" table provides counts of various variables. You should check this table to make sure the data set and model have been entered properly. PROC NLMIXED selects five quadrature points to achieve the default accuracy in the likelihood calculations.

Figure 88.8 Starting Values of Parameter Estimates

Initial Parameters			
beta0	**beta1**	**s2u**	**Negative Log Likelihood**
-1	1	2	37.5945925

The "Parameters" table lists the starting point of the optimization and the negative log likelihood at the starting values (Figure 88.8).

Figure 88.9 Iteration History and Fit Statistics for Logistic-Normal Model

Iteration History

Iteration	Calls	Negative Log Likelihood	Difference	Maximum Gradient	Slope
1	4	37.3622692	0.232323	2.88208	-19.3762
2	6	37.1460375	0.216232	0.92193	-0.82852
3	9	37.0300936	0.115944	0.31590	-0.59175
4	11	37.0223017	0.007792	0.019060	-0.01615
5	13	37.0222472	0.000054	0.001743	-0.00011
6	16	37.0222466	6.57E-7	0.000091	-1.28E-6
7	19	37.0222466	5.38E-10	2.078E-6	-1.1E-9

NOTE: GCONV convergence criterion satisfied.

Fit Statistics

-2 Log Likelihood	74.0
AIC (smaller is better)	80.0
AICC (smaller is better)	82.0
BIC (smaller is better)	80.3

The "Iteration History" table indicates successful convergence in seven iterations (Figure 88.9). The "Fit Statistics" table lists some useful statistics based on the maximized value of the log likelihood.

Figure 88.10 Parameter Estimates for Logistic-Normal Model

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|	95% Confidence Limits		Gradient
beta0	-1.1974	0.5561	7	-2.15	0.0683	-2.5123	0.1175	-3.1E-7
beta1	0.7385	0.3004	7	2.46	0.0436	0.02806	1.4488	-2.08E-6
s2u	1.9591	1.1903	7	1.65	0.1438	-0.8555	4.7737	-2.48E-7

The "Parameter Estimates" table indicates marginal significance of the two fixed-effects parameters (Figure 88.10). The positive value of the estimate of β_1 indicates that the treatment significantly increases the chance of a favorable cure.

Figure 88.11 Table of Additional Estimates

Additional Estimates

Label	Estimate	Standard Error	DF	t Value	Pr > |t|	Alpha	Lower	Upper
1/beta1	1.3542	0.5509	7	2.46	0.0436	0.05	0.05146	2.6569

The "Additional Estimates" table displays results from the ESTIMATE statement (Figure 88.11). The estimate of $1/\beta_1$ equals $1/0.7385 = 1.3542$ and its standard error equals $0.3004/0.7385^2 = 0.5509$ by the delta method (Billingsley 1986; Cox 1998). Note that this particular approximation produces a t-statistic identical to that for the estimate of β_1. Not shown is the eta data set, which contains the original 16 observations and predictions of the η_{ij}.

Syntax: NLMIXED Procedure

The following statements are available in the NLMIXED procedure:

PROC NLMIXED < *options* > ;
  ARRAY *array-specification* ;
  BOUNDS *boundary-constraints* ;
  BY *variables* ;
  CMPTMODEL *required-options conditionally-required-options* < *options* > ;
  CONTRAST '*label*' *expression* < , *expression* > < *option* > ;
  ESTIMATE '*label*' *expression* < *options* > ;
  ID *names* ;
  MODEL *model-specification* ;
  PARMS *parameters-and-starting-values* ;
  PREDICT *expression* **OUT=***SAS-data-set* < *options* > ;
  RANDOM *random-effects-specification* ;
  REPLICATE *variable* ;
  Programming statements ;

The following sections provide a detailed description of each of these statements.

PROC NLMIXED Statement

PROC NLMIXED < *options* > ;

The PROC NLMIXED statement invokes the NLMIXED procedure. Table 88.1 summarizes the options available in the PROC NLMIXED statement.

Table 88.1 PROC NLMIXED Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data set
METHOD=	Specifies the integration method
NOSORTSUB	Requests that the unique SUBJECT= variable values not be used
NTHREADS=	Specifies the number of threads to use
Displayed Output Specifications	
ALPHA=	Specifies α for confidence limits
CORR	Requests the correlation matrix
COV	Requests the covariance matrix
DF=	Specifies the degrees of freedom for p-values and confidence limits
ECORR	Requests the correlation matrix of additional estimates
ECOV	Requests the covariance matrix of additional estimates
EDER	Requests derivatives of additional estimates
EMPIRICAL	Requests the empirical (“sandwich”) estimator of covariance matrix
HESS	Requests the Hessian matrix

Table 88.1 *continued*

Option	Description
ITDETAILS	Requests iteration details
START	Specifies the gradient at starting values
Debugging Output	
FLOW	Displays the model execution messages
LISTCODE	Displays compiled model program
LISTDEP	Produces a model dependency listing
LISTDER	Displays the model derivatives
LIST	Displays the model program and variables
TRACE	Displays detailed model execution messages
XREF	Displays the model cross references
Quadrature Options	
NOADSCALE	Requests no adaptive scaling
NOAD	Requests no adaptive centering
OUTQ=	Displays output data set
QFAC=	Specifies the search factor
QMAX=	Specifies the maximum points
QPOINTS=	Specifies the number of points
QSCALEFAC=	Specifies the scale factor
QTOL=	Specifies the tolerance
Empirical Bayes Options	
EBOPT	Requests comprehensive optimization
EBSSFRAC=	Specifies the step-shortening fraction
EBSSTOL=	Specifies the step-shortening tolerance
EBSTEPS=	Specifies the number of Newton steps
EBSUBSTEPS=	Specifies the number of substeps
EBTOL=	Specifies the convergence tolerance
EBZSTART	Requests zero as the starting values
OUTR=	Displays an output data set that contains empirical Bayes estimates of random effects and their approximate standard errors
Optimization Specifications	
HESCAL=	Specifies the type of Hessian scaling
INHESSIAN<=>	Specifies the start for approximated Hessian
LINESEARCH=	Specifies the line-search method
LSPRECISION=	Specifies the line-search precision
OPTCHECK<=>	Checks optimality in a neighborhood
RESTART=	Specifies the iteration number for update restart
TECHNIQUE=	Specifies the minimization technique
UPDATE=	Specifies the update technique
Derivatives Specifications	
DIAHES	Uses only the diagonal of Hessian
FDHESSIAN<=>	Specifies the finite-difference second derivatives

Table 88.1 *continued*

Option	Description
FD<=>	Specifies the finite-difference derivatives
Constraint Specifications	
LCDEACT=	Specifies the Lagrange multiplier tolerance for deactivating
LCEPSILON=	Specifies the range for active constraints
LCSINGULAR=	Specifies the tolerance for dependent constraints
Termination Criteria Specifications	
ABSCONV=	Specifies the absolute function convergence criterion
ABSFCONV=	Specifies the absolute function difference convergence criterion
ABSGCONV=	Specifies the absolute gradient convergence criterion
ABSXCONV=	Specifies the absolute parameter convergence criterion
FCONV=	Specifies the relative function convergence criterion
FCONV2=	Specifies another relative function convergence criterion
FDIGITS=	Specifies the number accurate digits in objective function
FSIZE=	Specifies the FSIZE parameter of the relative function and relative gradient termination criteria
GCONV=	Specifies the relative gradient convergence criterion
MAXFUNC=	Specifies the maximum number of function calls
MAXITER=	Specifies the maximum number of iterations
MAXTIME=	Specifies the upper limit seconds of CPU time
MINITER=	Specifies the minimum number of iterations
XCONV=	Specifies the relative parameter convergence criterion
XSIZE=	Used in XCONV criterion
Step Length Specifications	
DAMPSTEP<=>	Specifies the damped steps in line search
INSTEP=	Specifies the initial trust-region radius
MAXSTEP=	Specifies the maximum trust-region radius
Singularity Tolerances	
SINGCHOL=	Specifies the tolerance for Cholesky roots
SINGHESS=	Specifies the tolerance for Hessian
SINGSWEEP=	Specifies the tolerance for sweep
SINGVAR=	Specifies the tolerance for variances
Covariance Matrix Tolerances	
ASINGULAR=	Specifies the absolute singularity for inertia
CFACTOR=	Specifies the multiplication factor for COV matrix
COVSING=	Specifies the tolerance for singular COV matrix
G4=	Specifies the threshold for Moore-Penrose inverse
MSINGULAR=	Specifies the relative M singularity for inertia
VSINGULAR=	Specifies the relative V singularity for inertia

These options are described in alphabetical order. For a description of the mathematical notation used in the following sections, see the section "Modeling Assumptions and Notation" on page 7106.

ABSCONV=*r*

ABSTOL=*r*

specifies an absolute function convergence criterion. For minimization, termination requires $f(\boldsymbol{\theta}^{(k)}) \leq r$. The default value of r is the negative square root of the largest double-precision value, which serves only as a protection against overflows.

ABSFCONV=*r***<** **[***n***]** **>**

ABSFTOL=*r***<** **[***n***]** **>**

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

$$|f(\boldsymbol{\theta}^{(k-1)}) - f(\boldsymbol{\theta}^{(k)})| \leq r$$

The same formula is used for the NMSIMP technique, but $\boldsymbol{\theta}^{(k)}$ is defined as the vertex with the lowest function value, and $\boldsymbol{\theta}^{(k-1)}$ is defined as the vertex with 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:

$$\max_j |g_j(\boldsymbol{\theta}^{(k)})| \leq r$$

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. If you specify more than one RANDOM statement, the default value is r = 1E–3.

ABSXCONV=*r***<** **[***n***]** **>**

ABSXTOL=*r***<** **[***n***]** **>**

specifies an absolute parameter convergence criterion. For all techniques except NMSIMP, termination requires a small Euclidean distance between successive parameter vectors,

$$\| \boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)} \|_2 \leq r$$

For the NMSIMP technique, termination requires either a small length $\alpha^{(k)}$ of the vertices of a restart simplex,

$$\alpha^{(k)} \leq r$$

or a small simplex size,

$$\delta^{(k)} \leq r$$

where the simplex size $\delta^{(k)}$ is defined as the L1 distance from the simplex vertex $\boldsymbol{\xi}^{(k)}$ with the smallest function value to the other n simplex points $\boldsymbol{\theta}_l^{(k)} \neq \boldsymbol{\xi}^{(k)}$:

$$\delta^{(k)} = \sum_{\boldsymbol{\theta}_l \neq y} \| \boldsymbol{\theta}_l^{(k)} - \boldsymbol{\xi}^{(k)} \|_1$$

The default is r = 1E–8 for the NMSIMP technique and $r = 0$ otherwise. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

ALPHA=α

specifies the alpha level to be used in computing confidence limits. The default value is 0.05.

ASINGULAR=*r*

ASING=*r*

specifies an absolute singularity criterion for the computation of the inertia (number of positive, negative, and zero eigenvalues) of the Hessian and its projected forms. The default value is the square root of the smallest positive double-precision value.

CFACTOR=*f*

specifies a multiplication factor *f* for the estimated covariance matrix of the parameter estimates.

CORR

requests the approximate correlation matrix for the parameter estimates.

COV

requests the approximate covariance matrix for the parameter estimates.

COVSING=*r* > 0

specifies a nonnegative threshold that determines whether the eigenvalues of a singular Hessian matrix are considered to be zero.

DAMPSTEP<=*r***>**

DS<=*r***>**

specifies that the initial step-size value $\alpha^{(0)}$ for each line search (used by the QUANEW, CONGRA, or NEWRAP technique) cannot be larger than *r* times the step-size value used in the former iteration. If you specify the DAMPSTEP option without factor *r*, the default value is *r* = 2. The DAMPSTEP=*r* option can prevent the line-search algorithm from repeatedly stepping into regions where some objective functions are difficult to compute or where they could lead to floating-point overflows during the computation of objective functions and their derivatives. The DAMPSTEP=*r* option can save time-costly function calls that result in very small step sizes α. For more details on setting the start values of each line search, see the section "Restricting the Step Length" on page 7134.

DATA=*SAS-data-set*

specifies the input data set. Observations in this data set are used to compute the log likelihood function that you specify with PROC NLMIXED statements.

NOTE: In SAS/STAT 12.3 and previous releases, if you are using a RANDOM statement, the input data set must be clustered according to the SUBJECT= variable. One easy way to accomplish this is to sort your data by the SUBJECT= variable before calling the NLMIXED procedure. PROC NLMIXED does not sort the input data set for you.

DF=*d*

specifies the degrees of freedom to be used in computing *p* values and confidence limits. PROC NLMIXED calculates the default degrees of freedom as follows:

- When there is no RANDOM statement in the model, the default value is the number of observations.
- When only one RANDOM statement is specified, the default value is the number of subjects minus the number of random effects for random-effects models.

- When multiple RANDOM statements are specified, the default degrees of freedom is the number of subjects in the lowest nested level minus the total number of random effects. For example, if the highest level of hierarchy is specified by SUBJECT=S1 and the next level of hierarchy (nested within S1) is specified by SUBJECT=S2(S1), then the degrees of freedom is computed as the total number of subjects from S2(S1) minus the total number of random-effects variables in the model.

If the degrees of freedom computation leads to a nonpositive value, then the default value is the total number of observations.

DIAHES
specifies that only the diagonal of the Hessian be used.

EBOPT
requests that a more comprehensive optimization be carried out if the default empirical Bayes optimization fails to converge. If you specify more than one RANDOM statement, this option is ignored.

EBSSFRAC=*r* > 0
specifies the step-shortening fraction to be used while computing empirical Bayes estimates of the random effects. The default value is 0.8. If you specify more than one RANDOM statement, this option is ignored.

EBSSTOL=*r* ≥ 0
specifies the objective function tolerance for determining the cessation of step-shortening while computing empirical Bayes estimates of the random effects. The default value is r = 1E–8. If you specify more than one RANDOM statement, this option is ignored.

EBSTEPS=*n* ≥ 0
specifies the maximum number of Newton steps for computing empirical Bayes estimates of random effects. The default value is $n = 50$. If you specify more than one RANDOM statement, this option is ignored.

EBSUBSTEPS=*n* ≥ 0
specifies the maximum number of step-shortenings for computing empirical Bayes estimates of random effects. The default value is $n = 20$. If you specify more than one RANDOM statement, this option is ignored.

EBTOL=*r* ≥ 0
specifies the convergence tolerance for empirical Bayes estimation. The default value is $r = \epsilon \mathrm{E}4$, where ϵ is the machine precision. This default value equals approximately 1E–12 on most machines. If you specify more than one RANDOM statement, this option is ignored.

EBZSTART
requests that a zero be used as starting values during empirical Bayes estimation. By default, the starting values are set equal to the estimates from the previous iteration (or zero for the first iteration).

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

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

EDER
requests the derivatives of all expressions specified in ESTIMATE statements with respect to each of the model parameters.

EMPIRICAL
requests that the covariance matrix of the parameter estimates be computed as a likelihood-based empirical ("sandwich") estimator (White 1982). If $f(\boldsymbol{\theta}) = -\log\{m(\boldsymbol{\theta})\}$ is the objective function for the optimization and $m(\boldsymbol{\theta})$ denotes the marginal log likelihood (see the section "Modeling Assumptions and Notation" on page 7106 for notation and further definitions) the empirical estimator is computed as

$$\mathbf{H}(\hat{\boldsymbol{\theta}})^{-1}\left(\sum_{i=1}^{s}\mathbf{g}_i(\hat{\boldsymbol{\theta}})\mathbf{g}_i(\hat{\boldsymbol{\theta}})'\right)\mathbf{H}(\hat{\boldsymbol{\theta}})^{-1}$$

where $\mathbf{H}$ is the second derivative matrix of f and $\mathbf{g}_i$ is the first derivative of the contribution to f by the ith subject. If you choose the EMPIRICAL option, this estimator of the covariance matrix of the parameter estimates replaces the model-based estimator $\mathbf{H}(\hat{\boldsymbol{\theta}})^{-1}$ in subsequent calculations. You can output the subject-specific gradients $\mathbf{g}_i$ to a SAS data set with the SUBGRADIENT option in the PROC NLMIXED statement.

The EMPIRICAL option requires the presence of a RANDOM statement and is available for METHOD=GAUSS and METHOD=ISAMP only.

If you specify more than one RANDOM statement, this option is ignored.

FCONV=*r*<[*n*]>
FTOL=*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,

$$\frac{|f(\boldsymbol{\theta}^{(k)}) - f(\boldsymbol{\theta}^{(k-1)})|}{\max(|f(\boldsymbol{\theta}^{(k-1)})|, \text{FSIZE})} \leq r$$

where FSIZE is defined by the FSIZE= option. The same formula is used for the NMSIMP technique, but $\boldsymbol{\theta}^{(k)}$ is defined as the vertex with the lowest function value, and $\boldsymbol{\theta}^{(k-1)}$ is defined as the vertex with the highest function value in the simplex. The default is $r = 10^{-\text{FDIGITS}}$, where FDIGITS is the value of the FDIGITS= option. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

FCONV2=*r*<[*n*]>
FTOL2=*r*<[*n*]>
specifies another function convergence criterion. For all techniques except NMSIMP, termination requires a small predicted reduction

$$df^{(k)} \approx f(\boldsymbol{\theta}^{(k)}) - f(\boldsymbol{\theta}^{(k)} + \mathbf{s}^{(k)})$$

of the objective function. The predicted reduction

$$\begin{aligned} df^{(k)} &= -\mathbf{g}^{(k)\prime}\mathbf{s}^{(k)} - \frac{1}{2}\mathbf{s}^{(k)\prime}\mathbf{H}^{(k)}\mathbf{s}^{(k)} \\ &= -\frac{1}{2}\mathbf{s}^{(k)\prime}\mathbf{g}^{(k)} \\ &\leq r \end{aligned}$$

is computed by approximating the objective function f by the first two terms of the Taylor series and substituting the Newton step:

$$\mathbf{s}^{(k)} = -[\mathbf{H}^{(k)}]^{-1}\mathbf{g}^{(k)}$$

For the NMSIMP technique, termination requires a small standard deviation of the function values of the $n+1$ simplex vertices $\boldsymbol{\theta}_l^{(k)}$, $l = 0, \ldots, n$,

$$\sqrt{\frac{1}{n+1}\sum_l \left[f(\boldsymbol{\theta}_l^{(k)}) - \overline{f}(\boldsymbol{\theta}^{(k)}) \right]^2} \leq r$$

where $\overline{f}(\boldsymbol{\theta}^{(k)}) = \frac{1}{n+1}\sum_l f(\boldsymbol{\theta}_l^{(k)})$. If there are n_{act} boundary constraints active at $\boldsymbol{\theta}^{(k)}$, the mean and standard deviation are computed only for the $n+1-n_{act}$ unconstrained vertices. The default value is r = 1E–6 for the NMSIMP technique and $r = 0$ otherwise. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

FD < = FORWARD | CENTRAL | *r* >
specifies that all derivatives be computed using finite difference approximations. The following specifications are permitted:

FD is equivalent to FD=100.

FD=CENTRAL uses central differences.

FD=FORWARD uses forward differences.

FD=*r* uses central differences for the initial and final evaluations of the gradient and for the Hessian. During iteration, start with forward differences and switch to a corresponding central-difference formula during the iteration process when one of the following two criteria is satisfied:

- The absolute maximum gradient element is less than or equal to r times the ABSGCONV= threshold.
- The normalized predicted function reduction (see the GTOL option) is less than or equal to $\max(1\mathrm{E}-6,\ r \times \mathrm{GTOL})$. The 1E–6 ensures that the switch is done, even if you set the GTOL threshold to zero.

Note that the FD and FDHESSIAN options cannot apply at the same time. The FDHESSIAN option is ignored when only first-order derivatives are used. See the section "Finite-Difference Approximations of Derivatives" on page 7129 for more information.

FDHESSIAN< =FORWARD | CENTRAL >
FDHES< =FORWARD | CENTRAL >
FDH< =FORWARD | CENTRAL >

specifies that second-order derivatives be computed using finite difference approximations based on evaluations of the gradients.

FDHESSIAN=FORWARD uses forward differences.

FDHESSIAN=CENTRAL uses central differences.

FDHESSIAN uses forward differences for the Hessian except for the initial and final output.

Note that the FD and FDHESSIAN options cannot apply at the same time. See the section "Finite-Difference Approximations of Derivatives" on page 7129 for more information.

FDIGITS=*r*

specifies the number of accurate digits in evaluations of the objective function. Fractional values such as FDIGITS=4.7 are allowed. The default value is $r = -\log_{10}\epsilon$, where ϵ is the machine precision. The value of r is used to compute the interval size h for the computation of finite-difference approximations of the derivatives of the objective function and for the default value of the FCONV= option. If you specify more than one RANDOM statement, the default value is $r = 0.5(-\log_{10}\epsilon)$.

FLOW

displays a message for each statement in the model program as it is executed. This debugging option is very rarely needed and produces voluminous output.

FSIZE=*r*

specifies the FSIZE parameter of the relative function and relative gradient termination criteria. The default value is $r = 0$. For more information, see the FCONV= and GCONV= options.

G4=*n* > 0

specifies a dimension to determine the type of generalized inverse to use when the approximate covariance matrix of the parameter estimates is singular. The default value of n is 60. See the section "Covariance Matrix" on page 7138 for more information.

GCONV=*r*< [*n*] >
GTOL=*r*< [*n*] >

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

$$\frac{\mathbf{g}(\boldsymbol{\theta}^{(k)})'[\mathbf{H}^{(k)}]^{-1}\mathbf{g}(\boldsymbol{\theta}^{(k)})}{\max(|f(\boldsymbol{\theta}^{(k)})|, \mathrm{FSIZE})} \leq r$$

where FSIZE is defined by the FSIZE= option. For the CONGRA technique (where a reliable Hessian estimate H is not available), the following criterion is used:

$$\frac{\| \mathbf{g}(\boldsymbol{\theta}^{(k)}) \|_2^2 \quad \| \mathbf{s}(\boldsymbol{\theta}^{(k)}) \|_2}{\| \mathbf{g}(\boldsymbol{\theta}^{(k)}) - \mathbf{g}(\boldsymbol{\theta}^{(k-1)}) \|_2 \max(|f(\boldsymbol{\theta}^{(k)})|, \mathrm{FSIZE})} \leq r$$

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. If you specify more than one RANDOM statement, the default value is r = 1E–6.

HESCAL= 0 | 1 | 2 | 3

HS=0 | 1 | 2 | 3

specifies the scaling version of the Hessian matrix used in NRRIDG, TRUREG, NEWRAP, or DBLDOG optimization.

If HS is not equal to 0, the first iteration and each restart iteration sets the diagonal scaling matrix $\mathbf{D}^{(0)} = \mathrm{diag}(d_i^{(0)})$:

$$d_i^{(0)} = \sqrt{\max(|H_{i,i}^{(0)}|, \epsilon)}$$

where $H_{i,i}^{(0)}$ are the diagonal elements of the Hessian. In every other iteration, the diagonal scaling matrix $\mathbf{D}^{(0)} = \mathrm{diag}(d_i^{(0)})$ is updated depending on the HS option:

0 specifies that no scaling is done.

1 specifies the Moré (1978) scaling update:

$$d_i^{(k+1)} = \max\left[d_i^{(k)}, \sqrt{\max(|H_{i,i}^{(k)}|, \epsilon)}\right]$$

2 specifies the Dennis, Gay, and Welsch (1981) scaling update:

$$d_i^{(k+1)} = \max\left[0.6 * d_i^{(k)}, \sqrt{\max(|H_{i,i}^{(k)}|, \epsilon)}\right]$$

3 specifies that d_i is reset in each iteration:

$$d_i^{(k+1)} = \sqrt{\max(|H_{i,i}^{(k)}|, \epsilon)}$$

In each scaling update, ϵ is the relative machine precision. The default value is HS=0. Scaling of the Hessian can be time-consuming in the case where general linear constraints are active.

HESS

requests the display of the final Hessian matrix after optimization. If you also specify the START option, then the Hessian at the starting values is also printed.

INHESSIAN< =*r* >

INHESS< =*r* >

specifies how the initial estimate of the approximate Hessian is defined for the quasi-Newton techniques QUANEW and DBLDOG. There are two alternatives:

- If you do not use the *r* specification, the initial estimate of the approximate Hessian is set to the Hessian at $\boldsymbol{\theta}^{(0)}$.
- If you do use the *r* specification, the initial estimate of the approximate Hessian is set to the multiple of the identity matrix, $r\mathbf{I}$.

By default, if you do not specify the option INHESSIAN=*r*, the initial estimate of the approximate Hessian is set to the multiple of the identity matrix $r\mathbf{I}$, where the scalar r is computed from the magnitude of the initial gradient.

INSTEP=*r*

reduces the length of the first trial step during the line search of the first iterations. For highly nonlinear objective functions, such as the EXP function, the default initial radius of the trust-region algorithm TRUREG or DBLDOG or the default step length of the line-search algorithms can result in arithmetic overflows. If this occurs, you should specify decreasing values of $0 < r < 1$ such as INSTEP=1E–1, INSTEP=1E–2, INSTEP=1E–4, and so on, until the iteration starts successfully.

- For trust-region algorithms (TRUREG, DBLDOG), the INSTEP= option specifies a factor $r > 0$ for the initial radius $\Delta^{(0)}$ of the trust region. The default initial trust-region radius is the length of the scaled gradient. This step corresponds to the default radius factor of $r = 1$.
- For line-search algorithms (NEWRAP, CONGRA, QUANEW), the INSTEP= option specifies an upper bound for the initial step length for the line search during the first five iterations. The default initial step length is $r = 1$.
- For the Nelder-Mead simplex algorithm, using TECH=NMSIMP, the INSTEP=*r* option defines the size of the start simplex.

For more details, see the section "Computational Problems" on page 7135.

ITDETAILS

requests a more complete iteration history, including the current values of the parameter estimates, their gradients, and additional optimization statistics. For further details, see the section "Iterations" on page 7141.

LCDEACT=*r*

LCD=*r*

specifies a threshold r for the Lagrange multiplier that determines whether an active inequality constraint remains active or can be deactivated. During minimization, an active inequality constraint can be deactivated only if its Lagrange multiplier is less than the threshold value $r < 0$. The default value is

$$r = -\min(0.01, \max(0.1 \times \mathrm{ABSGCONV}, 0.001 \times \mathrm{gmax}^{(k)}))$$

where ABSGCONV is the value of the absolute gradient criterion, and $\mathrm{gmax}^{(k)}$ is the maximum absolute element of the (projected) gradient $\mathbf{g}^{(k)}$ or $\mathbf{Z}'\mathbf{g}^{(k)}$. (See the section "Active Set Methods" for a definition of $\mathbf{Z}$.)

LCEPSILON=*r* > 0

LCEPS=*r* > 0

LCE=*r* > 0

specifies the range for active and violated boundary constraints. The default value is r = 1E–8. During the optimization process, the introduction of rounding errors can force PROC NLMIXED to increase the value of r by a factor of $10, 100, \ldots$. If this happens, it is indicated by a message displayed in the log.

LCSINGULAR=*r* > 0

LCSING=*r* > 0

LCS=*r* > 0

specifies a criterion *r*, used in the update of the QR decomposition, that determines whether an active constraint is linearly dependent on a set of other active constraints. The default value is *r* = 1E–8. The larger *r* becomes, the more the active constraints are recognized as being linearly dependent. If the value of *r* is larger than 0.1, it is reset to 0.1.

LINESEARCH=*i*

LIS=*i*

specifies the line-search method for the CONGRA, QUANEW, and NEWRAP optimization techniques. See Fletcher (1987) for an introduction to line-search techniques. The value of *i* can be $1, \ldots, 8$. For CONGRA, QUANEW and NEWRAP, the default value is *i* = 2.

1 specifies a line-search method that needs the same number of function and gradient calls for cubic interpolation and cubic extrapolation; this method is similar to one used by the Harwell subroutine library.

2 specifies a line-search method that needs more function than gradient calls for quadratic and cubic interpolation and cubic extrapolation; this method is implemented as shown in Fletcher (1987) and can be modified to an exact line search by using the LSPRECISION= option.

3 specifies a line-search method that needs the same number of function and gradient calls for cubic interpolation and cubic extrapolation; this method is implemented as shown in Fletcher (1987) and can be modified to an exact line search by using the LSPRECISION= option.

4 specifies a line-search method that needs the same number of function and gradient calls for stepwise extrapolation and cubic interpolation.

5 specifies a line-search method that is a modified version of LIS=4.

6 specifies golden section line search (Polak 1971), which uses only function values for linear approximation.

7 specifies bisection line search (Polak 1971), which uses only function values for linear approximation.

8 specifies the Armijo line-search technique (Polak 1971), which uses only function values for linear approximation.

LIST

displays the model program and variable lists. The LIST option is a debugging feature and is not normally needed.

LISTCODE

displays the derivative tables and the compiled 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 program, the variables that depend on it and on which it depends. The LISTDEP option is a debugging feature and is not normally needed.

LISTDER

displays a table of derivatives. This table lists each nonzero derivative computed for the problem. The LISTDER option is a debugging feature and is not normally needed.

LOGNOTE<=*n*>

writes periodic notes to the log that describe the current status of computations. It is designed for use with analyses requiring extensive CPU resources. The optional integer value *n* specifies the desired level of reporting detail. The default is $n = 1$. Choosing $n = 2$ adds information about the objective function values at the end of each iteration. The most detail is obtained with $n = 3$, which also reports the results of function evaluations within iterations.

LSPRECISION=*r*

LSP=*r*

specifies the degree of accuracy that should be obtained by the line-search algorithms LIS=2 and LIS=3. Usually an imprecise line search is inexpensive and successful. For more difficult optimization problems, a more precise and expensive line search might be necessary (Fletcher 1987). The second line-search method (which is the default for the NEWRAP, QUANEW, and CONGRA techniques) and the third line-search method approach exact line search for small LSPRECISION= values. If you have numerical problems, you should try to decrease the LSPRECISION= value to obtain a more precise line search. The default values are shown in the following table.

TECH=	UPDATE=	LSP default
QUANEW	DBFGS, BFGS	$r = 0.4$
QUANEW	DDFP, DFP	$r = 0.06$
CONGRA	all	$r = 0.1$
NEWRAP	no update	$r = 0.9$

For more details, see Fletcher (1987).

MAXFUNC=*i*

MAXFU=*i*

specifies the maximum number (i) of function calls in the optimization process. The default value is $\max(10 \times j,\ n)$, where j is the value of the MAXITER= option and n is shown in the following table.

TECH=	n
TRUREG, NRRIDG, NEWRAP	125
QUANEW, DBLDOG	500
CONGRA	1000
NMSIMP	3000

Note that the optimization can terminate only after completing a full iteration. Therefore, the number of function calls that is actually performed can exceed the number that is specified by the MAXFUNC= option.

MAXITER=*j*

MAXIT=*j*

specifies the maximum number (*j*) of iterations in the optimization process. The default values are shown in the following table.

TECH=	*j*
TRUREG, NRRIDG, NEWRAP	50
QUANEW, DBLDOG	200
CONGRA	400
NMSIMP	1000

These default values are also valid when *j* is specified as a missing value.

MAXSTEP=*r*< [*n*] >

specifies an upper bound for the step length of the line-search algorithms during the first *n* iterations. By default, *r* is the largest double-precision value and *n* is the largest integer available. Setting this option can improve the speed of convergence for the CONGRA, QUANEW, and NEWRAP techniques.

MAXTIME=*r*

specifies an upper limit of *r* seconds of CPU time for the optimization process. The time checked only at the end of each iteration. Therefore, the actual run time might be longer than the specified time. By default, CPU time is not limited. The actual running time includes the rest of the time needed to finish the iteration and the time needed to generate the output of the results.

METHOD=*value*

specifies the method for approximating the integral of the likelihood over the random effects. Valid values are as follows:

FIRO

specifies the first-order method of Beal and Sheiner (1982). When using METHOD=FIRO, you must specify the NORMAL distribution in the MODEL statement and you must also specify a RANDOM statement.

GAUSS

specifies adaptive Gauss-Hermite quadrature (Pinheiro and Bates 1995). You can prevent the adaptation with the NOAD option or prevent adaptive scaling with the NOADSCALE option. This is the default integration method.

HARDY

specifies Hardy quadrature based on an adaptive trapezoidal rule. This method is available only for one-dimensional integrals; that is, you must specify only one random effect.

ISAMP

specifies adaptive importance sampling (Pinheiro and Bates 1995). You can prevent the adaptation with the NOAD option or prevent adaptive scaling with the NOADSCALE option. You can use the SEED= option to specify a starting seed for the random number generation used in the importance sampling. If you do not specify a seed, or if you specify a value less than or equal to zero, the seed is generated from reading the time of day from the computer clock.

MINITER=*i*

MINIT=*i*

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.

MSINGULAR=*r* > 0

MSING=*r* > 0

specifies a relative singularity criterion for the computation of the inertia (number of positive, negative, and zero eigenvalues) of the Hessian and its projected forms. The default value is 1E–12 if you do not specify the SINGHESS= option; otherwise, the default value is $\max(10\epsilon, (1\mathrm{E}-4) \times \mathrm{SINGHESS})$. See the section "Covariance Matrix" on page 7138 for more information.

NOAD

requests that the Gaussian quadrature be nonadaptive; that is, the quadrature points are centered at zero for each of the random effects and the current random-effects variance matrix is used as the scale matrix.

NOADSCALE

requests nonadaptive scaling for adaptive Gaussian quadrature; that is, the quadrature points are centered at the empirical Bayes estimates for the random effects, but the current random-effects variance matrix is used as the scale matrix. By default, the observed Hessian from the current empirical Bayes estimates is used as the scale matrix.

NOSORTSUB

requests that the data be processed sequentially and forms a new subject whenever the value of the SUBJECT= variable changes from the previous observation. This option enables PROC NLMIXED to use the behavior from SAS/STAT 12.3 and previous releases, in which the clusters are constructed by sequential processing. By default, starting with SAS/STAT 13.1, PROC NLMIXED constructs the clusters by using each of the unique SUBJECT= variable values, whether the input data set is sorted or not.

If you specify more than one RANDOM statement, this option is ignored.

NTHREADS=*n* > -2

specifies the number of threads to use for the log-likelihood calculation that uses the Gaussian quadrature method. When calculating the approximated marginal log likelihood by using the Gaussian quadrature method, PROC NLMIXED allocates data to different threads and calculates the objective function by accumulating values from each thread. NTHREADS=–1 sets the number of available threads to the number of hyperthreaded cores available on the system. By default, NTHREADS=1. This option is valid only when METHOD=GAUSS. Starting with SAS/STAT 14.2, this option has been extended to support models that include more than one RANDOM statement. In releases before SAS/STAT 14.2, this option is ignored if you specify more than one RANDOM statement.

OPTCHECK<=*r* > 0**>**
computes the function values $f(\boldsymbol{\theta}_l)$ of a grid of points $\boldsymbol{\theta}_l$ in a ball of radius of *r* about $\boldsymbol{\theta}^*$. If you specify the OPTCHECK option without factor *r*, the default value is *r* = 0.1 at the starting point and *r* = 0.01 at the terminating point. If a point $\boldsymbol{\theta}_l^*$ is found that has a better function value than $f(\boldsymbol{\theta}^*)$, then optimization is restarted at $\boldsymbol{\theta}_l^*$.

OUTQ=*SAS-data-set*
specifies an output data set that contains the quadrature points used for numerical integration.

OUTR=*SAS-data-set*
specifies an output data set that contains empirical Bayes estimates of the random effects of all hierarchies and their approximate standard errors.

QFAC=*r* > 0
specifies the additive factor used to adaptively search for the number of quadrature points. For METHOD=GAUSS, the search sequence is 1, 3, 5, 7, 9, 11, 11 + *r*, 11 + 2*r*, . . . , where the default value of *r* is 10. For METHOD=ISAMP, the search sequence is 10, 10 + *r*, 10 + 2*r*, . . . , where the default value of *r* is 50.

QMAX=*r* > 0
specifies the maximum number of quadrature points permitted before the adaptive search is aborted. The default values are 31 for adaptive Gaussian quadrature, 61 for nonadaptive Gaussian quadrature, 160 for adaptive importance sampling, and 310 for nonadaptive importance sampling.

QPOINTS=*n* > 0
specifies the number of quadrature points to be used during evaluation of integrals. For METHOD=GAUSS, *n* equals the number of points used in each dimension of the random effects, resulting in a total of n^r points, where *r* is the number of dimensions. For METHOD=ISAMP, *n* specifies the total number of quadrature points regardless of the dimension of the random effects. By default, the number of quadrature points is selected adaptively, and this option disables the adaptive search.

QSCALEFAC=*r* > 0
specifies a multiplier for the scale matrix used during quadrature calculations. The default value is 1.0.

QTOL=*r* > 0
specifies the tolerance used to adaptively select the number of quadrature points. When the relative difference between two successive likelihood calculations is less than *r*, then the search terminates and the lesser number of quadrature points is used during the subsequent optimization process. The default value is 1E–4.

RESTART=*i* > 0
REST=*i* > 0
specifies that the QUANEW or CONGRA algorithm is restarted with a steepest descent/ascent search direction after, at most, *i* iterations. Default values are as follows:

- CONGRA: UPDATE=PB: restart is performed automatically, *i* is not used.
- CONGRA: UPDATE≠PB: $i = \min(10n, 80)$, where *n* is the number of parameters.
- QUANEW: *i* is the largest integer available.

SEED=*i*

specifies the random number seed for METHOD=ISAMP. If you do not specify a seed, or if you specify a value less than or equal to zero, the seed is generated from reading the time of day from the computer clock. The value must be less than $2^{31} - 1$.

SINGCHOL=*r* > 0

specifies the singularity criterion *r* for Cholesky roots of the random-effects variance matrix and scale matrix for adaptive Gaussian quadrature. The default value is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

SINGHESS=*r* > 0

specifies the singularity criterion *r* for the inversion of the Hessian matrix. The default value is 1E–8. See the ASINGULAR, MSINGULAR=, and VSINGULAR= options for more information.

SINGSWEEP=*r* > 0

specifies the singularity criterion *r* for inverting the variance matrix in the first-order method and the empirical Bayes Hessian matrix. The default value is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

SINGVAR=*r* > 0

specifies the singularity criterion *r* below which statistical variances are considered to equal zero. The default value is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

START

requests that the gradient of the log likelihood at the starting values be displayed. If you also specify the HESS option, then the starting Hessian is displayed as well.

SUBGRADIENT=*SAS-data-set*

SUBGRAD=*SAS-data-set*

specifies a SAS data set that contains subgradients. In models that use the RANDOM statement, the data set contains the subject-specific gradients of the integrated, marginal log likelihood with respect to all parameters. The sum of the subject-specific gradients equals the gradient that is reported in the "Parameter Estimates" table. The data set contains a variable that identifies the subjects.

In models that do not use the RANDOM statement, the data set contains the observation-wise gradient. The variable identifying the SUBJECT= is then replaced with the Observation. This observation counter includes observations not used in the analysis and is reset in each BY group.

Saving disaggregated gradient information by specifying the SUBGRADIENT= option requires that you also specify METHOD=GAUSS or METHOD=ISAMP.

If you specify more than one RANDOM statement, this option is ignored.

TECHNIQUE=*value*

TECH=*value*

specifies the optimization technique. By default, TECH = QUANEW. Valid values are as follows:

- **CONGRA**
  performs a conjugate-gradient optimization, which can be more precisely specified with the UPDATE= option and modified with the LINESEARCH= option. When you specify this option, UPDATE=PB by default.

- **DBLDOG**
  performs a version of double-dogleg optimization, which can be more precisely specified with the UPDATE= option. When you specify this option, UPDATE=DBFGS by default.
- **NMSIMP**
  performs a Nelder-Mead simplex optimization.
- **NONE**
  does not perform any optimization. This option can be used as follows:
  - to perform a grid search without optimization
  - to compute estimates and predictions that cannot be obtained efficiently with any of the optimization techniques
- **NEWRAP**
  performs a Newton-Raphson optimization combining a line-search algorithm with ridging. The line-search algorithm LIS=2 is the default method.
- **NRRIDG**
  performs a Newton-Raphson optimization with ridging.
- **QUANEW**
  performs a quasi-Newton optimization, which can be defined more precisely with the UPDATE= option and modified with the LINESEARCH= option. This is the default estimation method.
- **TRUREG**
  performs a trust region optimization.

TRACE
displays the result of each operation in each statement in the model program as it is executed. This debugging option is very rarely needed, and it produces voluminous output.

UPDATE=*method*
UPD=*method*
specifies the update method for the quasi-Newton, double-dogleg, or conjugate-gradient optimization technique. Not every update method can be used with each optimizer. See the section "Optimization Algorithms" on page 7124 for more information.

Valid methods are as follows:

- **BFGS**
  performs the original Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update of the inverse Hessian matrix.
- **DBFGS**
  performs the dual BFGS update of the Cholesky factor of the Hessian matrix. This is the default update method.
- **DDFP**
  performs the dual Davidon, Fletcher, and Powell (DFP) update of the Cholesky factor of the Hessian matrix.
- **DFP**
  performs the original DFP update of the inverse Hessian matrix.
- **PB**
  performs the automatic restart update method of Powell (1977) and Beale (1972).

- **FR**
  performs the Fletcher-Reeves update (Fletcher 1987).
- **PR**
  performs the Polak-Ribiere update (Fletcher 1987).
- **CD**
  performs a conjugate-descent update of Fletcher (1987).

VSINGULAR=*r* > 0

VSING=*r* > 0

specifies a relative singularity criterion for the computation of the inertia (number of positive, negative, and zero eigenvalues) of the Hessian and its projected forms. The default value is *r* = 1E–8 if the SINGHESS= option is not specified, and it is the value of SINGHESS= option otherwise. See the section "Covariance Matrix" on page 7138 for more information.

XCONV=*r* **< [***n***] >**

XTOL=*r* **< [***n***] >**

specifies the relative parameter convergence criterion. For all techniques except NMSIMP, termination requires a small relative parameter change in subsequent iterations:

$$\frac{\max_j |\theta_j^{(k)} - \theta_j^{(k-1)}|}{\max(|\theta_j^{(k)}|, |\theta_j^{(k-1)}|, \text{XSIZE})} \leq r$$

For the NMSIMP technique, the same formula is used, but $\boldsymbol{\theta}^{(k)}$ is defined as the vertex with the lowest function value and $\boldsymbol{\theta}^{(k-1)}$ is defined as the vertex with the highest function value in the simplex.

The default value is *r* = 1E–8 for the NMSIMP technique and *r* = 0 otherwise. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

XREF

displays a cross-reference of the variables in the program showing where each variable is referenced or given a value. The XREF listing does not include derivative variables. This option is a debugging feature and is not normally needed.

XSIZE=*r* > 0

specifies the XSIZE parameter of the relative parameter termination criterion. The default value is *r* = 0. For more details, see the XCONV= option.

ARRAY Statement

ARRAY *arrayname* **[** *dimensions* **]** < **$** > < *variables-and-constants* > **;**

The ARRAY statement is similar to, but not exactly the same as, the ARRAY statement in the SAS DATA step, and it is exactly the same as the ARRAY statements in the NLIN, NLP, and MODEL procedures. The ARRAY statement is used to associate a name (of no more than eight characters) with a list of variables and constants. The array name is used with subscripts in the program to refer to the array elements. The following statements illustrate this:

```
array r[8] r1-r8;

do i = 1 to 8;
   r[i] = 0;
end;
```

The ARRAY statement does not support all the features of the ARRAY statement in the DATA step. It cannot be used to assign initial values to array elements. Implicit indexing of variables cannot be used; all array references must have explicit subscript expressions. Only exact array dimensions are allowed; lower-bound specifications are not supported. A maximum of six dimensions is allowed.

On the other hand, the ARRAY statement does allow both variables and constants to be used as array elements. (Constant array elements cannot have values assigned to them.) 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.

BOUNDS Statement

BOUNDS *b-con* < **,** *b-con . . .* > **;**

where	*b-con* :=	number *operator* parameter_list *operator* number
or	*b-con* :=	number *operator* parameter_list
or	*b-con* :=	parameter_list *operator* number
and	*operator* :=	<=, <, >=, or >

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

```
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 (upper) bound for the same parameter, the maximum (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 θ_j, the boundary constraint is replaced by $\theta_j := l_j := \min(u_j)$ defined by the minimum of all upper bounds specified for θ_j.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC NLMIXED 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 NLMIXED 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).

An optimization problem is solved for each BY group separately unless the TECH=NONE option is specified.

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*.

CMPTMODEL Statement

CMPTMODEL *required-options conditionally-required-options* < *options* > **;**

The CMPTMODEL statement computes predicted concentrations from a specified one-, two-, or three-compartment model. The CMPTMODEL statement includes three types of options:

- The following *required-options* are required: PCONC=, TIME=, NCOMPS=, ADMTYPE=, and PARMTYPE=.
- The following *conditionally-required-options* are conditionally required depending on the specification of the *required-options*: CL*n*=, VOL*n*=, K12=, K21=, K13=, K31=, KA=, K*n*0=, RATE=, and DURN=.
- The following *options* are optional: DOSE*n*=, SCALE*n*=, PCONC0=, PCONC2=, and PCONC3=.

You must specify the following *required-options*:

ADMTYPE=IVB | INF | ORAL
specifies the administration type. You can specify the following values:

IVB specifies the bolus type of administration.

INF specifies the infusion type of administration.

ORAL specifies the oral type of administration.

NCOMPS=1 | 2 | 3
specifies the number of compartments in a model.

PARMTYPE=1 | 2
PTYPE=1 | 2
specifies the parameterization type. You can specify the following values:

1 parameterizes the compartment model in terms of elimination and transfer rate constants.

2 parameterizes the compartment model in terms of clearance and volume constants.

PCONC=*variable*
PCONC1=*variable*
specifies the outcome variable that is the predicted concentration in the first or central compartment at each time point.

TIME=*variable*
specifies the time value at which the predicted concentrations are computed, where *variable* can be a data set variable or a defined variable.

You can combine the NCOMPS=, ADMTYPE=, and PARMTYPE= options to fit a compartment model (one-, two-, or three- compartments) with different administration methods (bolus, infusion, or oral) using different parameterizations. The PCONC= variable is the outcome that is the evaluated predicted concentration at the TIME= value. You can use this variable in model specification in your program.

You might need to specify one or more of the following *conditionally-required-options* based on what you specify in the *required-options*:

CL*n***=***variable*
specifies the clearance value for the *n*th compartment, where *n* takes the values of 1, 2, or 3. (CL is an alias for CL1.) This option is valid only in models that require clearance (PARMTYPE=2). The *variable* can be a data set variable or a defined variable.

- You must specify a CL1= option when you specify a one-compartment model (NCOMPS=1)
- You must specify CL1= and CL2= options when you specify a two-compartment model (NCOMPS=2)
- You must specify CL1=, CL2=, and CL3= options when you specify a three-compartment model (NCOMPS=3)

DURN=*variable*
specifies the duration of the infusion. This option is valid only in infusion models (ADMTYPE=INF). The *variable* can be a data set variable or a defined variable. In an infusion model, you must specify either this option or the RATE= option.

KA=*variable*
specifies the absorption rate constant. This option is valid only for the oral type of compartment models (ADMTYPE=ORAL).

K12=*variable*
specifies the transfer rate constant from the first (central) compartment to the second compartment. This option is valid only in models that require transfer rate (PARMTYPE=1). The *variable* can be a data set variable or a defined variable. You must specify this option in a two- or three-compartment model.

K21=*variable*
specifies the transfer rate constant from the second compartment to the first (central) compartment. This option is valid only in models that require transfer rate (PARMTYPE=1). The *variable* can be a data set variable or a defined variable. You must specify this option in a two- or three- compartment model.

K13=*variable*
specifies the transfer rate constant from the first (central) compartment to the third compartment. This option is valid only in models that require transfer rate (PARMTYPE=1). The *variable* can be a data set variable or a defined variable. You must specify this option in a three-compartment model.

K31=*variable*
specifies the transfer rate constant from the third compartment to the first (central) compartment. This option is valid only in models that require transfer rate (PARMTYPE=1). The *variable* can be a data set variable or a defined variable. You must specify this option in a three-compartment model.

K*n***0=***variable*
specifies the elimination rate constant for the *n*th compartment, where *n* takes the values of 1, 2, or 3. (Ke is an alias for K10.) This option is valid only in models that require elimination rate (PARMTYPE=1). The *variable* can be a data set variable or a defined variable.

- You must specify a K10= option when you specify a one-compartment model (NCOMPS=1)
- You must specify a K10= option and you can optionally specify a K20= option when you specify a two-compartment model (NCOMPS=2)
- You must specify a K10= option and you can optionally specify K20= and K30= options when you specify a three-compartment model (NCOMPS=3)

RATE=*variable*
specifies the rate of the infusion. This option is valid only in infusion models (ADMTYPE=INF). The *variable* can be a data set variable or a defined variable. In an infusion model, you must specify either this option or the DURN= option.

VOL*n***=***variable*
specifies the apparent volume of distribution of a drug in the *n*th compartment, where *n* takes the values of 1, 2, or 3. (VOL is an alias for VOL1.) This option is valid only in models that require volume (PARMTYPE=2). The *variable* can be a data set variable or a defined variable.

- You must specify a VOL1= option when you specify a one-compartment model (NCOMPS=1)
- You must specify VOL1= and VOL2= options when you specify a two-compartment model (NCOMPS=2)
- You must specify VOL1=, VOL2=, and VOL3= options when you specify a three-compartment model (NCOMPS=3)

You can also specify the following optional *options*:

DOSE*n*=*variable*
specifies the amount of the dose for the *n*th compartment, where *n* takes the values of 0, 1, 2, or 3. (DOSE is an alias for DOSE1.) The DOSE0= option specifies the amount of dose in the zero (depot) compartment, and is valid only in oral models (ADMTYPE=ORAL). The *variable* can be a data set variable or a defined variable.

- DOSE1= is optional and valid for one-, two-, and three-compartment models
- DOSE2= is optional and valid only for two-, and three-compartment models
- DOSE3= is optional and valid only for a three-compartment model

When ADMTYPE=ORAL is specified, DOSE0=1, DOSE1=0, DOSE2=0, and DOSE3=0 by default. Otherwise, DOSE1=1, DOSE2=0, and DOSE3=0 by default.

PCONC0=*variable*
specifies the outcome *variable* that is the predicted concentration of the zero (depot) compartment. This option is valid only when the ADMTYPE= ORAL option is specified.

PCONC2=*variable*
specifies the outcome *variable* that is the predicted concentration of the second compartment. This option is valid only in two- and three-compartment models.

PCONC3=*variable*
specifies the outcome *variable* that is the predicted concentration of the second compartment. This option is valid only in three-compartment models.

SCALE*n*=*variable*
specifies the scale value for the *n*th compartment. The *variable* can be a data set variable or a defined variable. SCALE0= option is valid only in oral models (ADMTYPE=ORAL).

- SCALE1= is optional and valid for one-, two-, and three-compartment models
- SCALE2= is optional and valid only for two-, and three-compartment models
- SCALE3= is optional and valid only for a three-compartment model

By default, all optional SCALE*n* values are set to 1.

For more information, see the section "Compartment Models" on page 7112.

CONTRAST Statement

CONTRAST '*label*' *expression* < , *expression* > < *option* > ;

The CONTRAST statement enables you to conduct a statistical test that several expressions simultaneously equal zero. The expressions are typically contrasts—that is, differences whose expected values equal zero under the hypothesis of interest.

In the CONTRAST statement, you must provide a quoted string to identify the contrast and then a list of valid SAS expressions separated by commas. Multiple CONTRAST statements are permitted, and results from all statements are listed in a common table. PROC NLMIXED constructs approximate F tests for each statement by using the delta method (Cox 1998) to approximate the variance-covariance matrix of the constituent expressions.

You can specify the following *option*:

DF=*d*
specifies the denominator degrees of freedom to be used in computing p values for the F statistics. By default, the value of *d* is the value of the DF= option in the PROC NLMIXED statement.

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 a valid SAS expression. Multiple ESTIMATE statements are permitted, and results from all statements are listed in a common table. PROC NLMIXED 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.

If you specify the ECOV option in the PROC NLMIXED statement, the procedure also produces a table that contains the approximate covariance matrix of all the additional estimates you specify. Similarly, the ECORR option produces a table of the corresponding correlation matrix, and the EDER option produces a table of the derivatives of the additional estimates with respect to each of the model parameters.

You can specify the following *options*:

ALPHA=α
specifies the alpha level to be used in computing confidence limits. By default, the value of α is the value of the ALPHA= option in the PROC NLMIXED statement.

DF=*d*
specifies the degrees of freedom to be used in computing p-values and confidence limits. By default, the value of the *d* is the value of the DF= option in the PROC NLMIXED statement.

ID Statement

ID *names* **;**

The ID statement identifies additional quantities to be included in the OUT= data set of the PREDICT statement. These can be any symbols you have defined with SAS programming statements.

MODEL Statement

MODEL *dependent-variable* ∼ *distribution* **;**

The MODEL statement is the mechanism for specifying the conditional distribution of the data given the random effects. You must specify a single dependent variable from the input data set, a tilde ∼, and then a distribution with its parameters. Valid distributions are as follows.

- **normal**(m,v) specifies a normal (Gaussian) distribution with mean *m* and variance *v*.
- **binary**(p) specifies a binary (Bernoulli) distribution with probability *p*.
- **binomial**(n,p) specifies a binomial distribution with count *n* and probability *p*.
- **gamma**(a,b) specifies a gamma distribution with shape *a* and scale *b*.
- **negbin**(n,p) specifies a negative binomial distribution with count *n* and probability *p*.
- **poisson**(m) specifies a Poisson distribution with mean *m*.
- **general**(ll) specifies a general log likelihood function that you construct using SAS programming statements.

The MODEL statement must follow any SAS programming statements you specify for computing parameters of the preceding distributions. See the section "Built-In Log-Likelihood Functions" on page 7109 for expressions of the built-in conditional log-likelihood functions.

PARMS Statement

PARMS < *name-list* < **=***numbers* > > < **,** *name-list* < **=***numbers* > . . . >
< / *options* > **;**

The PARMS statement lists names of parameters and specifies initial values, possibly over a grid. You can specify the parameters and values directly in a list, or you can provide the name of a SAS data set that contains them by using the DATA= option.

While the PARMS statement is not required, you are encouraged to use it to provide PROC NLMIXED with accurate starting values. Parameters not listed in the PARMS statement are assigned an initial value of 1. PROC NLMIXED considers all symbols not assigned values to be parameters, so you should specify your modeling statements carefully and check the output from the "Parameters" table to make sure the proper parameters are identified.

A list of parameter names in the PARMS statement is not separated by commas and is followed by an equal sign and a list of numbers. If the number list consists of only one number, this number defines the initial value for all the parameters listed to the left of the equal sign.

If the number list consists of more than one number, these numbers specify the grid locations for each of the parameters listed to the left of the equal sign. You can use the TO and BY keywords to specify a number list for a grid search. If you specify a grid of points in a PARMS statement, PROC NLMIXED computes the objective function value at each grid point and chooses the best (feasible) grid point as an initial point for the optimization process. You can use the BEST= option to save memory for the storing and sorting of all grid point information.

The following options are available in the PARMS statement after a slash (/):

BEST=*i* > 0
specifies the maximum number of points displayed in the "Parameters" table, selected as the points with the maximum likelihood values. By default, all grid values are displayed.

BYDATA
enables you to assign different starting values for each BY group by using the DATA=*SAS-data-set* option during BY processing. By default, BY groups are ignored in the PARMS data set. For the BYDATA option to be effective, the DATA= data set must contain the BY variables and the same BY groups as the primary input data set. When you supply a grid of starting values with the DATA= data set and the BYDATA option is in effect, the size of the grid is determined by the first BY group.

DATA=*SAS-data-set*
specifies a SAS data set containing parameter names and starting values. The data set should be in one of two forms: narrow or wide. The narrow-form data set contains the variables Parameter and Estimate, with parameters and values listed as distinct observations. The wide-form data set has the parameters themselves as variables, and each observation provides a different set of starting values. By default, BY groups are ignored in this data set, so the same starting grid is evaluated for each BY group. You can vary the starting values for BY groups by using the BYDATA option.

PREDICT Statement

PREDICT *expression* **OUT=***SAS-data-set* < *options* > **;**

The PREDICT statement enables you to construct predictions of an expression across all of the observations in the input data set. Any valid SAS programming expression involving the input data set variables, parameters, and random effects is valid. Predicted values are computed using the parameter estimates and empirical Bayes estimates of the random effects. Standard errors of prediction are computed using the delta method (Billingsley 1986; Cox 1998). Results are placed in an output data set that you specify with the OUT= option. Besides all variables from the input data set, the OUT= data set contains the following variables: Pred, StdErrPred, DF, tValue, Probt, Alpha, Lower, Upper. You can also add other computed quantities to this data set with the ID statement.

The following options are available in the PREDICT statement:

ALPHA=*α*
specifies the alpha level to be used in computing *t* statistics and intervals. The default value corresponds to the ALPHA= option in the PROC NLMIXED statement.

DER
requests that derivatives of the predicted expression with respect to all parameters be included in the OUT= data set. The variable names for the derivatives are the same as the parameter names with the prefix "Der_" appended. All of the derivatives are evaluated at the final estimates of the parameters and the empirical Bayes estimates of the random effects.

DF=*d*
specifies the degrees of freedom to be used in computing *t* statistics and intervals in the OUT= data set. The default value corresponds to the DF= option in the PROC NLMIXED statement.

RANDOM Statement

RANDOM *random-effects* ∼ *distribution* **SUBJECT=***variable* < *options* > **;**

The RANDOM statement defines the random effects and their distribution. The random effects must be represented by symbols that appear in your SAS programming statements. The random effects usually influence the mean value of the distribution that is specified in the MODEL statement. The RANDOM statement consists of a list of the random effects (usually just one or two symbols), a tilde (∼), the distribution of the random effects, and then a SUBJECT= variable.

The only distribution available for the random effects is normal(*m*,*v*), with mean *m* and variance *v*.

This syntax is illustrated as follows for one effect:

```
random u ~ normal(0,s2u) subject=clinic;
```

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

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

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

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

The SUBJECT= variable determines the unique realizations of the random effects.

PROC NLMIXED constructs the subject clusters based on unique values in the SUBJECT= variable. This construction process of clusters, starting in SAS/STAT 13.1, is different from that in earlier releases of PROC NLMIXED, where a change in the SUBJECT= variable value indicates a new cluster. Because of this change, starting in SAS/STAT 13.1, the input data set does not need to be sorted by SUBJECT= variable to use the unique SUBJECT= variable values.

On the other hand, in earlier releases of SAS/STAT, PROC NLMIXED does not sort the input data set for you; rather, it processes the data sequentially and considers an observation to be from a new subject whenever the value of its SUBJECT= variable changes from the previous observation. To revert to this sequential

process behavior of SAS/STAT 12.3 and previous releases, specify the NOSORTSUB option in the PROC NLMIXED statement.

You can specify multiple RANDOM statements; this is supported starting in SAS/STAT 13.2. When you specify more than one RANDOM statement, PROC NLMIXED assumes that the SUBJECT= variable from each RANDOM statement forms a containment hierarchy. For more information, see the section "Hierarchical Model Specification" on page 7122. The syntax for two RANDOM statements is illustrated as follows:

```
random r11 ~ normal(0,sd1) subject = school;
random r21 ~ normal(0,sd2) subject = class(school);
```

You can specify the following options in the RANDOM statement:

ALPHA=*α*
: specifies the alpha level to be used in computing *t* statistics and intervals. The default value corresponds to the value of the ALPHA= option in the PROC NLMIXED statement.

DF=*d*
: specifies the degrees of freedom to be used in computing *t* statistics and intervals in the OUT= data set. PROC NLMIXED calculates the default degrees of freedom as follows:

- When only one or no RANDOM statement is specified, the default value corresponds to the value of the DF= option in the PROC NLMIXED statement.
- When multiple RANDOM statements are specified, the default value is the number of subjects minus the number of random-effects variables in the corresponding RANDOM statement.

OUT=*SAS-data-set*
: requests an output data set that contains empirical Bayes estimates of the random effects and their approximate standard errors of prediction.

REPLICATE Statement

REPLICATE *variable* **;**

The REPLICATE statement provides a way to accommodate models in which different subjects have identical data. This occurs most commonly when the dependent variable is binary. When you specify a REPLICATE variable, PROC NLMIXED treats its value as the number of subjects that have data identical to the data for the current value of the SUBJECT= variable (specified in the RANDOM statement). Only the last observation of the REPLICATE variable for each subject is used, and the replicate variable must have only positive values.

The function of a REPLICATE statement is related to but not quite the same as the function of either a FREQ or a WEIGHT statement in other statistical modeling procedures, such as the GLM, GENMOD, GLIMMIX, and LOGISTIC procedures. A FREQ or a WEIGHT value essentially multiplies the log likelihood or sum of squares contribution for each observation. On the other hand, a REPLICATE value multiplies the log likelihood contribution of each subject, which consists of one or more observations. When no SUBJECT= variable is specified, the REPLICATE value behaves like a weight, multiplying each observation's log likelihood contribution.

NOTE: The REPLICATE statement is not allowed when there is more than one RANDOM statement.

Programming Statements

This section lists the programming statements that are used to code the log-likelihood function in PROC NLMIXED. It also documents the differences between programming statements in PROC NLMIXED and programming statements in the SAS DATA step. The syntax of programming statements used in PROC NLMIXED is identical to that used in the CALIS and GENMOD procedures (see Chapter 32 and Chapter 50, respectively), and the MODEL procedure (see the *SAS/ETS User's Guide*). Most of the programming statements that can be used in the SAS DATA step can also be used in the NLMIXED procedure. See *SAS DATA Step Statements: Reference* for a description of SAS programming statements. The following are valid statements:

ABORT;
ARRAY *arrayname* < **[** *dimensions* **]** > < **$** > < *variables-and-constants* >**;**
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 Programmers Guide: Essentials*; however, there are the following differences:

- The ABORT statement does not allow any arguments.
- The DO statement does not allow a character index variable. Thus

```
do i = 1,2,3;
```

  is supported, but the following statement is not supported:

```
do i = 'A','B','C';
```

- The LAG function does work appropriately with PROC NLMIXED, but you can use the ZLAG function instead.
- The PUT statement, used mostly for program debugging in PROC NLMIXED, 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 PROC NLMIXED PUT statement does not support line pointers, factored lists, iteration factors, overprinting, _INFILE_, the colon (:) format modifier, or "$".
  - The PROC NLMIXED PUT statement does support 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 NLMIXED PUT statement supports the item _PDV_ to display 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 target statement. That is, DO/END groups are not necessary for multiple statement WHENs. For example, the following syntax is valid:

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

When coding your programming statements, you should avoid defining variables that begin with an underscore (_), because they might conflict with internal variables created by PROC NLMIXED. The MODEL statement must come after any SAS programming statements that define or modify terms used in the construction of the log-likelihood.

Details: NLMIXED Procedure

This section contains details about the underlying theory and computations of PROC NLMIXED.

Modeling Assumptions and Notation

PROC NLMIXED operates under the following general framework for nonlinear mixed models. Assume that you have an observed data vector $\mathbf{y}_i$ for each of i subjects, $i = 1, \ldots, s$. The $\mathbf{y}_i$ are assumed to be independent across i, but within-subject covariance is likely to exist because each of the elements of $\mathbf{y}_i$ is measured on the same subject. As a statistical mechanism for modeling this within-subject covariance, assume that there exist latent random-effect vectors $\mathbf{u}_i$ of small dimension (typically one or two) that are also independent across i. Assume also that an appropriate model linking $\mathbf{y}_i$ and $\mathbf{u}_i$ exists, leading to the joint probability density function

$$p(\mathbf{y}_i|\mathbf{X}_i, \boldsymbol{\phi}, \mathbf{u}_i)q(\mathbf{u}_i|\boldsymbol{\xi})$$

where $\mathbf{X}_i$ is a matrix of observed explanatory variables and $\boldsymbol{\phi}$ and $\boldsymbol{\xi}$ are vectors of unknown parameters.

Let $\boldsymbol{\theta} = [\boldsymbol{\phi}, \boldsymbol{\xi}]$ and assume that it is of dimension n. Then inferences about $\boldsymbol{\theta}$ are based on the marginal likelihood function

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

In particular, the function

$$f(\boldsymbol{\theta}) = -\log m(\boldsymbol{\theta})$$

is minimized over $\boldsymbol{\theta}$ numerically in order to estimate $\boldsymbol{\theta}$, and the inverse Hessian (second derivative) matrix at the estimates provides an approximate variance-covariance matrix for the estimate of $\boldsymbol{\theta}$. The function $f(\boldsymbol{\theta})$ is referred to both as the negative log likelihood function and as the objective function for optimization.

As an example of the preceding general framework, consider the nonlinear growth curve example in the section "Getting Started: NLMIXED Procedure" on page 7068. Here, the conditional distribution $p(\mathbf{y}_i|\mathbf{X}_i, \boldsymbol{\phi}, u_i)$ is normal with mean

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

and variance σ_e^2; thus $\boldsymbol{\phi} = [b_1, b_2, b_3, \sigma_e^2]$. Also, u_i is a scalar and $q(u_i|\xi)$ is normal with mean 0 and variance σ_u^2; thus $\xi = \sigma_u^2$.

The following additional notation is also found in this chapter. The quantity $\boldsymbol{\theta}^{(k)}$ refers to the parameter vector at the kth iteration, the vector $\mathbf{g}(\boldsymbol{\theta})$ refers to the gradient vector $\nabla f(\boldsymbol{\theta})$, and the matrix $\mathbf{H}(\boldsymbol{\theta})$ refers to the Hessian $\nabla^2 f(\boldsymbol{\theta})$. Other symbols are used to denote various constants or option values.

Nested Multilevel Nonlinear Mixed Models

The general framework for nested multilevel nonlinear mixed models in cases of two levels can be explained as follows. Let $\mathbf{y}_{j(i)}$ be the response vector observed on subject j that is nested within subject i, where j is commonly referred as the second-level subject and i is the first-level subject. There are s first-level subjects, and each has s_i second-level subjects that are nested within. An example is $\mathbf{y}_{j(i)}$, which are the heights of students in class j of school i, where $j = 1, \ldots, s_i$ for each i and $i = 1, \ldots, s$. Suppose there exist latent random-effect vectors $\mathbf{v}_{j(i)}$ and $\mathbf{v}_i$ of small dimensions for modeling within subject covariance. Assume also that an appropriate model that links $\mathbf{y}_{j(i)}$ and $(\mathbf{v}_{j(i)}, \mathbf{v}_i)$ exists, and if you use the notation $\mathbf{y}_i = (\mathbf{y}_{1(i)}, \ldots, \mathbf{y}_{s_i(i)})$, $\mathbf{u}_i = (\mathbf{v}_i, \mathbf{v}_{1(i)}, \ldots, \mathbf{v}_{s_i(i)})$, and $\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \boldsymbol{\xi}_2)$, the joint density function in terms of the first-level subject can be expressed as

$$p(\mathbf{y}_i|\mathbf{X}_i, \boldsymbol{\phi}, \mathbf{u}_i)q(\mathbf{u}_i|\boldsymbol{\xi}) = \left(\prod_{j=1}^{s_i} p(\mathbf{y}_{j(i)}|\mathbf{X}_i, \boldsymbol{\phi}, \mathbf{v}_i, \mathbf{v}_{j(i)})q_2(\mathbf{v}_{j(i)}|\boldsymbol{\xi}_2) \right) q_1(\mathbf{v}_i|\boldsymbol{\xi}_1)$$

As defined in the previous section, the marginal likelihood function where $\boldsymbol{\theta} = [\boldsymbol{\phi}, \boldsymbol{\xi}]$ is

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

Again, the function

$$f(\boldsymbol{\theta}) = -\log m(\boldsymbol{\theta})$$

is minimized over $\boldsymbol{\theta}$ numerically in order to estimate $\boldsymbol{\theta}$. Models that have more than two levels follow similar notation.

Integral Approximations

An important part of the marginal maximum likelihood method described previously is the computation of the integral over the random effects. The default method in PROC NLMIXED for computing this integral is adaptive Gaussian quadrature as described in Pinheiro and Bates (1995). Another approximation method is the first-order method of Beal and Sheiner (1982, 1988). A description of these two methods follows.

Adaptive Gaussian Quadrature

A quadrature method approximates a given integral by a weighted sum over predefined abscissas for the random effects. A good approximation can usually be obtained with an adequate number of quadrature points as well as appropriate centering and scaling of the abscissas. Adaptive Gaussian quadrature for the integral over $\mathbf{u}_i$ centers the integral at the empirical Bayes estimate of $\mathbf{u}_i$, defined as the vector $\widehat{\mathbf{u}}_i$ that minimizes

$$-\log\left[p(\mathbf{y}_i|\mathbf{X}_i, \boldsymbol{\phi}, \mathbf{u}_i)q(\mathbf{u}_i|\boldsymbol{\xi})\right]$$

with $\boldsymbol{\phi}$ and $\boldsymbol{\xi}$ set equal to their current estimates. The final Hessian matrix from this optimization can be used to scale the quadrature abscissas.

Suppose $(z_j, w_j; j = 1, \ldots, p)$ denote the standard Gauss-Hermite abscissas and weights (Golub and Welsch 1969, or Table 25.10 of Abramowitz and Stegun 1972). The adaptive Gaussian quadrature integral

approximation is as follows:

$$\int p(\mathbf{y}_i|\mathbf{X}_i,\boldsymbol{\phi},\mathbf{u}_i)q(\mathbf{u}_i|\boldsymbol{\xi})d\mathbf{u}_i \approx$$
$$2^{r/2}\,|\boldsymbol{\Gamma}(\mathbf{X}_i,\boldsymbol{\theta})|^{-1/2}\sum_{j_1=1}^{p}\cdots\sum_{j_r=1}^{p}\left[p(\mathbf{y}_i|\mathbf{X}_i,\boldsymbol{\phi},\mathbf{a}_{j_1,\ldots,j_r})q(\mathbf{a}_{j_1,\ldots,j_r}|\boldsymbol{\xi})\prod_{k=1}^{r}w_{j_k}\exp z_{j_k}^2\right]$$

where r is the dimension of u_i, $\boldsymbol{\Gamma}(\mathbf{X}_i,\boldsymbol{\theta})$ is the Hessian matrix from the empirical Bayes minimization, $\mathbf{z}_{j_1,\ldots,j_r}$ is a vector with elements $(z_{j_1},\ldots,z_{j_r})$, and

$$\mathbf{a}_{j_1,\ldots,j_r} = \widehat{\mathbf{u}}_i + 2^{1/2}\boldsymbol{\Gamma}(\mathbf{X}_i,\boldsymbol{\theta})^{-1/2}\mathbf{z}_{j_1,\ldots,j_r}$$

PROC NLMIXED selects the number of quadrature points adaptively by evaluating the log-likelihood function at the starting values of the parameters until two successive evaluations have a relative difference less than the value of the QTOL= option. The specific search sequence is described under the QFAC= option. Using the QPOINTS= option, you can adjust the number of quadrature points p to obtain different levels of accuracy. Setting $p = 1$ results in the Laplacian approximation as described in: Beal and Sheiner (1992); Wolfinger (1993); Vonesh (1992, 1996); Vonesh and Chinchilli (1997); Wolfinger and Lin (1997).

The NOAD option in the PROC NLMIXED statement requests nonadaptive Gaussian quadrature. Here all $\widehat{\mathbf{u}}_i$ are set equal to zero, and the Cholesky root of the estimated variance matrix of the random effects is substituted for $\boldsymbol{\Gamma}(\mathbf{X}_i,\boldsymbol{\theta})^{-1/2}$ in the preceding expression for $\mathbf{a}_{j_1,\ldots,j_r}$. In this case derivatives are computed using the algorithm of Smith (1995). The NOADSCALE option requests the same scaling substitution but with the empirical Bayes $\widehat{\mathbf{u}}_i$.

When there is one RANDOM statement, the dimension of $\mathbf{u}_i$ is the number of random effects that are specified in that RANDOM statement. For nested multilevel nonlinear mixed models, the dimension of $\mathbf{u}_i$ could increase significantly. In this case, the dimension of $\mathbf{u}_i$ is equal to the sum of all nested subjects times the corresponding random-effects dimension. For example, consider a three-level nested nonlinear mixed model that contains s first-level subjects, where each first-level subject has s_i second-level subjects that are nested within and s_{ij} third-level subjects that are nested within each combination of first-level subject i and second-level subject j. Then, based on notation in the previous section, $\mathbf{u}_i = (\mathbf{v}_i, \mathbf{v}_{1(i1)}, \ldots, \mathbf{v}_{s_{i1}(i1)}, \ldots, \mathbf{v}_{s_i(i)}, \mathbf{v}_{1(is_i)}, \ldots, \mathbf{v}_{s_{is_i}(is_i)})$. Suppose r_i is the random-effects dimension at level i. Then the dimension of $\mathbf{u}_i$, denoted as r, can be computed as $r = r_1 + (r_2 \times s_i) + \left(r_3 \times \sum_{j=1}^{s_i} s_{ij}\right)$. Hence the joint estimation of $\mathbf{u}_i$ is more costly.

When only one RANDOM statement is specified for the model, by default PROC NLMIXED uses Newton-Raphson optimization for empirical Bayes minimization of random effects for each subject. If you specify the EBOPT option in the PROC NLMIXED statement, then the procedure uses Newton-Raphson ridge optimization. But when multiple RANDOM statements are specified, quasi-Newton optimization is used for the empirical Bayes minimization of random effects for each subject. Therefore, all the options that are related to the Newton-Raphson method (such as the EBOPT, EBSSFRAC=, EBSSTOL=, EBSTEPS=, EBSUBSTEPS=, and EBTOL= options) are ignored.

PROC NLMIXED computes the derivatives of the adaptive Gaussian quadrature approximation when carrying out the default dual quasi-Newton optimization. For nested multilevel nonlinear mixed models, PROC NLMIXED does not explicitly compute these derivatives. In this case, it ignores the EMPIRICAL

and SUBGRADIENT= options in the PROC NLMIXED statement. Also, nested multilevel nonlinear mixed models use METHOD=GAUSS only. In addition to these changes, for nested multilevel nonlinear mixed models, the default values of GTOL and ABSGTOL are changed to 1E–6 and 1E–3, respectively.

First-Order Method

Another integral approximation available in PROC NLMIXED is the first-order method of Beal and Sheiner (1982, 1988) and Sheiner and Beal (1985). This approximation is used only in the case where $p(\mathbf{y}_i|\mathbf{X}_i,\boldsymbol{\phi},\mathbf{u}_i)$ is normal—that is,

$$p(\mathbf{y}_i|\mathbf{X}_i,\boldsymbol{\phi},\mathbf{u}_i) = (2\pi)^{-n_i/2}\,|\mathbf{R}_i(\mathbf{X}_i,\boldsymbol{\phi})|^{-1/2} \\ \exp\left\{-(1/2)\,[\mathbf{y}_i-\mathbf{m}_i(\mathbf{X}_i,\boldsymbol{\phi},\mathbf{u}_i)]'\,\mathbf{R}_i(\mathbf{X}_i,\boldsymbol{\phi})^{-1}\,[\mathbf{y}_i-\mathbf{m}_i(\mathbf{X}_i,\boldsymbol{\phi},\mathbf{u}_i)]\right\}$$

where n_i is the dimension of $\mathbf{y}_i$, $\mathbf{R}_i$ is a diagonal variance matrix, and $\mathbf{m}_i$ is the conditional mean vector of $\mathbf{y}_i$.

The first-order approximation is obtained by expanding $\mathbf{m}_i(\mathbf{X}_i,\boldsymbol{\phi},\mathbf{u}_i)$ with a one-term Taylor series expansion about $\mathbf{u}_i=\mathbf{0}$, resulting in the approximation

$$p(\mathbf{y}_i|\mathbf{X}_i,\boldsymbol{\phi},\mathbf{u}_i) \approx (2\pi)^{-n_i/2}\,|\mathbf{R}_i(\mathbf{X}_i,\boldsymbol{\phi})|^{-1/2} \\ \exp\Big(-(1/2)\,[\mathbf{y}_i-\mathbf{m}_i(\mathbf{X}_i,\boldsymbol{\phi},\mathbf{0})-\mathbf{Z}_i(\mathbf{X}_i,\boldsymbol{\phi})\mathbf{u}_i]' \\ \mathbf{R}_i(\mathbf{X}_i,\boldsymbol{\phi})^{-1}\,[\mathbf{y}_i-\mathbf{m}_i(\mathbf{X}_i,\boldsymbol{\phi},\mathbf{0})-\mathbf{Z}_i(\mathbf{X}_i,\boldsymbol{\phi})\mathbf{u}_i]\Big)$$

where $\mathbf{Z}_i(\mathbf{X}_i,\boldsymbol{\phi})$ is the Jacobian matrix $\partial\mathbf{m}_i(\mathbf{X}_i,\boldsymbol{\phi},\mathbf{u}_i)/\partial\mathbf{u}_i$ evaluated at $\mathbf{u}_i=\mathbf{0}$.

Assuming that $q(\mathbf{u}_i|\boldsymbol{\xi})$ is normal with mean $\mathbf{0}$ and variance matrix $\mathbf{G}(\boldsymbol{\xi})$, the first-order integral approximation is computable in closed form after completing the square:

$$\int p(\mathbf{y}_i|\mathbf{X}_i,\boldsymbol{\phi},\mathbf{u}_i)q(\mathbf{u}_i|\boldsymbol{\xi})\,d\mathbf{u}_i \approx (2\pi)^{-n_i/2}\,|\mathbf{V}_i(\mathbf{X}_i,\boldsymbol{\theta})|^{-1/2} \\ \exp\left(-(1/2)\,[\mathbf{y}_i-\mathbf{m}_i(\mathbf{X}_i,\boldsymbol{\phi},\mathbf{0})]'\,\mathbf{V}_i(\mathbf{X}_i,\boldsymbol{\theta})^{-1}\,[\mathbf{y}_i-\mathbf{m}_i(\mathbf{X}_i,\boldsymbol{\phi},\mathbf{0})]\right)$$

where $\mathbf{V}_i(\mathbf{X}_i,\boldsymbol{\theta}) = \mathbf{Z}_i(\mathbf{X}_i,\boldsymbol{\phi})\mathbf{G}(\boldsymbol{\xi})\mathbf{Z}_i(\mathbf{X}_i,\boldsymbol{\phi})' + \mathbf{R}_i(\mathbf{X}_i,\boldsymbol{\phi})$. The resulting approximation for $f(\boldsymbol{\theta})$ is then minimized over $\boldsymbol{\theta}=[\boldsymbol{\phi},\boldsymbol{\xi}]$ to obtain the first-order estimates. PROC NLMIXED uses finite-difference derivatives of the first-order integral approximation when carrying out the default dual quasi-Newton optimization.

Built-In Log-Likelihood Functions

This section displays the basic formulas used by the NLMIXED procedure to compute the conditional log-likelihood functions of the data given the random effects. Note, however, that in addition to these basic equations, the NLMIXED procedure employs a number of checks for missing values and floating-point arithmetic. You can see the entire program used by the NLMIXED procedure to compute the conditional log-likelihood functions $l(\boldsymbol{\phi};y)$ by adding the LIST debugging option to the PROC NLMIXED statement.

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

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

$Y \sim \text{binary}(p)$

$$
\begin{aligned}
l_1(p; y) &= \begin{cases} y \ \log\{p\} & y > 0 \\ 0 & \text{otherwise} \end{cases} \\
l_2(p; y) &= \begin{cases} (1-y) \ \log\{1-p\} & y < 1 \\ 0 & \text{otherwise} \end{cases} \\
l(p; y) &= l_1(p; y) + l_2(p; y) \\
\text{E}[Y] &= p \\
\text{Var}[Y] &= p\,(1-p) \\
0 &< p < 1
\end{aligned}
$$

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

$$
\begin{aligned}
l_c &= \log\{\Gamma(n+1)\} - \log\{\Gamma(y+1)\} - \log\{\Gamma(n-y+1)\} \\
l_1(n, p; y) &= \begin{cases} y \ \log\{p\} & y > 0 \\ 0 & \text{otherwise} \end{cases} \\
l_2(n, p; y) &= \begin{cases} (n-y) \ \log\{1-p\} & n-y > 0 \\ 0 & \text{otherwise} \end{cases} \\
l(n, p; y) &= l_c + l_1(n, p; y) + l_2(n, p; y) \\
\text{E}[Y] &= n\ p \\
\text{Var}[Y] &= n\ p\,(1-p) \\
0 &< p < 1
\end{aligned}
$$

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

$$
\begin{aligned}
l(a, b; y) &= -a\log\{b\} - \log\{\Gamma(a)\} + (a-1)\log\{y\} - y/b \\
\text{E}[Y] &= ab \\
\text{Var}[Y] &= ab^2 \\
a &> 0 \\
b &> 0
\end{aligned}
$$

This parameterization of the gamma distribution differs from the parameterization used in the GLIMMIX and GENMOD procedures. The following statements show the equivalent reparameterization in the NLMIXED procedure that fits a generalized linear model for gamma-distributed data in the parameterization of the GLIMMIX procedure:

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


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

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

$$
\begin{aligned}
l(n, p; y) &= \log\{\Gamma(n + y)\} - \log\{\Gamma(n)\} - \log\{\Gamma(y + 1)\} \\
&\quad + n \log\{p\} + y \log\{1 - p\} \\
\mathrm{E}[Y] &= nP = n\left(\frac{1-p}{p}\right) \\
\mathrm{Var}[Y] &= nP(1 - P) = n\left(\frac{1-p}{p}\right)\frac{1}{p} \\
n &\geq 0 \\
0 &< p < 1
\end{aligned}
$$

This form of the negative binomial distribution is one of the many parameterizations in which the mass function or log-likelihood function appears. Another common parameterization uses

$$
\begin{aligned}
l(n, p; y) &= \log\{\Gamma(n + y)\} - \log\{\Gamma(n)\} - \log\{\Gamma(y + 1)\} \\
&\quad + n \log\{1 - P/(1 + P)\} + y \log\{P/(1 + P)\}
\end{aligned}
$$

with $P = (1 - p)/p$, $P > 0$.

Note that the parameter n can be real-numbered; it does not have to be integer-valued. The parameterization of the negative binomial distribution in the NLMIXED procedure differs from that in the GLIMMIX and GENMOD procedures. The following statements show the equivalent formulations for maximum likelihood estimation in the GLIMMIX and NLMIXED procedures in a negative binomial regression model:

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


proc nlmixed;
   parms b0=3, b1=1, k=0.8;
   linp = b0 + b1*x;
   mu = exp(linp);
   p  = 1/(1+mu*k);
```

```
   model y ~ negbin(1/k,p);
run;
```

$Y \sim \text{Poisson}(m)$

$$
\begin{aligned}
l(m;y) &= y\log\{m\} - m - \log\{\Gamma(y+1)\} \\
\text{E}[Y] &= m \\
\text{Var}[Y] &= m \\
m &> 0
\end{aligned}
$$

Compartment Models

Pharmacokinetics (PK) is a branch of medicine that models the movement of a drug through the body (Gabrielsson and Weiner 2006). PK is sometimes referred to as the study of what the body does to a drug. Compartment models are basic building blocks of PK models. In a study, a body is divided in to several *compartments*, groups of organs or tissues that are kinetically homogeneous. The main interest of PK is to model how a drug moves through these compartments—for example, to estimate the amount of a drug and further concentrations of the drug that are present in a compartment at any given time. The concentrations of the drug, over time, are typically modeled via a set of differential equations that depend on a variety of variables, such as the amount of drug given, elimination rates, transfer rates, and so on. The sets of differential equations can also depend on how the drug is administered to the body, distributed through the body, and eliminated from the body. These models are known as compartment models, which are divided based on the number of compartments. Not all analytical solutions to multiple-compartment differential equations models are known, but Abuhelwa, Foster, and Upton (2015) and Fisher and Shafer (2007) provide closed-form solutions for those that correspond to the one-, two-, and three- compartments. These are the models that are handled by the CMPTMODEL statement.

Routes of Drug Administration

There are three types of drug administration methods: intravenous bolus, intravenous infusion, and extravascular dose administrations. A bolus medication typically has no or a very short time lag for the drug to enter a compartment. Intravenous infusions are given over a period of time, and the drug enters into the body at a constant rate. In infusion, the two quantities, *rate* and *duration* of the infusion, are of interest and are used in calculating the amount of drug present in the body. The rate and the duration of the infusion are related, so knowing one determines the other. For compartment models that have infusion type of administration, it is sufficient to provide either the rate or the duration information.

One-, Two-, and Three-Compartment Models for Intravenous Administration

This section lists a number of assumptions that are made on the set of compartment models that the CMPTMODEL statement supports. The one-, two-, and three-compartment models all have a central compartment and can have one or more peripheral compartments that are linked only to the central compartment but not to each other. After a drug is administered to an administration site, it is distributed to the central compartment and then to other peripheral compartments. The rates at which the drug moves from the central compartment to and from the other peripheral compartments are characterized by transfer rate constants. The rates at which the drug is eliminated from the central or the peripheral compartments are characterized by elimination rate constants.

Schematic representation of the one-, two-, and three- compartment models are given in Figure 88.12, Figure 88.13, and Figure 88.14, respectively. In each figure, Dose is the dosage of the drug that is given intravenously, and k_{10} represents the rate at which the drug is eliminated from compartment 1. In Figure 88.13 and Figure 88.14, k_{12} represents the transfer rate constant from compartment 1 to compartment 2 and k_{21} represents the transfer rate constant from compartment 2 to compartment 1. Similarly, in Figure 88.14, k_{13} represents the transfer rate constant from compartment 1 to compartment 3 and k_{31} represents the transfer rate constant from compartment 3 to compartment 1. In these scenarios, compartment 1 is the central compartment and compartments 2 and 3 are the peripheral compartments. In compartment models, a drug can be administered only to and eliminated only from the central compartment.

Figure 88.12 One-Compartment Model

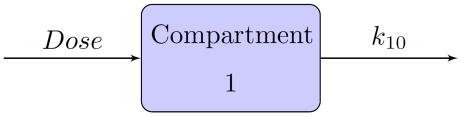


Figure 88.13 Two-Compartment Model

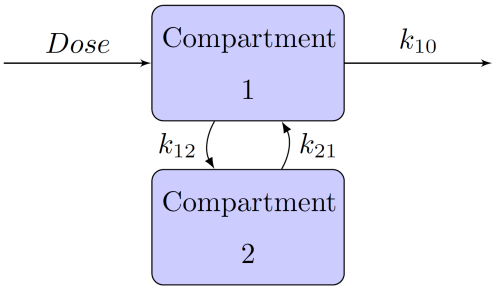
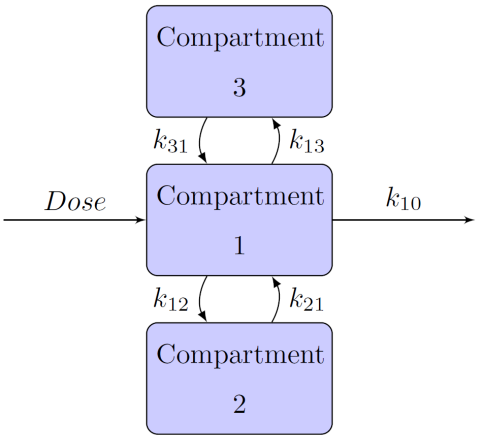


Figure 88.14 Three-Compartment Model



One-, Two-, and Three-Compartment Models for Extravascular Administration

Extravascular administrations, such as oral administration, are different from bolus and infusion in the sense that there is an absorption phase before the drug enters the central compartment. Schematic representation of the one-, two-, and three- compartment models when the drug is administered using an oral method are given in Figure 88.15, Figure 88.16, and Figure 88.17, respectively. In these models, compartment 0 is also called the *depot* compartment.

Figure 88.15 One-Compartment Model with Absorption Phase

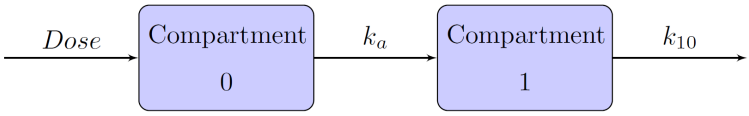


Figure 88.16 Two-Compartment Model with Absorption Phase

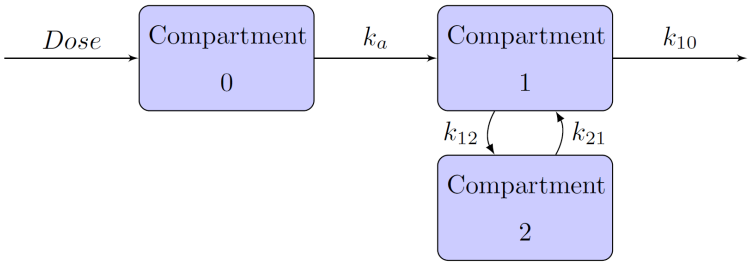
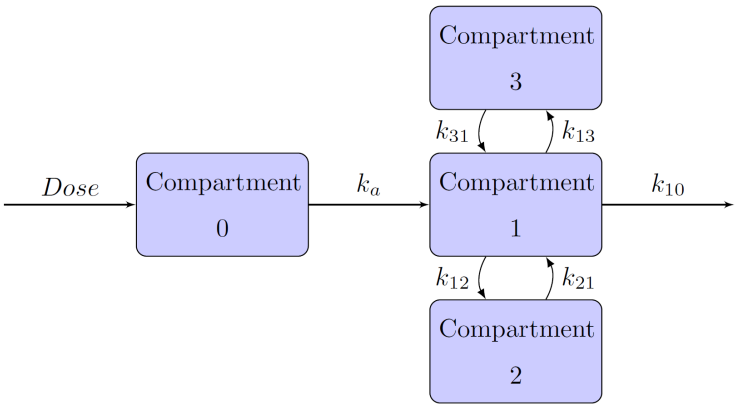


Figure 88.17 Three-Compartment Model with Absorption Phase



Compartment Models Specification

The CMPTMODEL statement in PROC NLMIXED enables you to fit one-, two-, and three-compartment models that use bolus, infusion, or oral types of drug administration. The CMPTMODEL statement computes the predicted concentrations of the drug for all compartments at each time point.

The CMPTMODEL statement supports a large number of options, which are grouped into three types: required, conditionally-required, and optional options. You use the required options to specify the type of compartment model that you want to fit (the NCOMPS=, ADMTYPE=, and PARMTYPE= options) and to

provide the input time variable (in the TIME= option) and the outcome variable (in the PCONC= option) that is the predicted concentration in the first (central) compartment at each time point.

The statement supports the following nine types of compartment models:

1. one-compartment model with bolus dose administration
2. one-compartment model with infusion type of dose administration
3. one-compartment model with oral dose administration
4. two-compartment model with bolus dose administration
5. two-compartment model with infusion type of dose administration
6. two-compartment model with oral dose administration
7. three-compartment model with bolus dose administration
8. three-compartment model with infusion type of dose administration
9. three-compartment model with oral dose administration.

Each of these nine models can be parameterized in two ways: in terms of elimination and transfer rate constants of each compartment, or in terms of clearance and volume parameters of each compartment. The relationship between rate constants and the clearance and volumes of each compartment are:

$$K_{10} = CL1/VOL1$$
$$K_{12} = CL2/VOL1$$
$$K_{13} = CL3/VOL1$$
$$K_{21} = CL2/VOL2$$
$$K_{31} = CL3/VOL3$$

where CL*n* and VOL*n* are the clearance and volume of the *n*th compartment for *n*=1,2,3. You can use the PARMTYPE= option in the CMPTMODEL statement to specify these parameterizations. In total, you can specify 18 types of compartment models by using combinations of the NCOMPS=, ADMTYPE=, and PARMTYPE= options in the CMPTMODEL statement.

The conditionally required options define the specifics needed in some of the compartment models. For example, if you specify a two-compartment model with a bolus type of drug administration in terms of rate constants by using NCOMPS=2, ADMTYPE=IVB, and PARMTYPE=1, then you must specify the K12=, K21=, and K10= options. Here is an example:

```
PROC NLMIXED data=TwoComp;

...other statements...

erate10= exp(beta1);
trate12= exp(beta2);
trate21= exp(beta3);

CMPTMODEL ncomps=2 admtype=ivb parmtype=1 pconc=pred time=time
          k10=erate10 k12=trate12 k21=trate21
          dose1=ddose scale1=vol/1000;
model conc ~ normal(pred,s2);

...other statements...

run;
```

The K12= and K21= options specify the transfer rate constants, and the K10= option specifies the elimination rate constant from the central compartment. In addition, you can specify dosage information (DOSE*n*=) or scaling information (SCALE*n*=) in the syntax. For more information, see the section "CMPTMODEL Statement" on page 7095.

Table 88.2, Table 88.3, and Table 88.4 list all conditionally required options and valid optional options for each of the 18 compartment models.

Table 88.2 One-Compartment Models

Model	Required	Conditionally Required	Optional
One compartment with bolus dose	NCOMPS=1 ADMTYPE=IVB PARMTYPE=1	K10=	DOSE1= SCALE1=
One compartment with bolus dose	NCOMPS=1 ADMTYPE=IVB PARMTYPE=2	CL1= VOL1=	DOSE1= SCALE1=
One compartment with infusion dose	NCOMPS=1 ADMTYPE=INF PARMTYPE=1	K10= RATE= (or) DURN=	DOSE1= SCALE1=
One compartment with infusion dose	NCOMPS=1 ADMTYPE=INF PARMTYPE=2	CL1= VOL1= RATE= (or) DURN=	DOSE1= SCALE1=
One compartment with oral dose	NCOMPS=1 ADMTYPE=ORAL PARMTYPE=1	K10= Ka=	DOSE1= SCALE1=
One compartment with oral dose	NCOMPS=1 ADMTYPE=ORAL PARMTYPE=2	CL1= VOL1= Ka=	DOSE1= SCALE1=

Table 88.3 Two-Compartment Models

Model	Required	Conditionally Required	Optional
Two compartments with bolus dose	NCOMPS=2 ADMTYPE=IVB PARMTYPE=1	K10= K12= K21=	DOSE1= DOSE2= SCALE1= SCALE2= K20= PCONC2=
Two compartments with bolus dose	NCOMPS=2 ADMTYPE=IVB PARMTYPE=2	CL1= VOL1= CL2= VOL2=	DOSE1= DOSE2= SCALE1= SCALE2= K20= PCONC2=
Two compartments with infusion dose	NCOMPS=2 ADMTYPE=INF PARMTYPE=1	K10= K12= K21= RATE= (or) DURN=	DOSE1= SCALE1=
Two compartments with infusion dose	NCOMPS=2 ADMTYPE=INF PARMTYPE=2	CL1= VOL1= CL2= VOL2= RATE= (or) DURN=	DOSE1= SCALE1=
Two compartments with oral dose	NCOMPS=2 ADMTYPE=ORAL PARMTYPE=1	K10= K12= K21= Ka=	DOSE1= DOSE2= SCALE1= SCALE2= K20= PCONC2=
Two compartments with oral dose	NCOMPS=2 ADMTYPE=ORAL PARMTYPE=2	CL1= VOL1= CL2= VOL2= Ka=	DOSE1= DOSE2= SCALE1= SCALE2= K20= PCONC2=

Table 88.4 Three-Compartment Models

Model	Required	Conditionally Required	Optional
Three compartments with bolus dose	NCOMPS=3 ADMTYPE=IVB PARMTYPE=1	K10= K12= K21= K13= K31=	DOSE1= DOSE2= DOSE3= SCALE1= SCALE2= SCALE3= K20= K30= PCONC2= PCONC3=
Three compartments with bolus dose	NCOMPS=3 ADMTYPE=IVB PARMTYPE=2	CL1= VOL1= CL2= VOL2= CL3= VOL3=	DOSE1= DOSE2= DOSE3= SCALE1= SCALE2= SCALE3= K20= K30= PCONC2= PCONC3=
Three compartments with infusion dose	NCOMPS=3 ADMTYPE=INF PARMTYPE=1	K10= K12= K21= K13= K31= RATE= (or) DURN=	DOSE1= SCALE1=
Three compartments with infusion dose	NCOMPS=3 ADMTYPE=INF PARMTYPE=2	CL1= VOL1= CL2= VOL2= CL3= VOL3= RATE= (or) DURN=	DOSE1= SCALE1=
Three compartments with oral dose	NCOMPS=3 ADMTYPE=ORAL PARMTYPE=1	K10= K12= K21= K13= K31= Ka=	DOSE1= DOSE2= DOSE3= SCALE1= SCALE2= SCALE3= K20= K30= PCONC2= PCONC3=
Three compartments with oral dose	NCOMPS=3 ADMTYPE=ORAL PARMTYPE=2	CL1= VOL1= CL2= VOL2= CL3= VOL3= Ka=	DOSE1= DOSE2= DOSE3= SCALE1= SCALE2= SCALE3= K20= K30= PCONC2= PCONC3=

One-Compartment Model with an Oral Dose Administration

Consider the example "Example 88.1: One-Compartment Model with Pharmacokinetic Data" on page 7144, which studies the dispersion of the theophylline drug through a living individual. In this example, Pinheiro and Bates (1995) considered an one-compartment model with an oral dose. The paper mentions the solution to the one-compartment model. Instead of using the explicit solution, you can use the CMPTMODEL statement to fit the same model as follows:

```
proc nlmixed data=theoph;
   parms beta1=-3.22 beta2=0.47 beta3=-2.45
   s2b1 =0.03 cb12 =0 s2b2 =0.4 s2=0.5;
   cl = exp(beta1 + b1);
   ka = exp(beta2 + b2);
```

```
   ke = exp(beta3);
   vl = cl/ke;
   CMPTMODEL ncomps=1 admtype=oral time=time pconc=predConc
             parmtype=1  ka=ka k10=ke dose0=dose scale1=vl;
   model conc ~ normal(predConc,s2);
   random b1 b2 ~ normal([0,0],[s2b1,cb12,s2b2]) subject=subject;
run;
```

The NCOMPS=1 and ADMTYPE=ORAL options specify a one-compartment model with oral administration. Time is a data set variable that indicates time, and predConc is an outcome variable that contains predicted concentration. The PARMTYPE=1 option requests the compartment model using absorption and elimination rate constants, with KA=ka and K10=ke indicating the absorption and elimination rate constants, respectively. The dosage value for each patient in the depot compartment is specified in DOSE0=option, where dose is a data set variable. Lastly, SCALE1=vl scales the predicted concentrations by vl.

Multiple Doses

In PK field experiments or observational studies, a patient can often receive a drug multiple times, either continually or periodically over a period of time. The patient can also receive multiple types of drug in a study. For example, a patient might receive a bolus injection in the morning and an infusion drug at a constant rate in the evening. Calculation of predicted concentration in each compartment in the presence of these multiple doses or multiple types of dosing is different from single-dosing compartment models. For differential equations of a number of multiple-dosage scenarios and solutions to predicted concentration for the central compartment, see Gabrielsson and Weiner (2006).

The CMPTMODEL statement handles multiple doses in one-, two-, or three-compartment models and computes predicted concentrations in the central compartment. The syntax specification does not change from single-dose models, and the structure and content of the input data set are understood by the statement to fit various types of multiple-dosage models.

Within the biopharmaceutical industry, data in multiple-dosage studies are often structured by following a convention that has been popularized by NONMEM software (Beal et al. 2011). This convention is used to name variables and specify variable values; see Owen and Fiedler-Kelly (2014). If your data are available in a SAS data set that follows this convention, the data set must be converted to a SAS data set that is suitable to be analyzed using PROC NLMIXED. You can use the autocall macro %PKCONVRT for this purpose.

The %PKCONVRT macro takes two arguments: an input data set (specified by the first argument) and an output data set (specified by the second argument). The organization of the input data set should follow the convention. The output data set can serve as the DATA= data set for PROC NLMIXED to fit single-dose or multiple-dose models. Here is a simple example:

```
data pk_ex;
input ID  TIME  AMT DV  EVID;
datalines;
1     0.00    60000        .       1
1     0.20    50000        .       1
1     0.25        0    1126.1      0
1     0.50        0     869.9      0
1     0.75        0     883.6      0
1     1.00        0    1244.0      0
1     1.50        0     995.2      0
2     0.00    70000        .       1
```

```
2     0.25        0    1126.1      0
2     0.50        0     869.9      0
2     0.75        0     883.6      0
2     1.00        0    1244.0      0

   .... more lines ...
;
run;

%pkconvrt(data=pk_ex, out=out_ex);
```

If you want to fit a two-compartment model for the preceding multiple dosage data, use the following NLMIXED program with the out_ex data set that is produced from the %PKCONVRT macro:

```
proc nlmixed data=out_ex;
   parms beta1=2 beta2=2 beta3=2 beta4=2 s2=0.6
         s2b1=0.4 s2b2=0.4 s2b3=0.4 s2b4=0.4 cb21=0.001
         cb31=0.001 cb32=0.001 cb41=0.001 cb42=0.001 cb43=0.001;
   bounds s2b1>0, s2b2 >0, s2b3>0, s2b4>0, s2>0;
   cl1 = exp(beta1+b1);
   vl1 = exp(beta2+b2);
   cl2 = exp(beta3+b3);
   vl2 = exp(beta4+b4);
   CMPTMODEL ncomps=2 admtype=ivb parmtype=2 pconc=pred time=time
             vol1=vl1 cl1=cl1 vol2=vl2 cl2=cl2
             scale1=vl1;
   model conc ~ normal(pred,s2);
   random b1 b2 b3 b4 ~ normal([0,0,0,0],[s2b1,
                                          cb21,s2b2,
                                          cb31,cb32,s2b3,
                                          cb41,cb42,cb43,s2b4]) subject = id;
run;
```

The CMPTMODEL statement computes the predictions only for the central compartment for multiple dosage data. Therefore, options PCONC2=, PCONC3=, and PCONC0= are not valid in a multiple dose model. In addition, the K20=, K30=, DOSE2=, DOSE3=, SCALE2=, SCALE3=, SCALE0= options are ignored.

The %PKCONVRT autocall macro computes the elapsed time (from the time that a dose is administered to the time that a concentration is measured) in scenarios for both single and multiple continuous doses. The macro writes the computed time in the data set that is specified in the OUT= option. This data set also contains all the dosage information for all the time points at which the concentrations are measured. When you specify this data set in the PROC NLMIXED statement and you specify a CMPTMODEL statement, the elapsed times and the dosage information from the data set are used directly in computing the predicted concentrations in the central compartment. As a result, the CMPTMODEL statement ignores any specified DOSE1= and TIME= options. Similarly, for absorption models (ADMTYPE = ORAL), the DOSE0= and TIME= options in the CMPTMODEL statement are ignored and overwritten with the elapsed times and the dosage information from the OUT= data set. For examples that use the %PKCONVRT autocall macro and the CMPTMODEL statement to fit various compartment models, see Kurada and Chen (2018).

Starting with SAS/STAT 15.1, the CMPTMODEL statement supports steady-state dosing scenarios (Owen and Fiedler-Kelly 2014) in one-, two-, and three-compartment models. Corresponding enhancements have been made to the %PKCONVRT macro to handle these cases—the macro keeps track of the elapsed time and dosing information that are required in the CMPTMODEL statement to fit steady-state dose

models. The %PKCONVRT autocall macro supports input data sets that have combinations of steady-state and regular dosing, single and multiple dosing. For example, your data set can contain dosing history information on patients who receive multiple regular bolus doses and others who receive a single steady-state dose. When you use the output data set from the %PKCONVRT macro as input to PROC NLMIXED, the CMPTMODEL statement properly interprets all these distinct scenarios and computes the predicted concentrations accordingly.

The %PKCONVRT macro creates new variables (such as the _CMPT_ variable) in the OUT= data set. According to SAS conventions, it is best to avoid having variables whose name begins with an underscore _ in the DATA = data set. Also programming variables in the NLMIXED program that start with _CMPT_ are reserved for the internal computations; hence, avoid programming variables names that begin with _CMPT_ .

Hierarchical Model Specification

PROC NLMIXED supports multiple RANDOM statements to accommodate nested multilevel nonlinear mixed models, starting with SAS/STAT 13.2. If you use multiple RANDOM statements, PROC NLMIXED assumes that the SUBJECT= variable from each RANDOM statement forms a containment hierarchy. In the containment hierarchy, each SUBJECT= variable is contained by another SUBJECT= variable, and the SUBJECT= variable that is contained by all SUBJECT= variables is considered "the" SUBJECT= variable. For example, consider the following three-level nested model that has three SUBJECT= variables:

A	B	C
1	1	1
1	1	2
1	2	1
1	2	2
2	1	1
2	1	2
2	1	3
2	2	1
2	2	2

Suppose you specify a three-level nested model by using the following three RANDOM statements:

```
random r11 ~ normal(0,sd1) subject = A;
random r21 ~ normal(0,sd2) subject = B(A);
random r31 ~ normal(0,sd3) subject = C(A*B);
```

Then PROC NLMIXED assumes the containment hierarchy as follows. The first-level hierarchy is defined using "the" SUBJECT= variable A. Similarly, the second-level hierarchy is defined using SUBJECT= variable B, which is nested within first level. Finally, SUBJECT= variable C defines the third-level hierarchy that is nested within both the first and second levels. In short, the SUBJECT= variable A is "the" SUBJECT= variable. B is contained in A, and C is contained in both A and B. In this example, there are two first-level subjects that are determined using "the" SUBJECT= variable A.

Based on the preceding hierarchy specification, PROC NLMIXED's indexing of nested subjects for each first-level subject can be visualized as follows:

$$\overbrace{\overbrace{\underbrace{C_{1(11)}}_{B_{1(1)}}\underbrace{C_{2(11)}}_{B_{2(1)}}}^{A_1}\overbrace{\underbrace{C_{1(12)}}_{B_{1(1)}}\underbrace{C_{2(12)}}_{B_{2(1)}}}^{A_1}}^{A_1}\quad\overbrace{\overbrace{\underbrace{C_{1(21)}}_{B_{1(2)}}\underbrace{C_{2(21)}}_{B_{1(2)}}\underbrace{C_{3(21)}}_{B_{2(2)}}}^{A_2}\overbrace{\underbrace{C_{1(22)}}_{B_{2(2)}}\underbrace{C_{2(22)}}_{B_{2(2)}}}^{A_2}}^{A_2}$$

The *i*th subject from the first level is denoted as A_i, the second-level nested subjects are denoted as $\mathsf{B}_{j(i)}$, and the third-level nested subjects are denoted as $\mathsf{C}_{k(ij)}$.

You can specify any nested structure by using the SUBJECT= syntax in PROC NLMIXED. For example, using the following three RANDOM statements, PROC NLMIXED fits a different model:

```
random r11 ~ normal(0,sd1) subject = C;
random r21 ~ normal(0,sd2) subject = B(C);
random r31 ~ normal(0,sd3) subject = A(C*B);
```

In this case, PROC NLMIXED processes the subjects by using SUBJECT= variable C as "the" SUBJECT= variable, and the containment hierarchy is changed as follows:

A	B	C
1	1	1
1	1	2
1	2	1
1	2	2
2	1	1
2	1	2
2	1	3
2	2	1
2	2	2

$\Rightarrow$

C	B	A
1	1	1
1	1	2
1	2	1
1	2	2
2	1	1
2	1	2
2	2	1
2	2	2
3	1	2

Again, PROC NLMIXED's indexing of the nested subjects in this containment hierarchy can be visualized as follows:

$$\overbrace{\overbrace{\underbrace{A_{1(11)}}_{B_{1(1)}}\underbrace{A_{2(11)}}_{B_{2(1)}}}^{C_1}\overbrace{\underbrace{A_{1(12)}}_{B_{1(1)}}\underbrace{A_{2(12)}}_{B_{2(1)}}}^{C_1}}^{C_1}\quad\overbrace{\overbrace{\underbrace{A_{1(21)}}_{B_{1(2)}}\underbrace{A_{2(21)}}_{B_{1(2)}}}^{C_2}\overbrace{\underbrace{A_{1(22)}}_{B_{2(2)}}\underbrace{A_{2(22)}}_{B_{2(2)}}}^{C_2}}^{C_2}\quad\overbrace{\underbrace{A_{1(31)}}_{B_{1(3)}}}^{C_3}$$

Here, PROC NLMIXED assumes that C is "the" SUBJECT= variable. B is contained in C, and A is contained in C and B. In this case, there are three first-level subjects that are determined using "the" SUBJECT= variable C.

As explained before, in this case, the *i*th subject from the first level is denoted as C_i, the second-level nested subjects are denoted as $\mathsf{B}_{j(i)}$ and for third level, the nested subjects are denoted as $\mathsf{A}_{k(ij)}$.

Note that the containment hierarchy could potentially create more subjects than the unique number of subjects. For example, consider the following table, in which A is "the" SUBJECT= variable and B is nested within the subject A:

A	B
a	1
a	2
b	1
b	2
c	1
c	2

Even though the SUBJECT = B variable has only two unique subjects (1 and 2), when the containment hierarchy that is specified along with B is nested within A, PROC NLMIXED creates six nested B subjects. These nested subjects can be denoted as 1(a), 2(a), 1(b), 2(b), 1(c), and 2(c).

PROC NLMIXED does not support noncontainment hierarchy (or non-nested) models. For example, the following statements are not supported, because subject C is not nested within B and A:

```
random r11 ~ normal(0,sd1) subject = A;
random r21 ~ normal(0,sd2) subject = B(A);
random r31 ~ normal(0,sd3) subject = C;
```

Optimization Algorithms

There are several optimization techniques available in PROC NLMIXED. You can choose a particular optimizer with the TECH= option in the PROC NLMIXED statement.

Algorithm	TECH=
trust region method	TRUREG
Newton-Raphson method with line search	NEWRAP
Newton-Raphson method with ridging	NRRIDG
quasi-Newton methods (DBFGS, DDFP, BFGS, DFP)	QUANEW
double-dogleg method (DBFGS, DDFP)	DBLDOG
conjugate gradient methods (PB, FR, PR, CD)	CONGRA
Nelder-Mead simplex method	NMSIMP

No algorithm for optimizing general nonlinear functions exists that always finds the global optimum for a general nonlinear minimization problem in a reasonable amount of time. Since no single optimization technique is invariably superior to others, PROC NLMIXED provides a variety of optimization techniques that work well in various circumstances. However, you can devise problems for which none of the techniques in PROC NLMIXED can find the correct solution. Moreover, nonlinear optimization can be computationally expensive in terms of time and memory, so you must be careful when matching an algorithm to a problem.

All optimization techniques in PROC NLMIXED use $O(n^2)$ memory except the conjugate gradient methods, which use only $O(n)$ of memory and are designed to optimize problems with many parameters. Since the techniques are iterative, they require the repeated computation of the following:

- the function value (optimization criterion)

- the gradient vector (first-order partial derivatives)
- for some techniques, the (approximate) Hessian matrix (second-order partial derivatives)

However, since each of the optimizers requires different derivatives, some computational efficiencies can be gained. The following table shows, for each optimization technique, which derivatives are required (FOD: first-order derivatives; SOD: second-order derivatives).

Algorithm	FOD	SOD
TRUREG	x	x
NEWRAP	x	x
NRRIDG	x	x
QUANEW	x	-
DBLDOG	x	-
CONGRA	x	-
NMSIMP	-	-

Each optimization method employs one or more convergence criteria that determine when it has converged. The various termination criteria are listed and described in the "PROC NLMIXED Statement" section. 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 will converge if ABSGCONV< 1E–5, FCONV< $10^{-\text{FDIGITS}}$, or GCONV< 1E–8.

Choosing an Optimization Algorithm

The factors that go into choosing a particular optimization technique for a particular problem are complex and can involve trial and error.

For many optimization problems, computing the gradient takes more computer time than computing the function value, and 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, and 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 more easily terminate at stationary points rather than at global optima.

A few general remarks about the various optimization techniques follow:

- The second-derivative methods TRUREG, NEWRAP, and NRRIDG are best for small problems where 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 $n(n+1)/2$ double words; TRUREG and NEWRAP require two such matrices.
- The first-derivative methods QUANEW and DBLDOG are best for medium-sized problems where the objective function and the gradient are much faster to evaluate than the Hessian. The QUANEW and DBLDOG algorithms, in general, 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 (essentially one matrix with $n(n+1)/2$ double words). QUANEW is the default optimization method.

- The first-derivative method CONGRA is best for large problems where the objective function and the gradient can be computed much faster than the Hessian and where too much memory is required to store the (approximate) Hessian. The CONGRA algorithm, in general, requires more iterations than QUANEW or DBLDOG, but each iteration can be much faster. Since CONGRA requires only a factor of n double-word memory, many large applications of PROC NLMIXED can be solved only by CONGRA.
- The no-derivative method NMSIMP is best for small problems where derivatives are not continuous or are very difficult to compute.

Algorithm Descriptions

Some details about the optimization techniques follow.

Trust Region Optimization (TRUREG)

The trust region method uses the gradient $\mathbf{g}(\boldsymbol{\theta}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\theta}^{(k)})$; thus, it requires that the objective function $f(\boldsymbol{\theta})$ 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 with radius Δ that constrains the step size corresponding to the quality of the quadratic approximation. The trust region method is implemented using: Dennis, Gay, and Welsch (1981); Gay (1983); Moré 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 (dual) quasi-Newton or conjugate gradient algorithms might be more efficient.

Newton-Raphson Optimization with Line Search (NEWRAP)

The NEWRAP technique uses the gradient $\mathbf{g}(\boldsymbol{\theta}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\theta}^{(k)})$; 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 (LINESEARCH=2).

Newton-Raphson Ridge Optimization (NRRIDG)

The NRRIDG technique uses the gradient $\mathbf{g}(\boldsymbol{\theta}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\theta}^{(k)})$; 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 (dual) quasi-Newton or conjugate gradient algorithms might be more efficient.

Since the NRRIDG technique uses an orthogonal decomposition of the approximate Hessian, each iteration of NRRIDG can be slower than that of the NEWRAP technique, which works with Cholesky decomposition. Usually, however, NRRIDG requires fewer iterations than NEWRAP.

Quasi-Newton Optimization (QUANEW)

The (dual) quasi-Newton method uses the gradient $\mathbf{g}(\boldsymbol{\theta}^{(k)})$, and it does not need to compute second-order derivatives since they are approximated. It works well for medium to moderately large optimization problems where the objective function and the gradient are much faster to compute than the Hessian; but, 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 required for most nonlinear mixed model applications.

The QUANEW technique is one of the following, depending on the value of the UPDATE= option.

- the original quasi-Newton algorithm, which updates an approximation of the inverse Hessian
- the dual quasi-Newton algorithm, which updates the Cholesky factor of an approximate Hessian (default)

You can specify four update formulas with the UPDATE= option:

- DBFGS performs the dual Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update of the Cholesky factor of the Hessian matrix. This is the default.
- DDFP performs the dual Davidon, Fletcher, and Powell (DFP) update of the Cholesky factor of the Hessian matrix.
- BFGS performs the original BFGS update of the inverse Hessian matrix.
- DFP performs the original DFP update of the inverse Hessian matrix.

In each iteration, a line search is performed along the search direction to find an approximate optimum. The default line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α satisfying the Goldstein conditions. 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 with an identity matrix, resulting in the steepest descent or ascent search direction. You can specify line-search algorithms other than the default with the LINESEARCH= option.

The QUANEW algorithm uses its own line-search technique. No options and parameters (except the INSTEP= option) controlling the line search in the other algorithms apply here. In several applications, large steps in the first iterations are troublesome. You can use the INSTEP= option to impose an upper bound for the step size α during the first five iterations. You can also use the INHESSIAN=*r* option to specify a different starting approximation for the Hessian. If you specify only the INHESSIAN option, the Cholesky factor of a (possibly ridged) finite difference approximation of the Hessian is used to initialize the quasi-Newton update process. The values of the LCSINGULAR=, LCEPSILON=, and LCDEACT= options, which control the

processing of linear and boundary constraints, are valid only for the quadratic programming subroutine used in each iteration of the QUANEW algorithm.

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 $\mathbf{s}^{(k)}$ as the linear combination of the steepest descent or ascent search direction $\mathbf{s}_1^{(k)}$ and a quasi-Newton search direction $\mathbf{s}_2^{(k)}$:

$$\mathbf{s}^{(k)} = \alpha_1 \mathbf{s}_1^{(k)} + \alpha_2 \mathbf{s}_2^{(k)}$$

The step is requested to remain within a prespecified trust region radius; see Fletcher (1987, p. 107). Thus, the DBLDOG subroutine uses the dual quasi-Newton update but does not perform a line search. You can specify two update formulas with the UPDATE= option:

- DBFGS performs the dual Broyden, Fletcher, Goldfarb, and Shanno update of the Cholesky factor of the Hessian matrix. This is the default.
- DDFP performs the dual Davidon, Fletcher, and Powell update of the Cholesky factor of the Hessian matrix.

The double-dogleg optimization technique works well for medium 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) and 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 $O(n)$ memory for unconstrained optimization. In general, many iterations are required to obtain a precise solution, but each of the CONGRA iterations is computationally cheap. You can specify four different update formulas for generating the conjugate directions by using the UPDATE= option:

- PB performs the automatic restart update method of Powell (1977) and Beale (1972). This is the default.
- FR performs the Fletcher-Reeves update (Fletcher 1987).
- PR performs the Polak-Ribiere update (Fletcher 1987).
- CD performs a conjugate-descent update of Fletcher (1987).

The default, UPDATE=PB, behaved best in most test examples. You are advised to avoid the option UPDATE=CD, which behaved worst in most test examples.

The CONGRA subroutine should be used for optimization problems with large n. For the unconstrained or boundary constrained case, CONGRA requires only $O(n)$ bytes of working memory, whereas all other

optimization methods require order $O(n^2)$ bytes of working memory. During n successive iterations, uninterrupted by restarts or changes in the working set, the conjugate gradient algorithm computes a cycle of n 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 α satisfying 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 LINESEARCH= option.

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 $n \gg 40$.

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 adapting to the nonlinearities of the objective function, which contributes to an increased speed of convergence. It uses a special termination criterion.

Finite-Difference Approximations of Derivatives

The FD= and FDHESSIAN= options specify the use of finite-difference approximations of the derivatives. The FD= option specifies that all derivatives are approximated using function evaluations, and the FDHESSIAN= option specifies that second-order derivatives are approximated using gradient evaluations.

Computing derivatives by finite-difference approximations can be very time-consuming, especially for second-order derivatives based only on values of the objective function (FD= option). If analytical derivatives are difficult to obtain (for example, if a function is computed by an iterative process), you might consider one of the optimization techniques that use first-order derivatives only (QUANEW, DBLDOG, or CONGRA). In the expressions that follow, $\boldsymbol{\theta}$ denotes the parameter vector, h_i denotes the step size for the ith parameter, and $\mathbf{e}_i$ is a vector of zeros with a 1 in the ith position.

Forward-Difference Approximations

The forward-difference derivative approximations consume less computer time, but they are usually not as precise as approximations that use central-difference formulas.

- For first-order derivatives, n additional function calls are required:

$$g_i = \frac{\partial f}{\partial \theta_i} \approx \frac{f(\boldsymbol{\theta} + h_i \mathbf{e}_i) - f(\boldsymbol{\theta})}{h_i}$$

- For second-order derivatives based on function calls only (Dennis and Schnabel 1983, p. 80), $n + n^2/2$ additional function calls are required for dense Hessian:

$$\frac{\partial^2 f}{\partial \theta_i \partial \theta_j} \approx \frac{f(\boldsymbol{\theta} + h_i \mathbf{e}_i + h_j \mathbf{e}_j) - f(\boldsymbol{\theta} + h_i \mathbf{e}_i) - f(\boldsymbol{\theta} + h_j \mathbf{e}_j) + f(\boldsymbol{\theta})}{h_i h_j}$$

- For second-order derivatives based on gradient calls (Dennis and Schnabel 1983, p. 103), n additional gradient calls are required:

$$\frac{\partial^2 f}{\partial\theta_i \partial\theta_j} \approx \frac{g_i(\boldsymbol{\theta} + h_j \mathbf{e}_j) - g_i(\boldsymbol{\theta})}{2h_j} + \frac{g_j(\boldsymbol{\theta} + h_i \mathbf{e}_i) - g_j(\boldsymbol{\theta})}{2h_i}$$

Central-Difference Approximations

Central-difference approximations are usually more precise, but they consume more computer time than approximations that use forward-difference derivative formulas.

- For first-order derivatives, $2n$ additional function calls are required:

$$g_i = \frac{\partial f}{\partial\theta_i} \approx \frac{f(\boldsymbol{\theta} + h_i \mathbf{e}_i) - f(\boldsymbol{\theta} - h_i \mathbf{e}_i)}{2h_i}$$

- For second-order derivatives based on function calls only (Abramowitz and Stegun 1972, p. 884), $2n + 4n^2/2$ additional function calls are required.

$$\frac{\partial^2 f}{\partial\theta_i^2} \approx \frac{-f(\boldsymbol{\theta} + 2h_i \mathbf{e}_i) + 16f(\boldsymbol{\theta} + h_i \mathbf{e}_i) - 30f(\boldsymbol{\theta}) + 16f(\boldsymbol{\theta} - h_i e_i) - f(\boldsymbol{\theta} - 2h_i \mathbf{e}_i)}{12h_i^2}$$

$$\frac{\partial^2 f}{\partial\theta_i \partial\theta_j} \approx \frac{f(\boldsymbol{\theta} + h_i \mathbf{e}_i + h_j \mathbf{e}_j) - f(\boldsymbol{\theta} + h_i \mathbf{e}_i - h_j \mathbf{e}_j) - f(\boldsymbol{\theta} - h_i \mathbf{e}_i + h_j \mathbf{e}_j) + f(\boldsymbol{\theta} - h_i \mathbf{e}_i - h_j \mathbf{e}_j)}{4h_i h_j}$$

- For second-order derivatives based on gradient calls, $2n$ additional gradient calls are required:

$$\frac{\partial^2 f}{\partial\theta_i \partial\theta_j} \approx \frac{g_i(\boldsymbol{\theta} + h_j \mathbf{e}_j) - g_i(\boldsymbol{\theta} - h_j \mathbf{e}_j)}{4h_j} + \frac{g_j(\boldsymbol{\theta} + h_i \mathbf{e}_i) - g_j(\boldsymbol{\theta} - h_i \mathbf{e}_i)}{4h_i}$$

You can use the FDIGITS= option to specify the number of accurate digits in the evaluation of the objective function. This specification is helpful in determining an appropriate interval size h to be used in the finite-difference formulas.

The step sizes h_j, $j = 1, \ldots, n$ are defined as follows:

- For the forward-difference approximation of first-order derivatives that use function calls and second-order derivatives that use gradient calls, $h_j = \sqrt[2]{\eta}(1 + |\theta_j|)$.
- For the forward-difference approximation of second-order derivatives that use only function calls and all central-difference formulas, $h_j = \sqrt[3]{\eta}(1 + |\theta_j|)$.

The value of η is defined by the FDIGITS= option:

- If you specify the number of accurate digits by using FDIGITS=r, η is set to 10^{-r}.
- If you do not specify the FDIGITS= option, η is set to the machine precision ϵ.

Hessian Scaling

The rows and columns of the Hessian matrix can be scaled when you use the trust region, Newton-Raphson, and double-dogleg optimization techniques. Each element $H_{i,j}, i, j = 1, \ldots, n$ is divided by the scaling factor $d_i d_j$, where the scaling vector $d = (d_1, \ldots, d_n)$ is iteratively updated in a way specified by the HESCAL=*i* option, as follows:

$i = 0$: No scaling is done (equivalent to $d_i = 1$).

$i \neq 0$: First iteration and each restart iteration sets:

$$d_i^{(0)} = \sqrt{\max(|H_{i,i}^{(0)}|, \epsilon)}$$

$i = 1$: See Moré (1978):

$$d_i^{(k+1)} = \max\left[d_i^{(k)}, \sqrt{\max(|H_{i,i}^{(k)}|, \epsilon)}\right]$$

$i = 2$: See Dennis, Gay, and Welsch (1981):

$$d_i^{(k+1)} = \max\left[0.6 d_i^{(k)}, \sqrt{\max(|H_{i,i}^{(k)}|, \epsilon)}\right]$$

$i = 3$: d_i is reset in each iteration:

$$d_i^{(k+1)} = \sqrt{\max(|H_{i,i}^{(k)}|, \epsilon)}$$

In the preceding equations, ϵ is the relative machine precision or, equivalently, the largest double-precision value that, when added to 1, results in 1.

Active Set Methods

The parameter vector $\boldsymbol{\theta} \in \mathcal{R}^n$ can be subject to a set of m linear equality and inequality constraints:

$$\sum_{j=1}^{n} a_{ij}\theta_j = b_i \qquad i = 1, \ldots, m_e$$
$$\sum_{j=1}^{n} a_{ij}\theta_j \geq b_i \qquad i = m_e + 1, \ldots, m$$

The coefficients a_{ij} and right-hand sides b_i of the equality and inequality constraints are collected in the $m \times n$ matrix $\mathbf{A}$ and the m vector $\mathbf{b}$.

The m linear constraints define a feasible region $\mathcal{G}$ in $\mathcal{R}^n$ that must contain the point $\boldsymbol{\theta}_*$ that minimizes the problem. If the feasible region $\mathcal{G}$ is empty, no solution to the optimization problem exists.

In PROC NLMIXED, all optimization techniques use *active set methods*. The iteration starts with a feasible point $\boldsymbol{\theta}^{(0)}$, which you can provide or which can be computed by the Schittkowski and Stoer (1979) algorithm

implemented in PROC NLMIXED. The algorithm then moves from one feasible point $\boldsymbol{\theta}^{(k-1)}$ to a better feasible point $\boldsymbol{\theta}^{(k)}$ along a feasible search direction $\mathrm{s}^{(k)}$,

$$\boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}^{(k-1)} + \alpha^{(k)}\mathrm{s}^{(k)}, \quad \alpha^{(k)} > 0$$

Theoretically, the path of points $\boldsymbol{\theta}^{(k)}$ never leaves the feasible region $\mathcal{G}$ of the optimization problem, but it can reach its boundaries. The active set $\mathcal{A}^{(k)}$ of point $\boldsymbol{\theta}^{(k)}$ is defined as the index set of all linear equality constraints and those inequality constraints that are satisfied at $\boldsymbol{\theta}^{(k)}$. If no constraint is active $\boldsymbol{\theta}^{(k)}$, the point is located in the interior of $\mathcal{G}$, and the active set $\mathcal{A}^{(k)} = \emptyset$ is empty. If the point $\boldsymbol{\theta}^{(k)}$ in iteration k hits the boundary of inequality constraint i, this constraint i becomes active and is added to $\mathcal{A}^{(k)}$. Each equality constraint and each active inequality constraint reduce the dimension (degrees of freedom) of the optimization problem.

In practice, the active constraints can be satisfied only with finite precision. The LCEPSILON=*r* option specifies the range for active and violated linear constraints. If the point $\boldsymbol{\theta}^{(k)}$ satisfies the condition

$$\left| \sum_{j=1}^{n} a_{ij}\theta_j^{(k)} - b_i \right| \leq t$$

where $t = r(|b_i| + 1)$, the constraint i is recognized as an active constraint. Otherwise, the constraint i is either an inactive inequality or a violated inequality or equality constraint. Due to rounding errors in computing the projected search direction, error can be accumulated so that an iterate $\boldsymbol{\theta}^{(k)}$ steps out of the feasible region.

In those cases, PROC NLMIXED might try to pull the iterate $\boldsymbol{\theta}^{(k)}$ back into the feasible region. However, in some cases the algorithm needs to increase the feasible region by increasing the LCEPSILON=*r* value. If this happens, a message is displayed in the log output.

If the algorithm cannot improve the value of the objective function by moving from an active constraint back into the interior of the feasible region, it makes this inequality constraint an equality constraint in the next iteration. This means that the active set $\mathcal{A}^{(k+1)}$ still contains the constraint i. Otherwise, it releases the active inequality constraint and increases the dimension of the optimization problem in the next iteration.

A serious numerical problem can arise when some of the active constraints become (nearly) linearly dependent. PROC NLMIXED removes linearly dependent equality constraints before starting optimization. You can use the LCSINGULAR= option to specify a criterion r used in the update of the QR decomposition that determines whether an active constraint is linearly dependent relative to a set of other active constraints.

If the solution $\boldsymbol{\theta}^*$ is subjected to n_{act} linear equality or active inequality constraints, the QR decomposition of the $n \times n_{act}$ matrix $\hat{\mathbf{A}}'$ of the linear constraints is computed by $\hat{\mathbf{A}}' = \mathbf{QR}$, where $\mathbf{Q}$ is an $n \times n$ orthogonal matrix and $\mathbf{R}$ is an $n \times n_{act}$ upper triangular matrix. The n columns of matrix $\mathbf{Q}$ can be separated into two matrices, $\mathbf{Q} = [\mathbf{Y}, \mathbf{Z}]$, where $\mathbf{Y}$ contains the first n_{act} orthogonal columns of $\mathbf{Q}$ and $\mathbf{Z}$ contains the last $n - n_{act}$ orthogonal columns of $\mathbf{Q}$. The $n \times (n - n_{act})$ column-orthogonal matrix $\mathbf{Z}$ is also called the *null-space matrix* of the active linear constraints $\hat{\mathbf{A}}'$. The $n - n_{act}$ columns of the $n \times (n - n_{act})$ matrix $\mathbf{Z}$ form a basis orthogonal to the rows of the $n_{act} \times n$ matrix $\hat{\mathbf{A}}$.

At the end of the iterating, PROC NLMIXED computes the *projected gradient* g_Z,

$$\mathrm{g}_Z = \mathbf{Z}'\mathrm{g}$$

In the case of boundary-constrained optimization, the elements of the projected gradient correspond to the gradient elements of the free parameters. A necessary condition for $\boldsymbol{\theta}^*$ to be a local minimum of the

optimization problem is

$$\mathbf{g}_Z(\boldsymbol{\theta}^*) = \mathbf{Z}'\mathbf{g}(\boldsymbol{\theta}^*) = \mathbf{0}$$

The symmetric $n_{act} \times n_{act}$ matrix $\mathbf{G}_Z$,

$$\mathbf{G}_Z = \mathbf{Z}'\mathbf{G}\mathbf{Z}$$

is called a *projected Hessian matrix*. A second-order necessary condition for $\boldsymbol{\theta}^*$ to be a local minimizer requires that the projected Hessian matrix is positive semidefinite.

Those elements of the n_{act} vector of first-order estimates of *Lagrange multipliers*,

$$\lambda = (\hat{\mathbf{A}}\hat{\mathbf{A}}')^{-1}\hat{\mathbf{A}}\mathbf{Z}\mathbf{Z}'\mathbf{g}$$

that correspond to active inequality constraints indicate whether an improvement of the objective function can be obtained by releasing this active constraint. For minimization, a significant negative Lagrange multiplier indicates that a possible reduction of the objective function can be achieved by releasing this active linear constraint. The LCDEACT=*r* option specifies a threshold *r* for the Lagrange multiplier that determines whether an active inequality constraint remains active or can be deactivated. (In the case of boundary-constrained optimization, the Lagrange multipliers for active lower (upper) constraints are the negative (positive) gradient elements corresponding to the active parameters.)

Line-Search Methods

In each iteration k, the (dual) quasi-Newton, conjugate gradient, and Newton-Raphson minimization techniques use iterative line-search algorithms that try to optimize a linear, quadratic, or cubic approximation of f along a feasible descent search direction $\mathbf{s}^{(k)}$,

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \alpha^{(k)}\mathbf{s}^{(k)}, \quad \alpha^{(k)} > 0$$

by computing an approximately optimal scalar $\alpha^{(k)}$.

Therefore, a line-search algorithm is an iterative process that optimizes a nonlinear function $f(\alpha)$ of one parameter (α) within each iteration k of the optimization technique. Since the outside iteration process is based only on the approximation of the objective function, the inside iteration of the line-search algorithm does not have to be perfect. Usually, it is satisfactory that the choice of α significantly reduces (in a minimization) the objective function. Criteria often used for termination of line-search algorithms are the Goldstein conditions (see Fletcher 1987).

You can select various line-search algorithms by specifying the LINESEARCH= option. The line-search method LINESEARCH=2 seems to be superior when function evaluation consumes significantly less computation time than gradient evaluation. Therefore, LINESEARCH=2 is the default method for Newton-Raphson, (dual) quasi-Newton, and conjugate gradient optimizations.

You can modify the line-search methods LINESEARCH=2 and LINESEARCH=3 to be exact line searches by using the LSPRECISION= option and specifying the σ parameter described in Fletcher (1987). The line-search methods LINESEARCH=1, LINESEARCH=2, and LINESEARCH=3 satisfy the left-side and right-side Goldstein conditions (see Fletcher 1987). When derivatives are available, the line-search methods LINESEARCH=6, LINESEARCH=7, and LINESEARCH=8 try to satisfy the right-side Goldstein condition; if derivatives are not available, these line-search algorithms use only function calls.

Restricting the Step Length

Almost all line-search algorithms use iterative extrapolation techniques that can easily lead them to (feasible) points where the objective function f is no longer defined or is difficult to compute. Therefore, PROC NLMIXED provides options restricting the step length α or trust region radius Δ, especially during the first main iterations.

The inner product $\mathbf{g}'\mathbf{s}$ of the gradient $\mathbf{g}$ and the search direction $\mathbf{s}$ is the slope of $f(\alpha) = f(\boldsymbol{\theta} + \alpha\mathbf{s})$ along the search direction $\mathbf{s}$. The default starting value $\alpha^{(0)} = \alpha^{(k,0)}$ in each line-search algorithm ($\min_{\alpha>0} f(\boldsymbol{\theta} + \alpha\mathbf{s})$) during the main iteration k is computed in three steps:

1. The first step uses either the difference $|\Delta f| = |f^{(k)} - f^{(k-1)}|$ of the function values during the last two consecutive iterations or the final step-size value α^- of the last iteration $k-1$ to compute a first value of $\alpha_1^{(0)}$.

   - If the DAMPSTEP option is not used,

     $$\alpha_1^{(0)} = \begin{cases} step & \text{if } 0.1 \le step \le 10 \\ 10 & \text{if } step > 10 \\ 0.1 & \text{if } step < 0.1 \end{cases}$$

     with

     $$step = \begin{cases} |\Delta f|/|\mathbf{g}'\mathbf{s}| & \text{if } |\mathbf{g}'\mathbf{s}| \ge \epsilon \max(100 \times |\Delta f|, 1) \\ 1 & \text{otherwise} \end{cases}$$

     This value of $\alpha_1^{(0)}$ can be too large and can lead to a difficult or impossible function evaluation, especially for highly nonlinear functions such as the EXP function.

   - If the DAMPSTEP=*r* option is used,

     $$\alpha_1^{(0)} = \min(1, r\alpha^-)$$

     The initial value for the new step length can be no larger than r times the final step length α^- of the former iteration. The default value is $r = 2$.

2. During the first five iterations, the second step enables you to reduce $\alpha_1^{(0)}$ to a smaller starting value $\alpha_2^{(0)}$ by using the INSTEP=*r* option:

   $$\alpha_2^{(0)} = \min(\alpha_1^{(0)}, r)$$

   After more than five iterations, $\alpha_2^{(0)}$ is set to $\alpha_1^{(0)}$.

3. The third step can further reduce the step length by

   $$\alpha_3^{(0)} = \min(\alpha_2^{(0)}, \min(10, u))$$

   where u is the maximum length of a step inside the feasible region.

The INSTEP=*r* option enables you to specify a smaller or larger radius Δ of the trust region used in the first iteration of the trust region and double-dogleg algorithms. The default initial trust region radius $\Delta^{(0)}$ is the length of the scaled gradient (Moré 1978). This step corresponds to the default radius factor of $r = 1$. In most practical applications of the TRUREG and DBLDOG algorithms, this choice is successful. However, for bad initial values and highly nonlinear objective functions (such as the EXP function), the default start radius can result in arithmetic overflows. If this happens, you can try decreasing values of INSTEP=*r*, $0 < r < 1$, until the iteration starts successfully. A small factor r also affects the trust region radius $\Delta^{(k+1)}$ of the next steps because the radius is changed in each iteration by a factor $0 < c \leq 4$, depending on the ratio ρ expressing the goodness of quadratic function approximation. Reducing the radius Δ corresponds to increasing the ridge parameter λ, producing smaller steps aimed more closely toward the (negative) gradient direction.

Computational Problems

Floating-Point Errors and Overflows

Numerical optimization of a numerically integrated function is a difficult task, and the computation of the objective function and its derivatives can lead to arithmetic exceptions and overflows. A typical cause of these problems is parameters with widely varying scales. If the scaling of your parameters varies by more than a few orders of magnitude, the numerical stability of the optimization problem can be seriously reduced and can result in computational difficulties. A simple remedy is to rescale each parameter so that its final estimated value has a magnitude near 1.

If parameter rescaling does not help, consider the following actions:

- Specify the ITDETAILS option in the PROC NLMIXED statement to obtain more detailed information about when and where the problem is occurring.
- Provide different initial values or try a grid search of values.
- Use boundary constraints to avoid the region where overflows can happen.
- Delete outlying observations or subjects from the input data, if this is reasonable.
- Change the algorithm (specified in programming statements) that computes the objective function.

The line-search algorithms that work with cubic extrapolation are especially sensitive to arithmetic overflows. If an overflow occurs during a line search, you can use the INSTEP= option to reduce the length of the first trial step during the first five iterations, or you can use the DAMPSTEP or MAXSTEP option to restrict the step length of the initial α in subsequent iterations. If an arithmetic overflow occurs in the first iteration of the trust region or double-dogleg algorithm, you can use the INSTEP= option to reduce the default trust region radius of the first iteration. You can also change the optimization technique or the line-search method.

Long Run Times

PROC NLMIXED can take a long time to run for problems involving complex models, many parameters, or large input data sets. Although the optimization techniques used by PROC NLMIXED are some of the best ones available, they are not guaranteed to converge quickly for all problems. Ill-posed or misspecified models can cause the algorithms to use more extensive calculations designed to achieve convergence, and this can result in longer run times. So first make sure that your model is specified correctly, that your parameters are scaled to be of the same order of magnitude, and that your data reasonably match the model you are contemplating.

If you are using the default adaptive Gaussian quadrature algorithm and no iteration history is printing at all, then PROC NLMIXED might be bogged down trying to determine the number of quadrature points at the first set of starting values. Specifying the QPOINTS= option will bypass this stage and proceed directly to iterations; however, be aware that the likelihood approximation might not be accurate if there are too few quadrature points.

PROC NLMIXED can also have difficulty determining the number of quadrature points if the initial starting values are far from the optimum values. To obtain more accurate starting values for the model parameters, one easy method is to fit a model with no RANDOM statement. You can then use these estimates as starting values, although you will still need to specify values for the random-effects distribution. For normal-normal models, another strategy is to use METHOD=FIRO. If you can obtain estimates by using this approximate method, then they can be used as starting values for more accurate likelihood approximations.

If you are running PROC NLMIXED multiple times, you will probably want to include a statement like the following in your program:

```
ods output ParameterEstimates=pe;
```

This statement creates a SAS data set named PE upon completion of the run. In your next invocation of PROC NLMIXED, you can then specify

```
parms / data=pe;
```

to read in the previous estimates as starting values.

To speed general computations, you should double-check your programming statements to minimize the number of floating-point operations. Using auxiliary variables and factoring amenable expressions can be useful changes in this regard.

Problems Evaluating Code for Objective Function

The starting point $\boldsymbol{\theta}^{(0)}$ must be a point for which the programming statements can be evaluated. However, during optimization, the optimizer might iterate to a point $\boldsymbol{\theta}^{(k)}$ where the objective function or its derivatives cannot be evaluated. In some cases, the specification of boundary for parameters can avoid such situations. In many other cases, you can indicate that the point $\boldsymbol{\theta}^{(0)}$ is a bad point simply by returning an extremely large value for the objective function. In these cases, the optimization algorithm reduces the step length and stays closer to the point that has been evaluated successfully in the former iteration.

No Convergence

There are a number of things to try if the optimizer fails to converge.

- Change the initial values by using a grid search specification to obtain a set of good feasible starting values.
- Change or modify the update technique or the line-search algorithm.

  This method applies only to TECH=QUANEW and TECH=CONGRA. For example, if you use the default update formula and the default line-search algorithm, you can do the following:

  - change the update formula with the UPDATE= option
  - change the line-search algorithm with the LINESEARCH= option
  - specify a more precise line search with the LSPRECISION= option, if you use LINESEARCH=2 or LINESEARCH=3
- Change the optimization technique.

  For example, if you use the default option, TECH=QUANEW, you can try one of the second-derivative methods if your problem is small or the conjugate gradient method if it is large.
- Adjust finite-difference derivatives.

  The forward-difference derivatives specified with the FD= or FDHESSIAN= option might not be precise enough to satisfy strong gradient termination criteria. You might need to specify the more expensive central-difference formulas. The finite-difference intervals might be too small or too big, and the finite-difference derivatives might be erroneous.
- Double-check the data entry and program specification.

Convergence to Stationary Point

The gradient at a stationary point is the null vector, which always leads to a zero search direction. This point satisfies the first-order termination criterion. Search directions that are based on the gradient are zero, so the algorithm terminates. There are two ways to avoid this situation:

- Use the PARMS statement to specify a grid of feasible initial points.
- Use the OPTCHECK=*r* option to avoid terminating at the stationary point.

The signs of the eigenvalues of the (reduced) Hessian matrix contain the following information regarding a stationary point:

- If all of the eigenvalues are positive, the Hessian matrix is positive definite, and the point is a minimum point.
- If some of the eigenvalues are positive and all remaining eigenvalues are zero, the Hessian matrix is positive semidefinite, and the point is a minimum or saddle point.
- If all of the eigenvalues are negative, the Hessian matrix is negative definite, and the point is a maximum point.

- If some of the eigenvalues are negative and all remaining eigenvalues are zero, the Hessian matrix is negative semidefinite, and the point is a maximum or saddle point.
- If all of the eigenvalues are zero, the point can be a minimum, maximum, or saddle point.

Precision of Solution

In some applications, PROC NLMIXED can result in parameter values that are not precise enough. Usually, this means that the procedure terminated at a point too far from the optimal point. The termination criteria define the size of the termination region around the optimal point. Any point inside this region can be accepted for terminating the optimization process. The default values of the termination criteria are set to satisfy a reasonable compromise between the computational effort (computer time) and the precision of the computed estimates for the most common applications. However, there are a number of circumstances in which the default values of the termination criteria specify a region that is either too large or too small.

If the termination region is too large, then it can contain points with low precision. In such cases, you should determine which termination criterion stopped the optimization process. In many applications, you can obtain a solution with higher precision simply by using the old parameter estimates as starting values in a subsequent run in which you specify a smaller value for the termination criterion that was satisfied at the former run.

If the termination region is too small, the optimization process might take longer to find a point inside such a region, or it might not even find such a point due to rounding errors in function values and derivatives. This can easily happen in applications in which finite-difference approximations of derivatives are used and the GCONV and ABSGCONV termination criteria are too small to respect rounding errors in the gradient values.

Covariance Matrix

The estimated covariance matrix of the parameter estimates is computed as the inverse Hessian matrix, and for unconstrained problems it should be positive definite. If the final parameter estimates are subjected to $n_{act} > 0$ active linear inequality constraints, the formulas of the covariance matrices are modified similar to Gallant (1987) and Cramer (1986, p. 38) and additionally generalized for applications with singular matrices.

There are several steps available that enable you to tune the rank calculations of the covariance matrix.

1. You can use the ASINGULAR=, MSINGULAR=, and VSINGULAR= options to set three singularity criteria for the inversion of the Hessian matrix $\mathbf{H}$. The singularity criterion used for the inversion is

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

   where $d_{j,j}$ is the diagonal pivot of the matrix $\mathbf{H}$, and ASING, VSING, and MSING are the specified values of the ASINGULAR=, VSINGULAR=, and MSINGULAR= options, respectively. The default values are as follows:

   - ASING: the square root of the smallest positive double-precision value
   - MSING: 1E–12 if you do not specify the SINGHESS= option and $\max(10\epsilon, 1\text{E}{-}4{\times}\text{SINGHESS})$ otherwise, where ϵ is the machine precision
   - VSING: 1E–8 if you do not specify the SINGHESS= option and the value of SINGHESS otherwise

Note that, in many cases, a normalized matrix $\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}$ is decomposed, and the singularity criteria are modified correspondingly.

2. If the matrix $\mathbf{H}$ is found to be singular in the first step, a generalized inverse is computed. Depending on the G4= option, either a generalized inverse satisfying all four Moore-Penrose conditions is computed (a g_4-inverse) or a generalized inverse satisfying only two Moore-Penrose conditions is computed (a g_2-inverse, Pringle and Rayner 1971). If the number of parameters n of the application is less than or equal to G4=*i*, a g_4-inverse is computed; otherwise, only a g_2-inverse is computed. The g_4-inverse is computed by the (computationally very expensive but numerically stable) eigenvalue decomposition, and the g_2-inverse is computed by Gauss transformation. The g_4-inverse is computed using the eigenvalue decomposition $\mathbf{A} = \mathbf{Z}\mathbf{\Lambda}\mathbf{Z}'$, where $\mathbf{Z}$ is the orthogonal matrix of eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues, $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. The g_4-inverse of $\mathbf{H}$ is set to

$$\mathbf{A}^- = \mathbf{Z}\mathbf{\Lambda}^-\mathbf{Z}'$$

where the diagonal matrix $\mathbf{\Lambda}^- = \mathrm{diag}(\lambda_1^-, \ldots, \lambda_n^-)$ is defined using the COVSING= option:

$$\lambda_i^- = \begin{cases} 1/\lambda_i & \text{if } |\lambda_i| > \text{COVSING} \\ 0 & \text{if } |\lambda_i| \leq \text{COVSING} \end{cases}$$

If you do not specify the COVSING= option, the nr smallest eigenvalues are set to zero, where nr is the number of rank deficiencies found in the first step.

For optimization techniques that do not use second-order derivatives, the covariance matrix is computed using finite-difference approximations of the derivatives.

Prediction

The nonlinear mixed model is a useful tool for statistical prediction. Assuming a prediction is to be made regarding the *i*th subject, suppose that $f(\boldsymbol{\theta}, \mathbf{u}_i)$ is a differentiable function predicting some quantity of interest. Recall that $\boldsymbol{\theta}$ denotes the vector of unknown parameters and $\mathbf{u}_i$ denotes the vector of random effects for the *i*th subject. A natural point prediction is $f(\widehat{\boldsymbol{\theta}}, \widehat{\mathbf{u}}_i)$, where $\widehat{\boldsymbol{\theta}}$ is the maximum likelihood estimate of $\boldsymbol{\theta}$ and $\widehat{\mathbf{u}}_i$ is the empirical Bayes estimate of $\mathbf{u}_i$ described previously in the section "Integral Approximations" on page 7107.

An approximate prediction variance matrix for $(\widehat{\boldsymbol{\theta}}, \widehat{\mathbf{u}}_i)$ is

$$\mathbf{P} = \begin{bmatrix} \widehat{\mathbf{H}}^{-1} & \widehat{\mathbf{H}}^{-1}\left(\frac{\partial \widehat{\mathbf{u}}_i}{\partial \boldsymbol{\theta}}\right)' \\ \left(\frac{\partial \widehat{\mathbf{u}}_i}{\partial \boldsymbol{\theta}}\right)\widehat{\mathbf{H}}^{-1} & \widehat{\mathbf{\Gamma}}^{-1} + \left(\frac{\partial \widehat{\mathbf{u}}_i}{\partial \boldsymbol{\theta}}\right)\widehat{\mathbf{H}}^{-1}\left(\frac{\partial \widehat{\mathbf{u}}_i}{\partial \boldsymbol{\theta}}\right)' \end{bmatrix}$$

where $\widehat{\mathbf{H}}$ is the approximate Hessian matrix from the optimization for $\widehat{\boldsymbol{\theta}}$, $\widehat{\mathbf{\Gamma}}$ is the approximate Hessian matrix from the optimization for $\widehat{\mathbf{u}}_i$, and $(\partial \widehat{\mathbf{u}}_i / \partial \boldsymbol{\theta})$ is the derivative of $\widehat{\mathbf{u}}_i$ with respect to $\boldsymbol{\theta}$, evaluated at $(\widehat{\boldsymbol{\theta}}, \widehat{\mathbf{u}}_i)$. The approximate variance matrix for $\widehat{\boldsymbol{\theta}}$ is the standard one discussed in the previous section, and that for $\widehat{\mathbf{u}}_i$ is an approximation to the conditional mean squared error of prediction described by Booth and Hobert (1998).

The prediction variance for a general scalar function $f(\boldsymbol{\theta}, \mathbf{u}_i)$ is defined as the expected squared difference $E[f(\widehat{\boldsymbol{\theta}}, \widehat{\mathbf{u}}_i) - f(\boldsymbol{\theta}, \mathbf{u}_i)]^2$. PROC NLMIXED computes an approximation to it as follows. The derivative of $f(\boldsymbol{\theta}, \mathbf{u}_i)$ is computed with respect to each element of $(\boldsymbol{\theta}, \mathbf{u}_i)$ and evaluated at $(\widehat{\boldsymbol{\theta}}, \widehat{\mathbf{u}}_i)$. If $\mathbf{a}_i$ is the resulting vector, then the approximate prediction variance is $\mathbf{a}_i'\mathbf{P}\mathbf{a}_i$. This approximation is known as the delta method (Billingsley 1986; Cox 1998).

Computational Resources

Since nonlinear optimization is an iterative process that depends on many factors, it is difficult to estimate how much computer time is necessary to find an optimal solution satisfying one of the termination criteria. You can use the MAXTIME=, MAXITER=, and MAXFUNC= options to restrict the amount of CPU time, the number of iterations, and the number of function calls in a single run of PROC NLMIXED.

In each iteration k, the NRRIDG technique uses a symmetric Householder transformation to decompose the $n \times n$ Hessian matrix $\mathbf{H}$,

$$\mathbf{H} = \mathbf{V}'\mathbf{T}\mathbf{V}, \qquad \mathbf{V}\text{: orthogonal}, \quad \mathbf{T}\text{: tridiagonal}$$

to compute the (Newton) search direction $\mathbf{s}$,

$$\mathbf{s}^{(k)} = -[\mathbf{H}^{(k)}]^{-1}\mathbf{g}^{(k)} \qquad k = 1, 2, 3, \ldots$$

The TRUREG and NEWRAP techniques use the Cholesky decomposition to solve the same linear system while computing the search direction. The QUANEW, DBLDOG, CONGRA, and NMSIMP techniques do not need to invert or decompose a Hessian matrix; thus, they require less computational resources than the other techniques.

The larger the problem, the more time is needed to compute function values and derivatives. Therefore, you might want to compare optimization techniques by counting and comparing the respective numbers of function, gradient, and Hessian evaluations.

Finite-difference approximations of the derivatives are expensive because they require additional function or gradient calls:

- forward-difference formulas
  - For first-order derivatives, n additional function calls are required.
  - For second-order derivatives based on function calls only, for a dense Hessian, $n+n^2/2$ additional function calls are required.
  - For second-order derivatives based on gradient calls, n additional gradient calls are required.
- central-difference formulas
  - For first-order derivatives, $2n$ additional function calls are required.
  - For second-order derivatives based on function calls only, for a dense Hessian, $2n+2n^2$ additional function calls are required.
  - For second-order derivatives based on gradient calls, $2n$ additional gradient calls are required.

Many applications need considerably more time for computing second-order derivatives (Hessian matrix) than for computing first-order derivatives (gradient). In such cases, a dual quasi-Newton technique is recommended, which does not require second-order derivatives.

Displayed Output

This section describes the displayed output from PROC NLMIXED. See the section "ODS Table Names" on page 7143 for details about how this output interfaces with the Output Delivery System.

Specifications

The NLMIXED procedure first displays the "Specifications" table, listing basic information about the nonlinear mixed model that you have specified. It includes the principal variables and estimation methods.

Dimensions

The "Dimensions" table lists counts of important quantities in your nonlinear mixed model, including the number of observations, subjects, parameters, and quadrature points.

Parameters

The "Parameters" table displays the information you provided with the PARMS statement and the value of the negative log-likelihood function evaluated at the starting values.

Starting Gradient and Hessian

The START option in the PROC NLMIXED statement displays the gradient of the negative log-likelihood function at the starting values of the parameters. If you also specify the HESS option, then the starting Hessian is displayed as well.

Iterations

The iteration history consists of one line of output for each iteration in the optimization process. The iteration history is displayed by default because it is important that you check for possible convergence problems. The default iteration history includes the following variables:

- Iter, the iteration number
- Calls, the number of function calls
- NegLogLike, the value of the objective function
- Diff, the difference between adjacent function values
- MaxGrad, the maximum of the absolute (projected) gradient components (except NMSIMP)
- Slope, the slope $\mathbf{g}'\mathbf{s}$ of the search direction $\mathbf{s}$ at the current parameter iterate $\boldsymbol{\theta}^{(k)}$ (QUANEW only)
- Rho, the ratio between the achieved and predicted values of Diff (NRRIDG only)
- Radius, the radius of the trust region (TRUREG only)
- StdDev, the standard deviation of the simplex values (NMSIMP only)
- Delta, the vertex length of the simplex (NMSIMP only)
- Size, the size of the simplex (NMSIMP only)

For the QUANEW method, the value of Slope should be significantly negative. Otherwise, the line-search algorithm has difficulty reducing the function value sufficiently. If this difficulty is encountered, an asterisk (*) appears after the iteration number. If there is a tilde ~ after the iteration number, the BFGS update is skipped, and very high values of the Lagrange function are produced. A backslash (\ ) after the iteration number indicates that Powell's correction for the BFGS update is used.

For methods using second derivatives, an asterisk (*) after the iteration number means that the computed Hessian approximation was singular and had to be ridged with a positive value.

For the NMSIMP method, only one line is displayed for several internal iterations. This technique skips the output for some iterations because some of the termination tests (StdDev and Size) are rather time-consuming compared to the simplex operations, and they are performed only every five simplex operations.

The ITDETAILS option in the PROC NLMIXED statement provides a more detailed iteration history. Besides listing the current values of the parameters and their gradients, the ITDETAILS option provides the following values in addition to the default output:

- Restart, the number of iteration restarts
- Active, the number of active constraints
- Lambda, the value of the Lagrange multiplier (TRUREG and DBLDOG only)
- Ridge, the ridge value (NRRIDG only)
- Alpha, the line-search step size (QUANEW only)

An apostrophe (') trailing the number of active constraints indicates that at least one of the active constraints was released from the active set due to a significant Lagrange multiplier.

Convergence Status

The "Convergence Status" table contains a status message describing the reason for termination of the optimization. The ODS name of this table is ConvergenceStatus, and you can query the nonprinting numeric variable Status to check for a successful optimization. This is useful in batch processing, or when processing BY groups, for example, in simulations. Successful convergence is indicated by Status=0.

Fitting Information

The "Fitting Information" table lists the final minimized value of –2 times the log likelihood as well as the information criteria of Akaike (AIC) and Schwarz (BIC), as well as a finite-sample corrected version of AIC (AICC). The criteria are computed as follows:

$$\begin{aligned} \text{AIC} &= 2f(\widehat{\boldsymbol{\theta}}) + 2p \\ \text{AICC} &= 2f(\widehat{\boldsymbol{\theta}}) + 2pn/(n-p-1) \\ \text{BIC} &= 2f(\widehat{\boldsymbol{\theta}}) + p\log(s) \end{aligned}$$

where $f()$ is the negative of the marginal log-likelihood function, $\widehat{\boldsymbol{\theta}}$ is the vector of parameter estimates, p is the number of parameters, n is the number of observations, and s is the number of subjects. See Hurvich and Tsai (1989) and Burnham and Anderson (1998) for additional details.

Parameter Estimates

The "Parameter Estimates" table lists the estimates of the parameter values after successful convergence of the optimization problem or the final values of the parameters under nonconvergence. If the problem did converge, standard errors are computed from the final Hessian matrix. The ratio of the estimate with its standard error produces a t value, with approximate degrees of freedom computed as the number of subjects minus the number of random effects. A p-value and confidence limits based on this t distribution are also provided. Finally, the gradient of the negative log-likelihood function is displayed for each parameter, and you should verify that they each are sufficiently small for nonconstrained parameters.

Covariance and Correlation Matrices

Following standard maximum likelihood theory (for example, Serfling 1980), the asymptotic variance-covariance matrix of the parameter estimates equals the inverse of the Hessian matrix. You can display this matrix with the COV option in the PROC NLMIXED statement. The corresponding correlation form is available with the CORR option.

Additional Estimates

The "Additional Estimates" table displays the results of all ESTIMATE statements that you specify, with the same columns as the "Parameter Estimates" table. The ECOV and ECORR options in the PROC NLMIXED statement produce tables displaying the approximate covariance and correlation matrices of the additional estimates. They are computed using the delta method (Billingsley 1986; Cox 1998). The EDER option in the PROC NLMIXED statement produces a table that displays the derivatives of the additional estimates with respect to the model parameters evaluated at their final estimated values.

ODS Table Names

PROC NLMIXED assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 88.5. For more information about ODS, see Chapter 22, "Using the Output Delivery System."

Table 88.5 ODS Tables Produced by PROC NLMIXED

ODS Table Name	Description	Statement or Option
AdditionalEstimates	Results from ESTIMATE statements	ESTIMATE
Contrasts	Results from CONTRAST statements	CONTRAST
ConvergenceStatus	Convergence status	default
CorrMatAddEst	Correlation matrix of additional estimates	ECORR
CorrMatParmEst	Correlation matrix of parameter estimates	CORR
CovMatAddEst	Covariance matrix of additional estimates	ECOV
CovMatParmEst	Covariance matrix of parameter estimates	COV
DerAddEst	Derivatives of additional estimates	EDER
Dimensions	Dimensions of the problem	default
FitStatistics	Fit statistics	default
Hessian	Second derivative matrix	HESS
IterHistory	Iteration history	default
Parameters	Initial parameters	default
ParameterEstimates	Parameter estimates	default

Table 88.5 *continued*

ODS Table Name	Description	Statement or Option
Specifications	Model specifications	default
StartingHessian	Starting Hessian matrix	START HESS
StartingValues	Starting values and gradient	START

Examples: NLMIXED Procedure

Example 88.1: One-Compartment Model with Pharmacokinetic Data

A popular application of nonlinear mixed models is in the field of pharmacokinetics, which studies how a drug disperses through a living individual. This example considers the theophylline data from Pinheiro and Bates (1995). Serum concentrations of the drug theophylline are measured in 12 subjects over a 25-hour period after oral administration. The data are as follows.

```
data theoph;
   input subject time conc dose wt;
   datalines;
 1  0.00  0.74 4.02 79.6
 1  0.25  2.84 4.02 79.6
 1  0.57  6.57 4.02 79.6
 1  1.12 10.50 4.02 79.6
 1  2.02  9.66 4.02 79.6
 1  3.82  8.58 4.02 79.6
 1  5.10  8.36 4.02 79.6
 1  7.03  7.47 4.02 79.6

   ... more lines ...

12 24.15  1.17 5.30 60.5
;
```

Pinheiro and Bates (1995) consider the following first-order compartment model for these data:

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

where C_{it} is the observed concentration of the ith subject at time t, D is the dose of theophylline, k_{e_i} is the elimination rate constant for subject i, k_{a_i} is the absorption rate constant for subject i, Cl_i is the clearance for subject i, and e_{it} are normal errors. To allow for random variability between subjects, they assume

$$\begin{aligned} Cl_i &= \exp(\beta_1 + b_{i1}) \\ k_{a_i} &= \exp(\beta_2 + b_{i2}) \\ k_{e_i} &= \exp(\beta_3) \end{aligned}$$

where the βs denote fixed-effects parameters and the b_is denote random-effects parameters with an unknown covariance matrix.

The PROC NLMIXED statements to fit this model are as follows:

```
proc nlmixed data=theoph;
   parms beta1=-3.22 beta2=0.47 beta3=-2.45
         s2b1 =0.03  cb12 =0    s2b2 =0.4 s2=0.5;
   cl   = exp(beta1 + b1);
   ka   = exp(beta2 + b2);
   ke   = exp(beta3);
   pred = dose*ke*ka*(exp(-ke*time)-exp(-ka*time))/cl/(ka-ke);
   model conc ~ normal(pred,s2);
   random b1 b2 ~ normal([0,0],[s2b1,cb12,s2b2]) subject=subject;
run;
```

The PARMS statement specifies starting values for the three βs and four variance-covariance parameters. The clearance and rate constants are defined using SAS programming statements, and the conditional model for the data is defined to be normal with mean pred and variance s2. The two random effects are b1 and b2, and their joint distribution is defined in the RANDOM statement. Brackets are used in defining their mean vector (two zeros) and the lower triangle of their variance-covariance matrix (a general 2×2 matrix). The SUBJECT= variable is subject.

The results from this analysis are as follows.

Output 88.1.1 Model Specification for One-Compartment Model

The NLMIXED Procedure

Specifications	
Data Set	WORK.THEOPH
Dependent Variable	conc
Distribution for Dependent Variable	Normal
Random Effects	b1 b2
Distribution for Random Effects	Normal
Subject Variable	subject
Optimization Technique	Dual Quasi-Newton
Integration Method	Adaptive Gaussian Quadrature

The "Specifications" table lists the setup of the model (Output 88.1.1). The "Dimensions" table indicates that there are 132 observations, 12 subjects, and 7 parameters. PROC NLMIXED selects 5 quadrature points for each random effect, producing a total grid of 25 points over which quadrature is performed (Output 88.1.2).

Output 88.1.2 Dimensions Table for One-Compartment Model

Dimensions	
Observations Used	132
Observations Not Used	0
Total Observations	132
Subjects	12
Max Obs per Subject	11
Parameters	7
Quadrature Points	5

The "Parameters" table lists the 7 parameters, their starting values, and the initial evaluation of the negative log likelihood using adaptive Gaussian quadrature (Output 88.1.3). The "Iteration History" table indicates that 10 steps are required for the dual quasi-Newton algorithm to achieve convergence.

Output 88.1.3 Starting Values and Iteration History

Initial Parameters

beta1	beta2	beta3	s2b1	cb12	s2b2	s2	Negative Log Likelihood
-3.22	0.47	-2.45	0.03	0	0.4	0.5	177.789945

Iteration History

Iteration	Calls	Negative Log Likelihood	Difference	Maximum Gradient	Slope
1	7	177.7762	0.013697	2.87337	-63.0744
2	11	177.7643	0.011948	1.69814	-4.75239
3	14	177.7573	0.007036	1.29744	-1.97311
4	17	177.7557	0.001576	1.44141	-0.49772
5	20	177.7467	0.008988	1.13228	-0.82230
6	24	177.7464	0.000299	0.83129	-0.00244
7	27	177.7463	0.000083	0.72420	-0.00789
8	31	177.7457	0.000578	0.18002	-0.00583
9	34	177.7457	3.88E-6	0.017958	-8.25E-6
10	37	177.7457	3.222E-8	0.000143	-6.51E-8

NOTE: GCONV convergence criterion satisfied.

Output 88.1.4 Fit Statistics for One-Compartment Model

Fit Statistics	
-2 Log Likelihood	355.5
AIC (smaller is better)	369.5
AICC (smaller is better)	370.4
BIC (smaller is better)	372.9

The "Fit Statistics" table lists the final optimized values of the log-likelihood function and information criteria in the "smaller is better" form (Output 88.1.4).

The "Parameter Estimates" table contains the maximum likelihood estimates of the parameters (Output 88.1.5). Both s2b1 and s2b2 are marginally significant, indicating between-subject variability in the clearances and absorption rate constants, respectively. There does not appear to be a significant covariance between them, as seen by the estimate of cb12.

Output 88.1.5 Parameter Estimates for One-Compartment Model

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|	95% Confidence Limits		Gradient
beta1	-3.2268	0.05950	10	-54.23	<.0001	-3.3594	-3.0942	-0.00009
beta2	0.4806	0.1989	10	2.42	0.0363	0.03745	0.9238	3.645E-7
beta3	-2.4592	0.05126	10	-47.97	<.0001	-2.5734	-2.3449	0.000039
s2b1	0.02803	0.01221	10	2.30	0.0446	0.000828	0.05524	-0.00014
cb12	-0.00127	0.03404	10	-0.04	0.9710	-0.07712	0.07458	-0.00007
s2b2	0.4331	0.2005	10	2.16	0.0561	-0.01354	0.8798	-6.98E-6
s2	0.5016	0.06837	10	7.34	<.0001	0.3493	0.6540	6.133E-6

The estimates of β_1, β_2, and β_3 are close to the adaptive quadrature estimates listed in Table 3 of Pinheiro and Bates (1995). However, Pinheiro and Bates use a Cholesky-root parameterization for the random-effects variance matrix and a logarithmic parameterization for the residual variance. The PROC NLMIXED statements using their parameterization are as follows, and results are similar.

```
proc nlmixed data=theoph;
   parms l11=-1.5 l2=0 l13=-0.1 beta1=-3 beta2=0.5 beta3=-2.5 ls2=-0.7;
   s2   = exp(ls2);
   l1   = exp(l11);
   l3   = exp(l13);
   s2b1 = l1*l1*s2;
   cb12 = l2*l1*s2;
   s2b2 = (l2*l2 + l3*l3)*s2;
   cl   = exp(beta1 + b1);
   ka   = exp(beta2 + b2);
   ke   = exp(beta3);
   pred = dose*ke*ka*(exp(-ke*time)-exp(-ka*time))/cl/(ka-ke);
   model conc   ~ normal(pred,s2);
   random b1 b2 ~ normal([0,0],[s2b1,cb12,s2b2]) subject=subject;
run;
```

Example 88.2: Probit-Normal Model with Binomial Data

For this example, consider the data from Weil (1970), also studied by: Williams (1975); Ochi and Prentice (1984); McCulloch (1994). In this experiment 16 pregnant rats receive a control diet and 16 receive a chemically treated diet, and the litter size for each rat is recorded after 4 and 21 days. The SAS data set follows:

```
data rats;
   input trt $ m x @@;
   if (trt='c') then do;
      x1 = 1;
      x2 = 0;
   end;
   else do;
      x1 = 0;
```

```
      x2 = 1;
   end;
   litter = _n_;
   datalines;
c 13 13   c 12 12   c  9  9   c  9  9   c  8  8   c  8  8   c 13 12   c 12 11
c 10  9   c 10  9   c  9  8   c 13 11   c  5  4   c  7  5   c 10  7   c 10  7
t 12 12   t 11 11   t 10 10   t  9  9   t 11 10   t 10  9   t 10  9   t  9  8
t  9  8   t  5  4   t  9  7   t  7  4   t 10  5   t  6  3   t 10  3   t  7  0
;
```

Here, m represents the size of the litter after 4 days, and x represents the size of the litter after 21 days. Also, indicator variables x1 and x2 are constructed for the two treatment levels.

Following McCulloch (1994), assume a latent survival model of the form

$$y_{ijk} = t_i + \alpha_{ij} + e_{ijk}$$

where i indexes treatment, j indexes litter, and k indexes newborn rats within a litter. The t_i represent treatment means, the α_{ij} represent random litter effects assumed to be iid $N(0, s_i^2)$, and the e_{ijk} represent iid residual errors, all on the latent scale.

Instead of observing the survival times y_{ijk}, assume that only the binary variable indicating whether y_{ijk} exceeds 0 is observed. If x_{ij} denotes the sum of these binary variables for the ith treatment and the jth litter, then the preceding assumptions lead to the following generalized linear mixed model:

$$x_{ij}|\alpha_{ij} \sim \text{Binomial}(m_{ij}, p_{ij})$$

where m_{ij} is the size of each litter after 4 days and

$$p_{ij} = \Phi(t_i + \alpha_{ij})$$

The PROC NLMIXED statements to fit this model are as follows:

```
proc nlmixed data=rats;
   parms t1=1 t2=1 s1=.05 s2=1;
   eta = x1*t1 + x2*t2 + alpha;
   p   = probnorm(eta);
   model x ~ binomial(m,p);
   random alpha ~ normal(0,x1*s1*s1+x2*s2*s2) subject=litter;
   estimate 'gamma2' t2/sqrt(1+s2*s2);
   predict p out=p;
run;
```

As in Example 88.1, the PROC NLMIXED statement invokes the procedure and the PARMS statement defines the parameters. The parameters for this example are the two treatment means, t1 and t2, and the two random-effect standard deviations, s1 and s2.

The indicator variables x1 and x2 are used in the program to assign the proper mean to each observation in the input data set as well as the proper variance to the random effects. Note that programming expressions are permitted inside the distributional specifications, as illustrated by the random-effects variance specified here.

The ESTIMATE statement requests an estimate of $\gamma_2 = t_2/\sqrt{1 + s_2^2}$, which is a location-scale parameter from Ochi and Prentice (1984).

The PREDICT statement constructs predictions for each observation in the input data set. For this example, predictions of p and approximate standard errors of prediction are output to a SAS data set named P. These predictions are functions of the parameter estimates and the empirical Bayes estimates of the random effects α_i.

The output for this model is as follows.

Output 88.2.1 Specifications, Dimensions, and Starting Values

The NLMIXED Procedure

Specifications	
Data Set	WORK.RATS
Dependent Variable	x
Distribution for Dependent Variable	Binomial
Random Effects	alpha
Distribution for Random Effects	Normal
Subject Variable	litter
Optimization Technique	Dual Quasi-Newton
Integration Method	Adaptive Gaussian Quadrature

Dimensions	
Observations Used	32
Observations Not Used	0
Total Observations	32
Subjects	32
Max Obs per Subject	1
Parameters	4
Quadrature Points	7

Initial Parameters

t1	t2	s1	s2	Negative Log Likelihood
1	1	0.05	1	54.9362323

The "Specifications" table provides basic information about this nonlinear mixed model (Output 88.2.1). The "Dimensions" table provides counts of various variables. Note that each observation in the data comprises a separate subject. Using the starting values in the "Parameters" table, PROC NLMIXED determines that the log-likelihood function can be approximated with sufficient accuracy with a seven-point quadrature rule.

Output 88.2.2 Iteration History for Probit-Normal Model

Iteration History

Iteration	Calls	Negative Log Likelihood	Difference	Maximum Gradient	Slope
1	4	53.9933934	0.942839	11.0326	-81.9428
2	6	52.8753530	1.11804	2.14895	-2.86277
3	9	52.6350386	0.240314	0.32996	-1.05049
4	11	52.6319939	0.003045	0.12293	-0.00672
5	14	52.6313583	0.000636	0.028246	-0.00352
6	18	52.6313174	0.000041	0.013551	-0.00023
7	21	52.6313115	5.839E-6	0.000603	-0.00001
8	24	52.6313115	9.45E-9	0.000022	-1.68E-8

NOTE: GCONV convergence criterion satisfied.

The "Iteration History" table indicates successful convergence in 8 iterations (Output 88.2.2).

Output 88.2.3 Fit Statistics for Probit-Normal Model

Fit Statistics

-2 Log Likelihood	105.3
AIC (smaller is better)	113.3
AICC (smaller is better)	114.7
BIC (smaller is better)	119.1

The "Fit Statistics" table lists useful statistics based on the maximized value of the log likelihood (Output 88.2.3).

Output 88.2.4 Parameter Estimates for Probit-Normal Model

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|	95% Confidence Limits		Gradient
t1	1.3063	0.1685	31	7.75	<.0001	0.9626	1.6499	-0.00002
t2	0.9475	0.3055	31	3.10	0.0041	0.3244	1.5705	9.283E-6
s1	0.2403	0.3015	31	0.80	0.4315	-0.3746	0.8552	0.000014
s2	1.0292	0.2988	31	3.44	0.0017	0.4198	1.6386	-3.16E-6

The "Parameter Estimates" table indicates significance of all the parameters except S1 (Output 88.2.4).

Output 88.2.5 Additional Estimates

Additional Estimates

Label	Estimate	Standard Error	DF	t Value	Pr > |t|	Alpha	Lower	Upper
gamma2	0.6603	0.2165	31	3.05	0.0047	0.05	0.2186	1.1019

The "Additional Estimates" table displays results from the ESTIMATE statement (Output 88.2.5). The

estimate of γ_2 equals 0.66, agreeing with that obtained by McCulloch (1994). The standard error 0.22 is computed using the delta method (Billingsley 1986; Cox 1998).

Not shown is the P data set, which contains the original 32 observations and predictions of the p_{ij}.

Example 88.3: Probit-Normal Model with Ordinal Data

The data for this example are from Ezzet and Whitehead (1991), who describe a crossover experiment on two groups of patients using two different inhaler devices (A and B). Patients from group 1 used device A for one week and then device B for another week. Patients from group 2 used the devices in reverse order. The data entered as a SAS data set are as follows:

```
data inhaler;
   input clarity group time freq @@;
   gt = group*time;
   sub = floor((_n_+1)/2);
   datalines;
1 0 0 59   1 0 1 59   1 0 0 35   2 0 1 35   1 0 0  3   3 0 1  3   1 0 0  2
4 0 1  2   2 0 0 11   1 0 1 11   2 0 0 27   2 0 1 27   2 0 0  2   3 0 1  2
2 0 0  1   4 0 1  1   4 0 0  1   1 0 1  1   4 0 0  1   2 0 1  1   1 1 0 63
1 1 1 63   1 1 0 13   2 1 1 13   2 1 0 40   1 1 1 40   2 1 0 15   2 1 1 15
3 1 0  7   1 1 1  7   3 1 0  2   2 1 1  2   3 1 0  1   3 1 1  1   4 1 0  2
1 1 1  2   4 1 0  1   3 1 1  1
;
```

The response measurement, clarity, is the patients' assessment on the clarity of the leaflet instructions for the devices. The clarity variable is on an ordinal scale, with 1=easy, 2=only clear after rereading, 3=not very clear, and 4=confusing. The group variable indicates the treatment group, and the time variable indicates the time of measurement. The freq variable indicates the number of patients with exactly the same responses. A variable gt is created to indicate a group-by-time interaction, and a variable sub is created to indicate patients.

As in the previous example and in Hedeker and Gibbons (1994), assume an underlying latent continuous variable, here with the form

$$y_{ij} = \beta_0 + \beta_1 g_i + \beta_2 t_j + \beta_3 g_i t_j + u_i + e_{ij}$$

where i indexes patient and j indexes the time period, g_i indicates groups, t_j indicates time, u_i is a patient-level normal random effect, and e_{ij} are iid normal errors. The βs are unknown coefficients to be estimated.

Instead of observing y_{ij}, however, you observe only whether it falls in one of the four intervals: $(-\infty, 0)$, $(0, I1)$, $(I1, I1 + I2)$, or $(I1 + I2, \infty)$, where $I1$ and $I2$ are both positive. The resulting category is the value assigned to the clarity variable.

The following code sets up and fits this ordinal probit model:

```
proc nlmixed data=inhaler corr ecorr;
   parms b0=0 b1=0 b2=0 b3=0 sd=1 i1=1 i2=1;
   bounds i1 > 0, i2 > 0;
   eta = b0 + b1*group + b2*time + b3*gt + u;
   if (clarity=1) then p = probnorm(-eta);
   else if (clarity=2) then
      p = probnorm(i1-eta) - probnorm(-eta);
```

```
   else if (clarity=3) then
      p = probnorm(i1+i2-eta) - probnorm(i1-eta);
   else p = 1 - probnorm(i1+i2-eta);
   if (p > 1e-8) then ll = log(p);
   else ll = -1e20;
   model clarity ~ general(ll);
   random u ~ normal(0,sd*sd) subject=sub;
   replicate freq;
   estimate 'thresh2' i1;
   estimate 'thresh3' i1 + i2;
   estimate 'icc' sd*sd/(1+sd*sd);
run;
```

The PROC NLMIXED statement specifies the input data set and requests correlations both for the parameter estimates (CORR option) and for the additional estimates specified with ESTIMATE statements (ECORR option).

The parameters as defined in the PARMS statement are as follows: b0 (overall intercept), b1 (group main effect), B2 (time main effect), b3 (group-by-time interaction), sd (standard deviation of the random effect), i1 (increment between first and second thresholds), and i2 (increment between second and third thresholds). The BOUNDS statement restricts i1 and i2 to be positive.

The SAS programming statements begin by defining the linear predictor eta, which is a linear combination of the b parameters and a single random effect u. The next statements define the ordinal likelihood according to the clarity variable, eta, and the increment variables. An error trap is included in case the likelihood becomes too small.

A general log-likelihood specification is used in the MODEL statement, and the RANDOM statement defines the random effect u to have standard deviation sd and subject variable sub. The REPLICATE statement indicates that data for each subject should be replicated according to the freq variable.

The ESTIMATE statements specify the second and third thresholds in terms of the increment variables (the first threshold is assumed to equal zero for model identifiability). Also computed is the intraclass correlation.

The output is as follows.

Output 88.3.1 Specifications for Ordinal Data Model

The NLMIXED Procedure

Specifications	
Data Set	WORK.INHALER
Dependent Variable	clarity
Distribution for Dependent Variable	General
Random Effects	u
Distribution for Random Effects	Normal
Subject Variable	sub
Replicate Variable	freq
Optimization Technique	Dual Quasi-Newton
Integration Method	Adaptive Gaussian Quadrature

The "Specifications" table echoes some primary information specified for this nonlinear mixed model (Output 88.3.1). Because the log-likelihood function was expressed with SAS programming statements, the

distribution is displayed as *General* in the "Specifications" table.

The "Dimensions" table reveals a total of 286 subjects, which is the sum of the values of the FREQ variable for the second time point. Five quadrature points are selected for log-likelihood evaluation (Output 88.3.2).

Output 88.3.2 Dimensions Table for Ordinal Data Model

Dimensions	
Observations Used	38
Observations Not Used	0
Total Observations	38
Subjects	286
Max Obs per Subject	2
Parameters	7
Quadrature Points	5

Output 88.3.3 Parameter Starting Values and Negative Log Likelihood

Initial Parameters							
b0	b1	b2	b3	sd	i1	i2	Negative Log Likelihood
0	0	0	0	1	1	1	538.484276

The "Parameters" table lists the simple starting values for this problem (Output 88.3.3). The "Iteration History" table indicates successful convergence in 13 iterations (Output 88.3.4).

Output 88.3.4 Iteration History

Iteration History					
Iteration	Calls	Negative Log Likelihood	Difference	Maximum Gradient	Slope
1	4	476.3825	62.10176	43.7506	-1431.40
2	7	463.2282	13.15431	14.2465	-106.753
3	9	458.5281	4.70008	48.3132	-33.0389
4	11	450.9757	7.552383	22.6010	-40.9954
5	14	448.0127	2.963033	14.8688	-16.7453
6	17	447.2452	0.767549	7.77419	-2.26743
7	19	446.7277	0.517483	3.79353	-1.59278
8	22	446.5183	0.209396	0.86864	-0.37801
9	26	446.5145	0.003745	0.32857	-0.02356
10	29	446.5133	0.001187	0.056778	-0.00183
11	32	446.5133	0.000027	0.010785	-0.00004
12	35	446.5133	3.956E-6	0.004922	-5.41E-6
13	38	446.5133	1.989E-7	0.000470	-4E-7

NOTE: GCONV convergence criterion satisfied.

Output 88.3.5 Fit Statistics for Ordinal Data Model

Fit Statistics	
-2 Log Likelihood	893.0
AIC (smaller is better)	907.0
AICC (smaller is better)	910.8
BIC (smaller is better)	932.6

The "Fit Statistics" table lists the log likelihood and information criteria for model comparisons (Output 88.3.5).

Output 88.3.6 Parameter Estimates at Convergence

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|	95% Confidence Limits		Gradient
b0	-0.6364	0.1342	285	-4.74	<.0001	-0.9006	-0.3722	0.000470
b1	0.6007	0.1770	285	3.39	0.0008	0.2523	0.9491	0.000265
b2	0.6015	0.1582	285	3.80	0.0002	0.2900	0.9129	0.000080
b3	-1.4817	0.2385	285	-6.21	<.0001	-1.9512	-1.0122	0.000102
sd	0.6599	0.1312	285	5.03	<.0001	0.4017	0.9181	-0.00009
i1	1.7450	0.1474	285	11.84	<.0001	1.4548	2.0352	0.000202
i2	0.5985	0.1427	285	4.19	<.0001	0.3176	0.8794	0.000087

The "Parameter Estimates" table indicates significance of all the parameters (Output 88.3.6).

Output 88.3.7 Threshold and Intraclass Correlation Estimates

Additional Estimates

Label	Estimate	Standard Error	DF	t Value	Pr > |t|	Alpha	Lower	Upper
thresh2	1.7450	0.1474	285	11.84	<.0001	0.05	1.4548	2.0352
thresh3	2.3435	0.2073	285	11.31	<.0001	0.05	1.9355	2.7516
icc	0.3034	0.08402	285	3.61	0.0004	0.05	0.1380	0.4687

The "Additional Estimates" table displays results from the ESTIMATE statements (Output 88.3.7).

Example 88.4: Poisson-Normal Model with Count Data

This example uses the pump failure data of Gaver and O'Muircheartaigh (1987). The number of failures and the time of operation are recorded for 10 pumps. Each of the pumps is classified into one of two groups corresponding to either continuous or intermittent operation. The data are as follows:

```
data pump;
   input y t group;
   pump = _n_;
   logtstd = log(t) - 2.4564900;
   datalines;
 5  94.320 1
 1  15.720 2
 5  62.880 1
14 125.760 1
 3   5.240 2
19  31.440 1
 1   1.048 2
 1   1.048 2
 4   2.096 2
22  10.480 2
;
```

Each row denotes data for a single pump, and the variable logtstd contains the centered operation times.

Letting y_{ij} denote the number of failures for the *j*th pump in the *i*th group, Draper (1996) considers the following hierarchical model for these data:

$$
\begin{aligned}
y_{ij}|\lambda_{ij} &\sim \text{Poisson}(\lambda_{ij}) \\
\log \lambda_{ij} &= \alpha_i + \beta_i(\log t_{ij} - \overline{\log t}) + e_{ij} \\
e_{ij}|\sigma^2 &\sim \text{Normal}(0, \sigma^2)
\end{aligned}
$$

The model specifies different intercepts and slopes for each group, and the random effect is a mechanism for accounting for overdispersion.

The corresponding PROC NLMIXED statements are as follows:

```
proc nlmixed data=pump;
   parms logsig 0 beta1 1 beta2 1 alpha1 1 alpha2 1;
   if (group = 1) then eta = alpha1 + beta1*logtstd + e;
   else eta = alpha2 + beta2*logtstd + e;
   lambda = exp(eta);
   model y ~ poisson(lambda);
   random e ~ normal(0,exp(2*logsig)) subject=pump;
   estimate 'alpha1-alpha2' alpha1-alpha2;
   estimate 'beta1-beta2' beta1-beta2;
run;
```

The selected output is as follows.

Output 88.4.1 Dimensions Table for Poisson-Normal Model

The NLMIXED Procedure

Dimensions	
Observations Used	10
Observations Not Used	0
Total Observations	10
Subjects	10
Max Obs per Subject	1
Parameters	5
Quadrature Points	5

The "Dimensions" table indicates that data for 10 pumps are used with one observation for each (Output 88.4.1).

Output 88.4.2 Iteration History for Poisson-Normal Model

Iteration History

Iteration	Calls	Negative Log Likelihood	Difference	Maximum Gradient	Slope
1	4	30.6986932	2.162768	5.10725	-91.6020
2	9	30.0255468	0.673146	2.76174	-11.0489
3	12	29.7263250	0.299222	2.99040	-2.36048
4	16	28.7390263	0.987299	2.07443	-3.93678
5	18	28.3161933	0.422833	0.61253	-0.63084
6	21	28.0956400	0.220553	0.46216	-0.52684
7	24	28.0438024	0.051838	0.40505	-0.10018
8	27	28.0357134	0.008089	0.13506	-0.01875
9	30	28.0339250	0.001788	0.026279	-0.00514
10	33	28.0338744	0.000051	0.004020	-0.00012
11	36	28.0338727	1.681E-6	0.002864	-5.09E-6
12	39	28.0338724	3.199E-7	0.000147	-6.87E-7
13	42	28.0338724	2.532E-9	0.000017	-5.75E-9

NOTE: GCONV convergence criterion satisfied.

The "Iteration History" table indicates successful convergence in 13 iterations (Output 88.4.2).

Output 88.4.3 Fit Statistics for Poisson-Normal Model

Fit Statistics	
-2 Log Likelihood	56.1
AIC (smaller is better)	66.1
AICC (smaller is better)	81.1
BIC (smaller is better)	67.6

The "Fit Statistics" table lists the final log likelihood and associated information criteria (Output 88.4.3).

Output 88.4.4 Parameter Estimates and Additional Estimates

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|	95% Confidence Limits		Gradient
logsig	-0.3161	0.3213	9	-0.98	0.3508	-1.0429	0.4107	-0.00002
beta1	-0.4256	0.7473	9	-0.57	0.5829	-2.1162	1.2649	-0.00002
beta2	0.6097	0.3814	9	1.60	0.1443	-0.2530	1.4725	-1.61E-6
alpha1	2.9644	1.3826	9	2.14	0.0606	-0.1632	6.0921	-5.25E-6
alpha2	1.7992	0.5492	9	3.28	0.0096	0.5568	3.0415	-5.73E-6

Additional Estimates

Label	Estimate	Standard Error	DF	t Value	Pr > |t|	Alpha	Lower	Upper
alpha1-alpha2	1.1653	1.4855	9	0.78	0.4529	0.05	-2.1952	4.5257
beta1-beta2	-1.0354	0.8389	9	-1.23	0.2484	0.05	-2.9331	0.8623

The "Parameter Estimates" and "Additional Estimates" tables list the maximum likelihood estimates for each of the parameters and two differences (Output 88.4.4). The point estimates for the mean parameters agree fairly closely with the Bayesian posterior means reported by Draper (1996); however, the likelihood-based standard errors are roughly half the Bayesian posterior standard deviations. This is most likely due to the fact that the Bayesian standard deviations account for the uncertainty in estimating σ^2, whereas the likelihood values plug in its estimated value. This downward bias can be corrected somewhat by using the t_9 distribution shown here.

Example 88.5: Failure Time and Frailty Model

In this example an accelerated failure time model with proportional hazard is fitted with and without random effects. The data are from the "Getting Started" example of PROC LIFEREG; see Chapter 75, "The LIFEREG Procedure." Thirty-eight patients are divided into two groups of equal size, and different pain relievers are assigned to each group. The outcome reported is the time in minutes until headache relief. The variable censor indicates whether relief was observed during the course of the observation period (censor = 0) or whether the observation is censored (censor = 1). The SAS DATA step for these data is as follows:

```
data headache;
   input minutes group censor @@;
   patient = _n_;
   datalines;
11  1  0    12  1  0    19  1  0    19  1  0
19  1  0    19  1  0    21  1  0    20  1  0
21  1  0    21  1  0    20  1  0    21  1  0
20  1  0    21  1  0    25  1  0    27  1  0
30  1  0    21  1  1    24  1  1    14  2  0
16  2  0    16  2  0    21  2  0    21  2  0
23  2  0    23  2  0    23  2  0    23  2  0
25  2  1    23  2  0    24  2  0    24  2  0
26  2  1    32  2  1    30  2  1    30  2  0
32  2  1    20  2  1
;
```

In modeling survival data, censoring of observations must be taken into account carefully. In this example, only right censoring occurs. If $g(t,\boldsymbol{\beta})$, $h(t,\boldsymbol{\beta})$, and $G(t,\boldsymbol{\beta})$ denote the density of failure, the hazard function, and the survival distribution function at time t, respectively, then the log likelihood can be written as

$$\begin{aligned} l(\boldsymbol{\beta};\mathbf{t}) &= \sum_{i\in U_u} \log g(t_i,\boldsymbol{\beta}) + \sum_{i\in U_c} \log G(t_i,\boldsymbol{\beta}) \\ &= \sum_{i\in U_u} \log h(t_i,\boldsymbol{\beta}) + \sum_{i=1}^{n} \log G(t_i,\boldsymbol{\beta}) \end{aligned}$$

(See Cox and Oakes 1984, Ch. 3.) In these expressions U_u is the set of uncensored observations, U_c is the set of censored observations, and n denotes the total sample size.

The proportional hazards specification expresses the hazard in terms of a baseline hazard, multiplied by a constant. In this example the hazard is that of a Weibull model and is parameterized as $h(t,\boldsymbol{\beta}) = \gamma\alpha(\alpha t)^{\gamma-1}$ and $\alpha = \exp\{-\mathbf{x}'\boldsymbol{\beta}\}$.

The linear predictor is set equal to the intercept in the reference group (group = 2); this defines the baseline hazard. The corresponding distribution of survival past time t is $G(t,\boldsymbol{\beta}) = \exp\{-(\alpha t)^{\gamma}\}$. See Cox and Oakes (1984, Table 2.1) and the section "Supported Distributions" in Chapter 75, "The LIFEREG Procedure," for this and other survival distribution models and various parameterizations.

The following NLMIXED statements fit this accelerated failure time model and estimate the cumulative distribution function of time to headache relief:

```
proc nlmixed data=headache;
   bounds gamma > 0;
   linp  = b0 - b1*(group-2);
   alpha = exp(-linp);
   G_t   = exp(-(alpha*minutes)**gamma);
   g     = gamma*alpha*((alpha*minutes)**(gamma-1))*G_t;
   ll    = (censor=0)*log(g) + (censor=1)*log(G_t);
   model minutes ~ general(ll);
   predict 1-G_t out=cdf;
run;
```

Output 88.5.1 Specifications Table for Fixed-Effects Failure Time Model

The NLMIXED Procedure

Specifications	
Data Set	WORK.HEADACHE
Dependent Variable	minutes
Distribution for Dependent Variable	General
Optimization Technique	Dual Quasi-Newton
Integration Method	None

The "Specifications" table shows that no integration is required, since the model does not contain random effects (Output 88.5.1).

Output 88.5.2 Negative Log Likelihood with Default Starting Values

Initial Parameters			
gamma	b0	b1	Negative Log Likelihood
1	1	1	263.990327

No starting values were given for the three parameters. The NLMIXED procedure assigns the default value of 1.0 in this case. The negative log likelihood based on these starting values is shown in Output 88.5.2.

Output 88.5.3 Iteration History for Fixed-Effects Failure Time Model

Iteration History					
Iteration	Calls	Negative Log Likelihood	Difference	Maximum Gradient	Slope
1	4	169.2443	94.74602	22.5599	-2230.83
2	8	142.8735	26.3708	14.8863	-3.64643
3	11	140.6337	2.239814	11.2523	-9.49454
4	15	122.8907	17.74304	19.4496	-2.50807
5	17	121.3970	1.493699	13.8558	-4.55427
6	20	120.6238	0.773116	13.6706	-1.38064
7	22	119.2782	1.345647	15.7801	-1.69072
8	26	116.2713	3.006871	26.9403	-3.25290
9	30	109.4274	6.843925	19.8838	-6.92890
10	35	103.2981	6.129298	12.1565	-4.96054
11	40	101.6862	1.611863	14.2487	-4.34059
12	42	100.0279	1.658364	11.6985	-13.2049
13	46	99.9189048	0.108971	3.60255	-0.55176
14	49	99.8738836	0.045021	0.17071	-0.16645
15	52	99.8736392	0.000244	0.050822	-0.00041
16	55	99.8736351	4.071E-6	0.000705	-6.9E-6
17	58	99.8736351	6.1E-10	4.768E-6	-1.23E-9

NOTE: GCONV convergence criterion satisfied.

The "Iteration History" table shows that the procedure converges after 17 iterations and 34 evaluations of the objective function (Output 88.5.3).

Output 88.5.4 Fit Statistics and Parameter Estimates

Fit Statistics	
-2 Log Likelihood	199.7
AIC (smaller is better)	205.7
AICC (smaller is better)	206.5
BIC (smaller is better)	210.7

Output 88.5.4 *continued*

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|	95% Confidence Limits		Gradient
gamma	4.7128	0.6742	38	6.99	<.0001	3.3479	6.0777	5.327E-8
b0	3.3091	0.05885	38	56.23	<.0001	3.1900	3.4283	-4.77E-6
b1	-0.1933	0.07856	38	-2.46	0.0185	-0.3523	-0.03426	-1.22E-6

The parameter estimates and their standard errors shown in Output 88.5.4 are identical to those obtained with the LIFEREG procedure and the following statements:

```
proc lifereg data=headache;
   class group;
   model minutes*censor(1) = group / dist=weibull;
   output out=new cdf=prob;
run;
```

The t statistic and confidence limits are based on 38 degrees of freedom. The LIFEREG procedure computes z intervals for the parameter estimates.

For the two groups you obtain

$$\widehat{\alpha}(\text{group} = 1) = \exp\{-3.3091 + 0.1933\} = 0.04434$$
$$\widehat{\alpha}(\text{group} = 2) = \exp\{-3.3091\} = 0.03655$$

The probabilities of headache relief by t minutes are estimated as

$$1 - G(t, \text{group} = 1) = 1 - \exp\{-(0.04434 * t)^{4.7128}\}$$
$$1 - G(t, \text{group} = 2) = 1 - \exp\{-(0.03655 * t)^{4.7128}\}$$

These probabilities, calculated at the observed times, are shown for the two groups in Output 88.5.5 and printed with the following statements:

```
proc print data=cdf;
   var group censor patient minutes pred;
run;
```

Since the slope estimate is negative with p-value of 0.0185, you can infer that pain reliever 1 leads to overall significantly faster relief, but the estimated probabilities give no information about patient-to-patient variation within and between groups. For example, while pain reliever 1 provides faster relief overall, some patients in group 2 might respond more quickly than some patients in group 1. A frailty model enables you to accommodate and estimate patient-to-patient variation in health status by introducing random effects into a subject's hazard function.

Output 88.5.5 Estimated Cumulative Distribution Function

Obs	group	censor	patient	minutes	Pred
1	1	0	1	11	0.03336
2	1	0	2	12	0.04985
3	1	0	3	19	0.35975
4	1	0	4	19	0.35975
5	1	0	5	19	0.35975
6	1	0	6	19	0.35975
7	1	0	7	21	0.51063
8	1	0	8	20	0.43325
9	1	0	9	21	0.51063
10	1	0	10	21	0.51063
11	1	0	11	20	0.43325
12	1	0	12	21	0.51063
13	1	0	13	20	0.43325
14	1	0	14	21	0.51063
15	1	0	15	25	0.80315
16	1	0	16	27	0.90328
17	1	0	17	30	0.97846
18	1	1	18	21	0.51063
19	1	1	19	24	0.73838
20	2	0	20	14	0.04163
21	2	0	21	16	0.07667
22	2	0	22	16	0.07667
23	2	0	23	21	0.24976
24	2	0	24	21	0.24976
25	2	0	25	23	0.35674
26	2	0	26	23	0.35674
27	2	0	27	23	0.35674
28	2	0	28	23	0.35674
29	2	1	29	25	0.47982
30	2	0	30	23	0.35674
31	2	0	31	24	0.41678
32	2	0	32	24	0.41678
33	2	1	33	26	0.54446
34	2	1	34	32	0.87656
35	2	1	35	30	0.78633
36	2	0	36	30	0.78633
37	2	1	37	32	0.87656
38	2	1	38	20	0.20414

The following statements model the hazard for patient i in terms of $\alpha_i = \exp\{-\mathbf{x}_i'\boldsymbol{\beta} - z_i\}$, where z_i is a (normal) random patient effect. Notice that the only difference from the previous NLMIXED statements are the RANDOM statement and the addition of z in the linear predictor. The empirical Bayes estimates of the random effect (RANDOM statement), the parameter estimates (ODS OUTPUT statement), and the estimated cumulative distribution function (PREDICT statement) are saved to subsequently graph the patient-specific distribution functions.

```
ods output ParameterEstimates=est;
proc nlmixed data=headache;
   bounds gamma > 0;
   linp  = b0 - b1*(group-2) + z;
   alpha = exp(-linp);
   G_t   = exp(-(alpha*minutes)**gamma);
   g     = gamma*alpha*((alpha*minutes)**(gamma-1))*G_t;
   ll = (censor=0)*log(g) + (censor=1)*log(G_t);
   model minutes ~ general(ll);
   random z ~ normal(0,exp(2*logsig)) subject=patient out=EB;
   predict 1-G_t out=cdf;
run;
```

Output 88.5.6 Specifications for Random Frailty Model

The NLMIXED Procedure

Specifications	
Data Set	WORK.HEADACHE
Dependent Variable	minutes
Distribution for Dependent Variable	General
Random Effects	z
Distribution for Random Effects	Normal
Subject Variable	patient
Optimization Technique	Dual Quasi-Newton
Integration Method	Adaptive Gaussian Quadrature

The "Specifications" table shows that the objective function is computed by adaptive Gaussian quadrature because of the presence of random effects (compare Output 88.5.6 and Output 88.5.1). The "Dimensions" table reports that nine quadrature points are being used to integrate over the random effects (Output 88.5.7).

Output 88.5.7 Dimensions Table for Random Frailty Model

Dimensions	
Observations Used	38
Observations Not Used	0
Total Observations	38
Subjects	38
Max Obs per Subject	1
Parameters	4
Quadrature Points	9

Output 88.5.8 Iteration History for Random Frailty Model

Iteration History

Iteration	Calls	Negative Log Likelihood	Difference	Maximum Gradient	Slope
1	9	142.1214	28.82225	12.1448	-88.8664
2	12	136.4404	5.681042	25.9310	-65.7217
3	16	122.9720	13.46833	46.5655	-146.887
4	19	120.9048	2.067216	23.7794	-94.2862
5	23	109.2241	11.68068	57.6549	-92.4075
6	26	105.0647	4.159411	4.82465	-19.5879
7	28	101.9022	3.162526	14.1287	-6.33767
8	31	99.6907395	2.211468	7.67682	-3.42364
9	34	99.3654033	0.325336	5.68920	-0.93978
10	37	99.2602178	0.105185	0.31764	-0.23408
11	40	99.2544340	0.005784	1.17351	-0.00556
12	42	99.2456973	0.008737	0.24741	-0.00871
13	45	99.2445445	0.001153	0.10494	-0.00218
14	48	99.2444958	0.000049	0.005646	-0.00010
15	51	99.2444957	9.147E-8	0.000271	-1.84E-7

NOTE: GCONV convergence criterion satisfied.

The procedure converges after 15 iterations (Output 88.5.8). The achieved –2 log likelihood is only 1.2 less than that in the model without random effects (compare Output 88.5.9 and Output 88.5.4). Compared to a chi-square distribution with one degree of freedom, the addition of the random effect appears not to improve the model significantly. You must exercise care, however, in interpreting likelihood ratio tests when the value under the null hypothesis falls on the boundary of the parameter space (see, for example, Self and Liang 1987).

Output 88.5.9 Fit Statistics for Random Frailty Model

Fit Statistics	
-2 Log Likelihood	198.5
AIC (smaller is better)	206.5
AICC (smaller is better)	207.7
BIC (smaller is better)	213.0

Output 88.5.10 Parameter Estimates

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|	95% Confidence Limits		Gradient
gamma	6.2867	2.1334	37	2.95	0.0055	1.9640	10.6093	-1.89E-7
b0	3.2786	0.06576	37	49.86	<.0001	3.1453	3.4118	0.000271
b1	-0.1761	0.08264	37	-2.13	0.0398	-0.3436	-0.00867	0.000111
logsig	-1.9027	0.5273	37	-3.61	0.0009	-2.9710	-0.8343	0.000027

The estimate of the Weibull parameter has changed drastically from the model without random effects (compare Output 88.5.10 and Output 88.5.4). The variance of the patient random effect is $\exp\{-2\times 1.9027\} = 0.02225$. The listing in Output 88.5.11 shows the empirical Bayes estimates of the random effects. These are the adjustments made to the linear predictor in order to obtain a patient's survival distribution. The listing is produced with the following statements:

```
proc print data=eb;
   var Patient Effect Estimate StdErrPred;
run;
```

Output 88.5.11 Empirical Bayes Estimates of Random Effects

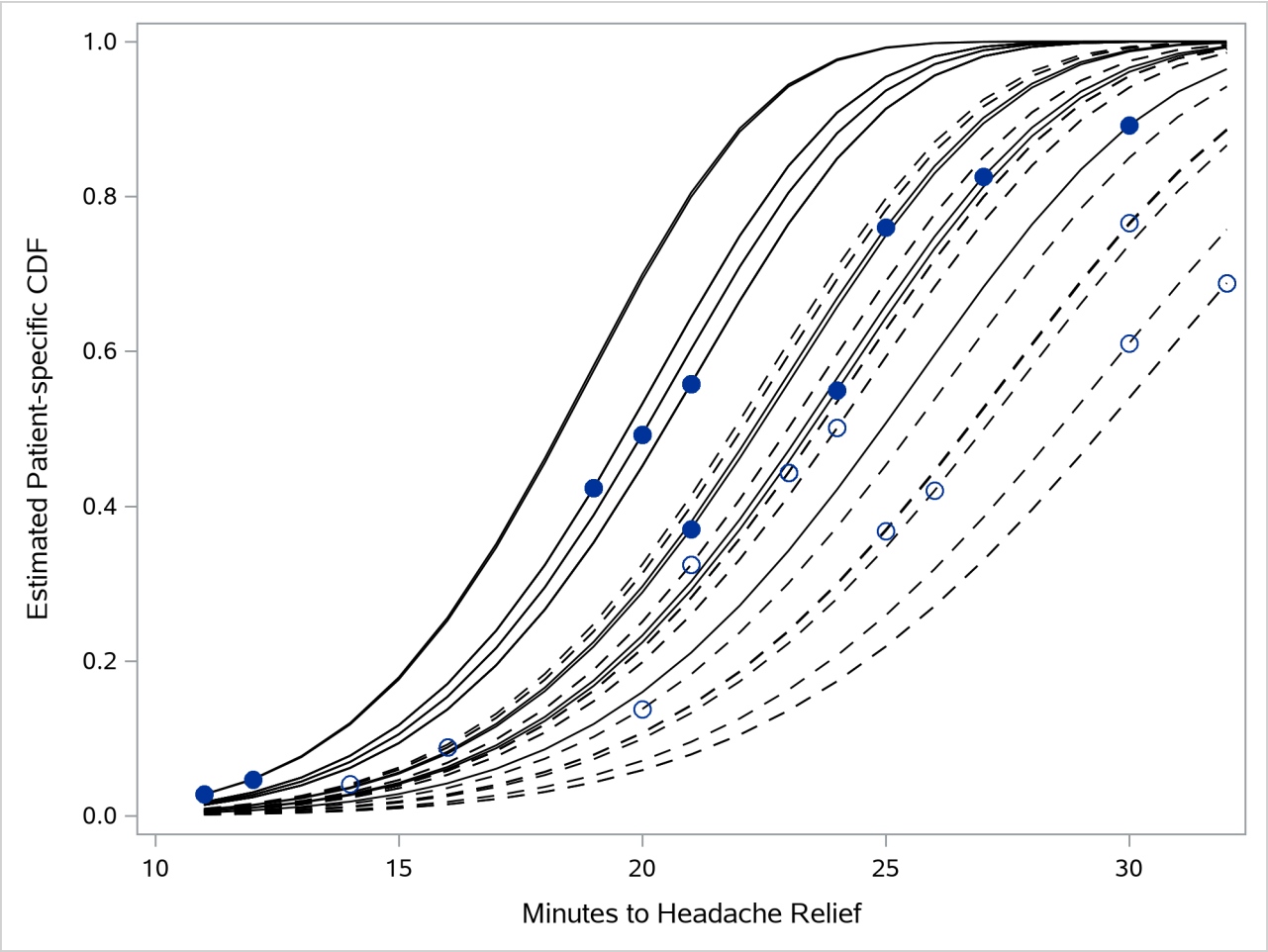
Obs	patient	Effect	Estimate	StdErrPred
1	1	z	-0.13597	0.23249
2	2	z	-0.13323	0.22793
3	3	z	-0.06294	0.13813
4	4	z	-0.06294	0.13813
5	5	z	-0.06294	0.13813
6	6	z	-0.06294	0.13813
7	7	z	-0.02568	0.11759
8	8	z	-0.04499	0.12618
9	9	z	-0.02568	0.11759
10	10	z	-0.02568	0.11759
11	11	z	-0.04499	0.12618
12	12	z	-0.02568	0.11759
13	13	z	-0.04499	0.12618
14	14	z	-0.02568	0.11759
15	15	z	0.05980	0.11618
16	16	z	0.10458	0.12684
17	17	z	0.17147	0.14550
18	18	z	0.06471	0.13807
19	19	z	0.11157	0.14604
20	20	z	-0.13406	0.22899
21	21	z	-0.12698	0.21666
22	22	z	-0.12698	0.21666
23	23	z	-0.08506	0.15701
24	24	z	-0.08506	0.15701
25	25	z	-0.05797	0.13294
26	26	z	-0.05797	0.13294
27	27	z	-0.05797	0.13294
28	28	z	-0.05797	0.13294
29	29	z	0.06420	0.13956
30	30	z	-0.05797	0.13294
31	31	z	-0.04266	0.12390
32	32	z	-0.04266	0.12390
33	33	z	0.07618	0.14132
34	34	z	0.16292	0.16459
35	35	z	0.13193	0.15528
36	36	z	0.06327	0.12124
37	37	z	0.16292	0.16459
38	38	z	0.02074	0.14160

The predicted values and patient-specific survival distributions can be plotted with the SAS code that follows:

```
proc transpose data=est(keep=estimate)
    out=trest(rename=(col1=gamma col2=b0 col3=b1));
run;

data pred;
   merge eb(keep=estimate) headache(keep=patient group);
   array pp{2} pred1-pred2;
   if _n_ = 1 then set trest(keep=gamma b0 b1);
   do time=11 to 32;
      linp      = b0 - b1*(group-2) + estimate;
      pp{group} = 1-exp(- (exp(-linp)*time)**gamma);
      symbolid  = patient+1;
      output;
   end;
   keep pred1 pred2 time patient;
run;

data pred;
   merge pred
         cdf(where  = (group=1)
             rename = (pred=pcdf1 minutes=minutes1)
             keep   = pred minutes group)
         cdf(where  = (group=2)
             rename = (pred=pcdf2 minutes=minutes2)
             keep   = pred minutes group);
   drop group;
run;

proc sgplot data=pred noautolegend;
   label minutes1='Minutes to Headache Relief'
         pcdf1   ='Estimated Patient-specific CDF';
   series  x=time     y=pred1  /
           group=patient
           lineattrs=(pattern=solid color=black);
   series  x=time     y=pred2  /
           group=patient
           lineattrs=(pattern=dash color=black);
   scatter x=minutes1 y=pcdf1  /
           markerattrs=(symbol=CircleFilled size=9);
   scatter x=minutes2 y=pcdf2  /
           markerattrs=(symbol=Circle       size=9);
run;
```

The separation of the distribution functions by groups is evident in Output 88.5.12. Most of the distributions of patients in the first group are to the left of the distributions in the second group. The separation is not complete, however. Several patients who are assigned the second pain reliever experience headache relief more quickly than patients assigned to the first group.

Output 88.5.12 Patient-Specific CDFs and Predicted Values. Pain Reliever 1: Solid Lines, Closed Circles; Pain Reliever 2: Dashed Lines, Open Circles.



Example 88.6: Simulated Nested Linear Random-Effects Model

This simulation study example demonstrates how to fit a hierarchical model with PROC NLMIXED by using a simple two-level nested linear model. In this example, the data are generated from a simple two-level nested hierarchical normal model (Littell et al. 2006). Here the conditional distribution of the response variable, given the random effects, is normal. The mean of this conditional normal distribution is a linear function of a fixed parameter and the random effects. In this simulation, only the intercept is used as a fixed parameter. Also, the first-level random effects are assumed to follow a normal distribution. Similarly, the second-level random effects that are nested within the first level follow a normal distribution. The model can be represented as follows:

$$
\begin{aligned}
y_{ij}|\beta_0, v_i, v_{j(i)} &\sim N(\mu_{ij}, \sigma^2) \\
\mu_{ij} &= \beta_0 + v_i + v_{j(i)} \\
v_i &\sim N(0, \sigma_a^2) \\
v_{j(i)} &\sim N(0, \sigma_b^2)
\end{aligned}
$$

The simulation code is as follows:

```
%let na = 100;
%let nb = 5;
%let nr = 2;
data nested;
   do A = 1 to &na;
      err1 = 3*rannor(339205);
      do B = 1 to &nb;
         err2 = 2*rannor(0);
         do rep = 1 to &nr;
            err3 = 1*rannor(0);
            resp = 10 + err1 + err2 + err3;
            output;
         end;
      end;
   end;
run;
```

You can use PROC NLMIXED to fit the preceding nested model as follows by using two RANDOM statements:

```
proc nlmixed data = nested;
   bounds vara >=0, varb_a >=0;
   mean = intercept + aeffect + beffect;
   model resp ~ normal (mean, s2);
   random aeffect ~ normal(0,vara) subject = A;
   random beffect ~ normal(0,varb_a) subject = B(A);
run;
```

The MODEL statement specifies the response variable distribution as normal. The linear relationship between the mean of this normal distribution and the fixed and random effects is specified by using a SAS programming statement. The BOUNDS statement specifies the nonnegative constraints on the variance parameters.

PROC NLMIXED with multiple RANDOM statements is used to fit hierarchical (nested) models. In this example, two RANDOM statements are used to fit a two-level nested model. The first RANDOM statement identifies aeffect as the first-level random effect, v_i, and it is followed by the subject variable A. The subject variable A indexes the number of distinct and independent subjects at the first level. Then the second RANDOM statement identifies the beffect, $v_{j(i)}$, as the second-level random effect, which is nested within the first level. The subject variable specification in the second RANDOM statement defines the nested structure. In this example, the subject variable B is nested within A.

The results from the analysis follow.

Output 88.6.1 Model Specifications for a Two-Level Nested Model

The NLMIXED Procedure

Specifications	
Data Set	WORK.NESTED
Dependent Variable	resp
Distribution for Dependent Variable	Normal
Random Effects	aeffect, beffect
Distribution for Random Effects	Normal, Normal
Subject Variables	A, B
Optimization Technique	Dual Quasi-Newton
Integration Method	Adaptive Gaussian Quadrature

The "Specifications" table (Output 88.6.1) lists the setup of the nested model. It lists all the random effects and their distribution along with the SUBJECT= variable in the nested sequence. Random effects that are listed in the specifications table are separated by a comma, indicating that aeffect is the first-level random effect, followed by the second-level random effect, beffect, which is nested within the first level. The same scheme applies to the distribution and subject items in the table. In this example, aeffect is the random effect for the first level and is specified using the subject variable A, which follows a normal distribution. Then beffect is the random effect for the second level, which is nested within the first level; it is specified using the subject variable B, which also follows a normal distribution.

The "Dimensions" table (Output 88.6.2) follows the same nested flow to exhibit the model dimensions. It indicates that 1,000 observations are used. There are 100 first-level subjects and 500 second-level subjects, because each first-level subject is contained by five nested second-level subjects. For this model, PROC NLMIXED selects one quadrature point to fit the model.

Output 88.6.2 Dimensions Table for a Two-Level Nested Model

Dimensions	
Observations Used	1000
Observations Not Used	0
Total Observations	1000
Subjects [A]	100
Max Obs per Subject	10
Subjects [B]	500
Max Obs per Subject	2
Parameters	4
Quadrature Points	1

A total of 15 steps are required to achieve convergence. They are shown in the "Iteration History" table (Output 88.6.3).

Output 88.6.3 Iteration History

Iteration History

Iteration	Calls	Negative Log Likelihood	Difference	Maximum Gradient	Slope
1	12	2982.0141	2695.598	215.363	-87641.4
2	37	2462.0898	519.9243	518.335	-346.648
3	43	2144.0047	318.0851	60.9954	-2483.38
4	49	2108.7720	35.23267	56.3198	-64.4603
5	61	2098.5627	10.20932	43.0731	-7.79282
6	67	2094.2091	4.353637	34.1533	-20.0832
7	73	2090.2805	3.928567	26.4428	-5.86543
8	80	2089.3442	0.936244	15.0337	-2.41804
9	86	2088.1104	1.233804	20.3856	-0.59116
10	98	2072.7735	15.33697	5.02801	-1.53285
11	105	2068.5554	4.218095	28.6274	-2.26437
12	112	2066.4456	2.109788	8.99816	-2.53461
13	119	2065.8119	0.633655	3.76693	-0.50862
14	126	2065.7590	0.052885	1.02710	-0.07170
15	133	2065.7550	0.004076	0.36106	-0.00577
16	139	2065.7502	0.004803	0.26122	-0.00171

NOTE: GCONV convergence criterion satisfied.

The "Parameter Estimates" table (Output 88.6.4) contains the maximum likelihood estimates of the parameters. You can see from this table that the intercept and the s2 variables are very close to the simulation parameters. Also, both variances are very close to the simulation parameters.

Output 88.6.4 Parameter Estimates for a Two-Level Nested Model

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|	95% Confidence Limits		Gradient
vara	9.2213	1.4303	498	6.45	<.0001	6.4110	12.0316	0.023371
varb_a	3.6997	0.2973	498	12.44	<.0001	3.1155	4.2839	0.008574
intercept	10.1554	0.3171	498	32.02	<.0001	9.5322	10.7785	-0.03291
s2	0.9662	0.06103	498	15.83	<.0001	0.8463	1.0861	-0.29813

Example 88.7: Overdispersion Hierarchical Nonlinear Mixed Model

This example describes the overdispersion hierarchical nonlinear mixed model that is given in Ghebretinsae et al. (2013). In this experiment, 24 rats are divided into four groups and receive a daily oral dose of 1,2-dimethylhydrazine dihydrochloride in three dose levels (low, medium, and high) and a vehicle control. In addition, an extra group of three animals receive a positive control. All 27 animals are euthanized 3 hours after the last dose is administered. For each animal, a cell suspension is prepared, and from each cell suspension, three replicate samples are prepared for scoring. Using a semi-automated scoring system, 50 randomly selected cells from each sample are scored per animal for DNA damage. The comet assay methodology is used to detect this DNA damage. In this methodology, the DNA damage is measured by the distance of DNA migration from the perimeter of the comet head to the last visible point in the tail. This distance is defined as the tail length and is observed for the 150 cells from each animal. The data are available in the Sashelp library. In order to use the dose levels as regressors in PROC NLMIXED, you need to create indicator variables for each dose level. The following DATA step creates indicator variables for the dose levels:

```
data comet;
   set sashelp.comet;
   L = 0; M = 0; H = 0; PC = 0;
   if (dose = 1.25) then L  = 1;
   if (dose = 2.50) then M  = 1;
   if (dose = 5.00) then H  = 1;
   if (dose = 200)  then PC = 1;
run;
```

Ghebretinsae et al. (2013) suggest a Weibull distribution to model the outcomes (tail lengths). Note that the data exhibit the nested nature because of the three replicate samples from each cell. Also, the 50 cells from each rat are correlated. To account for the dependency of these 150 correlated cells from each rat, two levels of normally distributed nested random effects are used. Let y_{ijk} be the tail length that is observed in the kth cell from the jth sample of the ith rat. Then the suggested two-level nested model can be described as follows:

$$\begin{aligned} y_{ijk}|r_i, s_{j(i)} &\sim \text{Weibull}(\rho, \kappa_{ijk}) \\ \kappa_{ijk} &= \exp(\beta_0 + \beta_1 \text{L}_{ijk} + \beta_2 \text{M}_{ijk} + \beta_3 \text{H}_{ijk} + \beta_4 \text{PC}_i + r_i + s_{j(i)}) \\ r_i &\sim N(0, d_1) \\ s_{j(i)} &\sim N(0, d_2) \end{aligned}$$

This model is referred to as the nested Weibull model in the rest of the example. The parameters of the Weibull distribution, ρ and κ_{ijk}, are known as the shape and scale parameters, respectively. Based on this model specification, the conditional expectation of the response variable, given the random effects, is a nonlinear function of fixed and random effects. The nonlinear relationship can be written as

$$E(y_{ijk}|r_i, s_{j(i)}) = \frac{\Gamma\left(\frac{1}{\rho}+1\right)}{\kappa_{ijk}^{1/\rho}} = \frac{\Gamma\left(\frac{1}{\rho}+1\right)}{\left(e^{(\beta_0+\beta_1 \text{L}_{ijk}+\beta_2 \text{M}_{ijk}+\beta_3 \text{H}_{ijk}+\beta_4 \text{PC}_i+r_i+s_{j(i)})}\right)^{1/\rho}}$$

These expected values can be obtained using the PREDICT statement in PROC NLMIXED, and they are useful for validating the model fitting. The specified nested Weibull model can be fitted using the following PROC NLMIXED statements:

```
proc nlmixed data = comet;
   parms b0 = -25  b1 = -10 b2 = -10 b3 = -10 b4 = -10 rho = 10;
   bounds sd1 >=0, sd2 >=0;
   mu = b0+ b1*L+ b2*M+ b3*H+ b4*PC + rateff + sampeff;
   term = (length**rho)*exp(mu);
   llike = log(rho) + (rho-1)*log(length) + mu - term;
   model length ~ general(llike);
   random ratEff ~ normal(0,sd1) subject = rat;
   random sampEff ~ normal(0,sd2) subject = sample(rat);
   predict gamma(1+(1/rho))/(exp(mu)**(1/rho)) out = p1;
run;
```

Note that the two-parameter Weibull density function is given by

$$f(y;\rho,\kappa) = \kappa\rho y^{\rho-1}e^{-\kappa y^{\rho}}$$

This implies that the log likelihood is

$$l(\rho,\kappa;y) = \log(\kappa) + \log(\rho) + (\rho-1)\log(y) - \kappa y^{\rho}$$

A general log-likelihood specification is used in the MODEL statement to specify the preceding log of the Weibull density. The linear combination of the parameters, b_0 to b_4, and the two random effects, RatEff and SampEff, is defined using the linear predictor MU in the first SAS programming statement. Then, the next two SAS programming statements together define the log of the Weibull density function. The first RANDOM statement defines the first level of the random effect, r_i, by using RatEff. Then, the second RANDOM statement defines the second level of the random effect, $s_{j(i)}$, nested within the first level by using SampEff. Both random-effects distributions are specified as normal with mean 0 and variance sd1 and sd2 for the first and second levels, respectively. The selected output from this model is shown in Output 88.7.1, Output 88.7.2, and Output 88.7.3.

Output 88.7.1 Model Specification for Nested Weibull Model

The NLMIXED Procedure

Specifications	
Data Set	WORK.COMET
Dependent Variable	Length
Distribution for Dependent Variable	General
Random Effects	rateff, sampeff
Distribution for Random Effects	Normal, Normal
Subject Variables	Rat, Sample
Optimization Technique	Dual Quasi-Newton
Integration Method	Adaptive Gaussian Quadrature

The "Specifications" table shows that RatEff is the first-level random effect, specified by the subject variable Rat, and it follows a normal distribution (Output 88.7.1). Similarly, SampEff is the second-level random effect, nested in RatEff and specified by the subject variable Sample, and it follows a normal distribution.

Output 88.7.2 Dimensions Table for Nested Weibull Model

Dimensions	
Observations Used	4050
Observations Not Used	0
Total Observations	4050
Subjects [Rat]	27
Max Obs per Subject	150
Subjects [Sample]	81
Max Obs per Subject	50
Parameters	8
Quadrature Points	1

The "Dimensions" table indicates that 4,050 observations are used in the analysis (Output 88.7.2). It also indicates that there are 27 rats and that 81 samples are nested within the rats. As explained earlier, three samples are selected randomly from each rat, so a total of 81 (27 times 3) samples are used in the analysis.

Output 88.7.3 Parameter Estimates for Nested Weibull Model

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|	95% Confidence Limits		Gradient
b0	-15.6004	0.2700	79	-57.77	<.0001	-16.1379	-15.0629	0.11014
b1	-4.5297	0.2393	79	-18.93	<.0001	-5.0060	-4.0533	-0.44060
b2	-4.6354	0.2229	79	-20.79	<.0001	-5.0792	-4.1917	-0.49383
b3	-4.8400	0.2210	79	-21.90	<.0001	-5.2800	-4.4000	0.41310
b4	-3.5203	0.2695	79	-13.06	<.0001	-4.0567	-2.9840	-0.30497
rho	4.9548	0.05901	79	83.96	<.0001	4.8373	5.0723	-0.90454
sd1	0.02598	0.04911	79	0.53	0.5982	-0.07177	0.1237	-0.35707
sd2	0.3869	0.07190	79	5.38	<.0001	0.2438	0.5300	-0.95705

The "Parameter Estimates" table list the maximum likelihood estimates of the parameters along with their standard errors (Output 88.7.3). The estimates are fairly close to the estimates that are given in Ghebretinsae et al. (2013), who use a gamma distribution for the Sample random effects instead of the normal distribution as in the preceding model. Note that the parameters β_1, β_2, and β_3 represent the effects of low, medium, and high doses, respectively, when compared to the vehicle control. Similarly, β_4 is the parameter that corresponds to the effect of positive control when compared to vehicle control. The p-value indicates that the point estimates of all these parameters, β_1 to β_4, are significant; this implies that the toxicity of 1,2-dimethylhydrazine dihydrochloride is significant at different dose levels.

In the nested Weibull model, the scale parameter, κ_{ijk}, is random because it is a function of random effects. This implies that the model assumes that each observation is drawn from a different Weibull distribution. Molenberghs et al. (2010) indicate that this way of modeling leads to an overdispersed Weibull model. Ghebretinsae et al. (2013) use the random-effects mechanism to account for this overdispersion. The

overdispersed Weibull model can be written as follows:

$$\begin{aligned} y_{ijk}|r_i, s_{j(i)}, \theta_{ijk} &\sim \text{Weibull}(\rho, \kappa_{ijk}) \\ \kappa_{ijk} &= \theta_{ijk} \exp(\beta_0 + \beta_1 \text{L}_{ijk} + \beta_2 \text{M}_{ijk} + \beta_3 \text{H}_{ijk} + \beta_4 \text{PC}_i + r_i + s_{j(i)}) \\ \theta_{ijk} &\sim \text{gamma}(\alpha, 1/\alpha) \\ r_i &\sim N(0, d_1) \\ s_{j(i)} &\sim N(0, d_2) \end{aligned}$$

This model is referred to as the nested Weibull overdispersion model in the rest of this example. Again, in this model, the shape parameter, κ_{ijk}, is the function of the normally distributed random effects, r_i and $s_{j(i)}$, along with other random effects, θ_{ijk}. Further, the model assumes that θ_{ijk} follow a gamma distribution. Note that the random effects, r_i and $s_{j(i)}$, account for the cluster effect of the observations from the same sample that is nested within a rat, whereas θ_{ijk} account for the overdispersion in the data. Molenberghs et al. (2010) integrate out the overdispersion random effects, θ_{ijk}, from the joint distribution $f(y_{ijk}|\theta_{ijk}, r_i, s_{j(i)})g(\theta_{ijk}|\alpha)$ and provide the conditional density of $y_{ijk}|r_i, s_{j(i)}$. The conditional density follows:

$$f(y_{ijk}|r_i, s_{j(i)}) = \frac{\rho y_{ij}^{\rho-1} e^{(\beta_0+\beta_1 \text{L}_{ijk}+\beta_2 \text{M}_{ijk}+\beta_3 \text{H}_{ijk}+\beta_4 \text{PC}_i+r_i+s_{j(i)})}}{\left(1 + \frac{1}{\alpha} y_{ij}^{\rho} e^{(\beta_0+\beta_1 \text{L}_{ijk}+\beta_2 \text{M}_{ijk}+\beta_3 \text{H}_{ijk}+\beta_4 \text{PC}_i+r_i+s_{j(i)})}\right)^{\alpha+1}}$$

Again, in this case, the conditional expectation of the response variable, given the random effects, is a nonlinear function of the fixed and random effects. Molenberghs et al. (2010) provide this nonlinear relationship as follows:

$$E(y_{ijk}|r_i, s_{j(i)}) = \frac{\Gamma\left(\alpha - \frac{1}{\rho}\right)\Gamma\left(\frac{1}{\rho} + 1\right)}{\left(\frac{1}{\alpha} e^{(\beta_0+\beta_1 \text{L}_{ijk}+\beta_2 \text{M}_{ijk}+\beta_3 \text{H}_{ijk}+\beta_4 \text{PC}_i+r_i+s_{j(i)})}\right)^{1/\rho} \Gamma(\alpha)}$$

As indicated previously, these values are useful for validating the fit of the model, and they can be obtained using a PREDICT statement in PROC NLMIXED. Using the preceding conditional distribution of response, given the random effects and the random-effects distribution, PROC NLMIXED fits the two-level nested Weibull overdispersion model by using the following statements:

```
ods select ParameterEstimates;
proc nlmixed data = comet;
   parms b0 = -25  b1 = -10 b2 = -10 b3 = -10 b4 = -10 alpha = 1 rho = 10;
   bounds sd1 >=0, sd2 >=0;
   mu = b0+ b1*L+ b2*M+ b3*H+ b4*PC + rateff + sampeff;
   num = rho*((length)**(rho-1))*exp(mu);
   den = (1+(((length)**rho)*exp(mu))/alpha)**(alpha+1);
   llike = log(num/den);
   model length ~ general(llike);
   random rateff ~ normal(0,sd1) subject = rat;
   random sampeff ~ normal(0,sd2) subject = sample(rat);
   predict (gamma(alpha-(1/rho))*gamma(1+(1/rho)))
           /(gamma(alpha)*((exp(mu)/alpha)**(1/rho))) out = p2;
run;
```

The parameter estimates that are obtained from the nested Weibull overdispersion model are given in Output 88.7.1. The "Parameter Estimates" table indicates that the point estimates are still significant, as they are in the previous model in which the overdispersion is not taken into consideration. So the same conclusion about the dose levels holds: the toxicity of 1,2-dimethylhydrazine dihydrochloride is significant at different dose levels.

Output 88.7.4 Parameter Estimates for Nested Weibull Overdispersion Model

The NLMIXED Procedure

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|	95% Confidence Limits		Gradient
b0	-31.0041	0.8155	79	-38.02	<.0001	-32.6272	-29.3809	-0.27078
b1	-11.9101	0.5159	79	-23.09	<.0001	-12.9370	-10.8832	-0.00704
b2	-12.1194	0.5173	79	-23.43	<.0001	-13.1492	-11.0897	0.096138
b3	-12.5516	0.5217	79	-24.06	<.0001	-13.5900	-11.5132	0.29512
b4	-9.6306	0.5554	79	-17.34	<.0001	-10.7362	-8.5250	-0.12557
alpha	0.8925	0.05048	79	17.68	<.0001	0.7920	0.9930	1.84243
rho	10.7188	0.2792	79	38.39	<.0001	10.1631	11.2745	-0.04672
sd1	0.1908	0.1548	79	1.23	0.2216	-0.1175	0.4990	0.50018
sd2	0.7926	0.1666	79	4.76	<.0001	0.4609	1.1242	-0.04066

Recall that, from both models, the conditional expected values of the response variable, given the random effects, are obtained using the PREDICT statement. These predicted values are stored in the data sets p1 and p2 for the nested Weibull and nested Weibull overdispersion models, respectively. Because, in this example, the regressor variables are only indicators, the prediction values for all the observations from the same sample that is nested within a rat are equal. Further, compute the average predicted value for each sample that is nested within a rat for both models. These values predict the average tail length of each sample that is nested within a rat based on the fitted model. In addition, compute the observed sample averages of the responses from each cluster, which can be viewed as the crude estimate of the average population tail length for each cluster. The following statements combine all these average tail lengths, both observed and predicted, from the two fitted models into a single data set:

```
proc means data = p1 mean noprint;
   class sample;
   var pred;
   output out = p11 (keep = sample weibull_mean)mean = weibull_mean;
run;

proc means data = p2 mean noprint;
   class sample;
   var pred;
   output out = p21 (keep = sample weibull_gamma_mean)mean = weibull_gamma_mean;
run;

proc means data = comet mean noprint;
   class sample;
   var length;
   id dose rat;
   output out = p31(keep = dose rat sample observed_mean) mean = observed_mean;
```

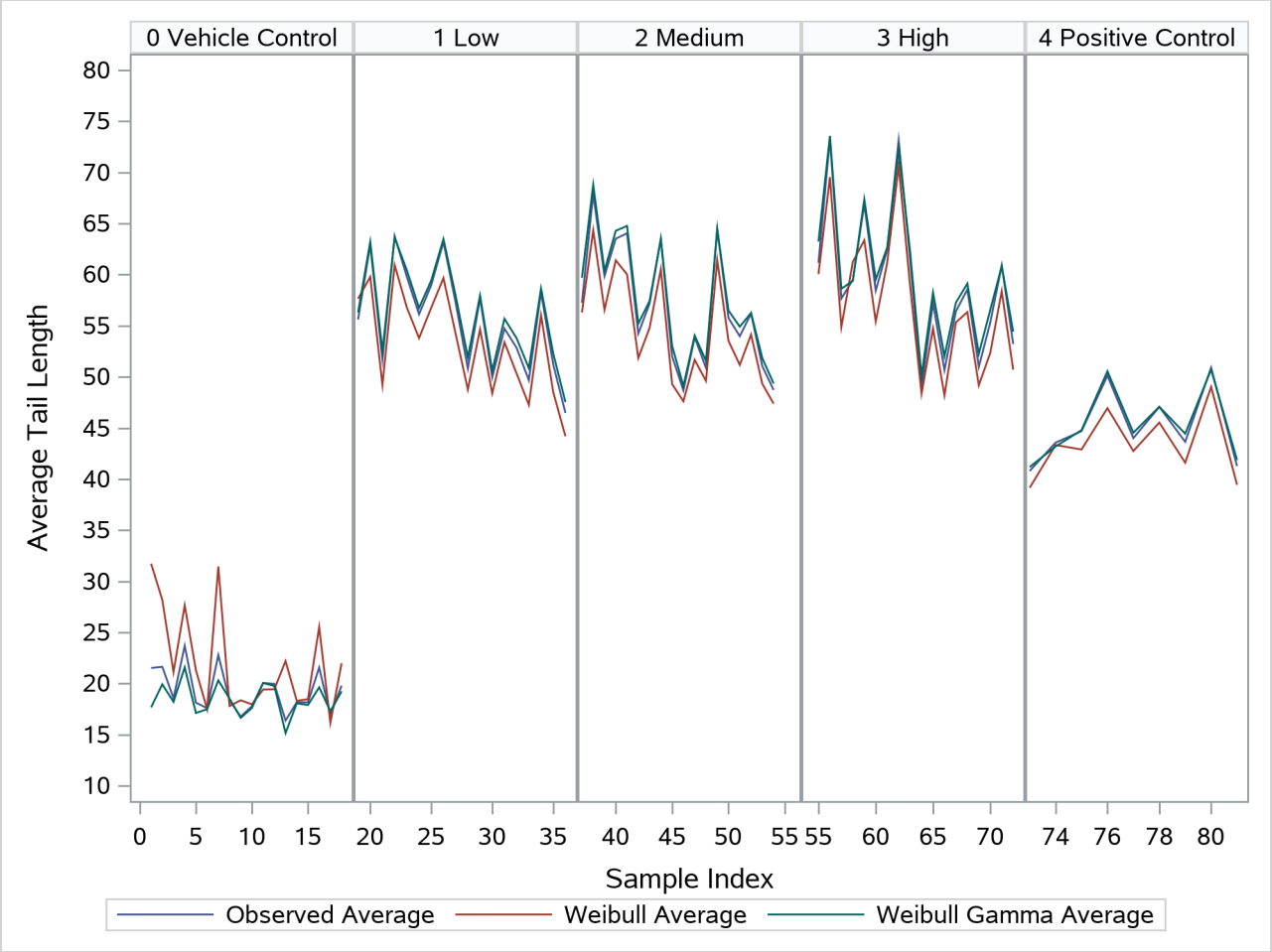
```
run;

data average;
   merge p11 p21 p31;
   by sample;
   if sample = . then delete;
   label observed_mean = "Observed Average";
   label weibull_gamma_mean = "Weibull Gamma Average";
   label weibull_mean = "Weibull Average";
   label sample = "Sample Index";
   if dose = 0 then dosage = "0 Vehicle Control ";
   if dose = 1.25 then dosage = "1 Low";
   if dose = 2.50 then dosage = "2 Medium";
   if dose = 5.00 then dosage = "3 High";
   if dose = 200  then dosage = "4 Positive Control";
run;
```

In order to validate the fit of the models, the observed average tail length values are plotted along with predicted average tail lengths from each model that is fitted using PROC NLMIXED. Moreover, the plots are created for each dosage level by using a PANELBY statement in PROC SGPANEL. The code follows:

```
proc sgpanel data = average;
   panelby dosage/onepanel layout = columnlattice novarname uniscale = row;
   rowaxis values=(10 to 80 by 5) label="Average Tail Length";
   series x = sample y = observed_mean;
   series x = sample y = weibull_mean;
   series x = sample y = weibull_gamma_mean;
run;
```

The resulting plots are given in Output 88.7.5.

Output 88.7.5 Comparison of Nested Weibull Model and Nested Weibull Overdispersion Model



You can see from the plots that the observed averages of the tail lengths from each sample that is nested within a rat are closer to the predicted averages of the nested Weibull overdispersion model than to the predicted averages of the nested Weibull model.

References

Abramowitz, M., and Stegun, I. A., eds. (1972). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* 10th printing. New York: Dover.

Abuhelwa, A. Y., Foster, D. J., and Upton, R. N. (2015). "ADVAN-Style Analytical Solutions for Common Pharmacokinetic Models." *Journal of Pharmacological and Toxicological Methods* 73:42–48.

Anderson, D. A., and Aitkin, M. (1985). "Variance Component Models with Binary Response: Interviewer Variability." *Journal of the Royal Statistical Society, Series B* 47:203–210.

Beal, S. L., and Sheiner, L. B. (1982). "Estimating Population Kinetics." *CRC Critical Reviews in Biomedical Engineering* 8:195–222.

Beal, S. L., and Sheiner, L. B. (1988). "Heteroscedastic Nonlinear Regression." *Technometrics* 30:327–338.

Beal, S. L., and Sheiner, L. B. (1992). *NONMEM User's Guide*. San Francisco: NONMEM Project Group, University of California, San Francisco.

Beal, S. L., Sheiner, L. B., Boeckmann, A. J., and Bauer, R. J., eds. (2011). *NONMEM User's Guides (1989–2011)*. Hanover, MD: Icon Development Solutions.

Beale, E. M. L. (1972). "A Derivation of Conjugate Gradients." In *Numerical Methods for Nonlinear Optimization*, edited by F. A. Lootsma, 39–43. London: Academic Press.

Beitler, P. J., and Landis, J. R. (1985). "A Mixed-Effects Model for Categorical Data." *Biometrics* 41:991–1000.

Billingsley, P. (1986). *Probability and Measure*. 2nd ed. New York: John Wiley & Sons.

Booth, J. G., and Hobert, J. P. (1998). "Standard Errors of Prediction in Generalized Linear Mixed Models." *Journal of the American Statistical Association* 93:262–272.

Breslow, N. E., and Clayton, D. G. (1993). "Approximate Inference in Generalized Linear Mixed Models." *Journal of the American Statistical Association* 88:9–25.

Burnham, K. P., and Anderson, D. R. (1998). *Model Selection and Inference: A Practical Information-Theoretic Approach*. New York: Springer-Verlag.

Cox, C. (1998). "Delta Method." In *Encyclopedia of Biostatistics*, edited by P. Armitage and T. Colton. New York: John Wiley & Sons.

Cox, D. R., and Oakes, D. (1984). *Analysis of Survival Data*. London: Chapman & Hall.

Cramer, J. S. (1986). *Econometric Applications of Maximum Likelihood Methods*. Cambridge: Cambridge University Press.

Crouch, E. A. C., and Spiegelman, D. (1990). "The Evaluation of Integrals of the Form $\int_{-\infty}^{\infty} f(t)\exp(-t^2)dt$: Application to Logistic-Normal Models." *Journal of the American Statistical Association* 85:464–469.

Davidian, M., and Gallant, A. R. (1993). "The Nonlinear Mixed Effects Model with a Smooth Random Effects Density." *Biometrika* 80:475–488.

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

Dennis, J. E., Gay, D. M., and Welsch, R. E. (1981). "An Adaptive Nonlinear Least-Squares Algorithm." *ACM Transactions on Mathematical Software* 7:348–368.

Dennis, J. E., and Mei, H. H. W. (1979). "Two New Unconstrained Optimization Algorithms Which Use Function and Gradient Values." *Journal of Optimization Theory and Applications* 28:453–482.

Dennis, J. E., and Schnabel, R. B. (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ: Prentice-Hall.

Diggle, P. J., Liang, K.-Y., and Zeger, S. L. (1994). *Analysis of Longitudinal Data*. Oxford: Clarendon Press.

Draper, D. (1996). "Discussion of the Paper by Lee and Nelder." *Journal of the Royal Statistical Society, Series B* 58:662–663.

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

Engel, B., and Keen, A. (1992). *A Simple Approach for the Analysis of Generalized Linear Mixed Models*, vol. LWA-92-6. Wageningen, Netherlands: Agricultural Mathematics Group (GLW-DLO).

Eskow, E., and Schnabel, R. B. (1991). "Algorithm 695: Software for a New Modified Cholesky Factorization." *ACM Transactions on Mathematical Software* 17:306–312.

Ezzet, F., and Whitehead, J. (1991). "A Random Effects Model for Ordinal Responses from a Crossover Trial." *Statistics in Medicine* 10:901–907.

Fisher, D., and Shafer, S. (2007). "Pharmacokinetic and Pharmacodynamic Analysis with NONMEM: Basic Concepts." Course materials for Fisher/Shafer NONMEM Workshop, March 7–11, Ghent, Belgium.

Fletcher, R. (1987). *Practical Methods of Optimization*. 2nd ed. Chichester, UK: John Wiley & Sons.

Gabrielsson, J., and Weiner, D. (2006). *Pharmacokinetic and Pharmacodynamic Data Analysis: Concepts and Applications*. 4th ed. Stockholm: Swedish Pharmaceutical Press.

Galecki, A. T. (1998). "NLMEM: New SAS/IML Macro for Hierarchical Nonlinear Models." *Computer Methods and Programs in Biomedicine* 55:207–216.

Gallant, A. R. (1987). *Nonlinear Statistical Models*. New York: John Wiley & Sons.

Gaver, D. P., and O'Muircheartaigh, I. G. (1987). "Robust Empirical Bayes Analysis of Event Rates." *Technometrics* 29:1–15.

Gay, D. M. (1983). "Subroutines for Unconstrained Minimization." *ACM Transactions on Mathematical Software* 9:503–524.

Ghebretinsae, A. H., Faes, C., Molenberghs, G., De Boeck, M., and Geys, H. (2013). "A Bayesian, Generalized Frailty Model for Comet Assays." *Journal of Biopharmaceutical Statistics* 23:618–636.

Gilmour, A. R., Anderson, R. D., and Rae, A. L. (1985). "The Analysis of Binomial Data by Generalized Linear Mixed Model." *Biometrika* 72:593–599.

Goldstein, H. (1991). "Nonlinear Multilevel Models, with an Application to Discrete Response Data." *Biometrika* 78:45–51.

Golub, G. H., and Welsch, J. H. (1969). "Calculation of Gaussian Quadrature Rules." *Mathematical Computing* 23:221–230.

Harville, D. A., and Mee, R. W. (1984). "A Mixed-Model Procedure for Analyzing Ordered Categorical Data." *Biometrics* 40:393–408.

Hedeker, D., and Gibbons, R. D. (1994). "A Random Effects Ordinal Regression Model for Multilevel Analysis." *Biometrics* 50:933–944.

Hurvich, C. M., and Tsai, C.-L. (1989). "Regression and Time Series Model Selection in Small Samples." *Biometrika* 76:297–307.

Kurada, R. R., and Chen, F. (2018). "Fitting Compartment Models Using PROC NLMIXED." In *Proceedings of the SAS Global Forum 2018 Conference*. Cary, NC: SAS Institute Inc. https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/1883-2018.pdf.

Lin, X., and Breslow, N. E. (1996). "Bias Correction in Generalized Linear Mixed Models with Multiple Components of Dispersion." *Journal of the American Statistical Association* 91:1007–1016.

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

Littell, R. C., Milliken, G. A., Stroup, W. W., Wolfinger, R. D., and Schabenberger, O. (2006). *SAS for Mixed Models*. 2nd ed. Cary, NC: SAS Institute Inc.

Liu, Q., and Pierce, D. A. (1994). "A Note on Gauss-Hermite Quadrature." *Biometrika* 81:624–629.

Longford, N. T. (1994). "Logistic Regression with Random Coefficients." *Computational Statistics and Data Analysis* 17:1–15.

McCulloch, C. E. (1994). "Maximum Likelihood Variance Components Estimation for Binary Data." *Journal of the American Statistical Association* 89:330–335.

McGilchrist, C. A. (1994). "Estimation in Generalized Mixed Models." *Journal of the Royal Statistical Society, Series B* 56:61–69.

Molenberghs, G., Verbeke, G., Demetrio, C. G. B., and Vieira, A. M. C. (2010). "A Family of Generalized Linear Models for Repeated Measures with Normal and Conjugate Random Effects." *Statistical Science* 25:325–347.

Moré, J. J. (1978). "The Levenberg-Marquardt Algorithm: Implementation and Theory." In *Lecture Notes in Mathematics*, vol. 30, edited by G. A. Watson, 105–116. Berlin: Springer-Verlag.

Moré, J. J., and Sorensen, D. C. (1983). "Computing a Trust-Region Step." *SIAM Journal on Scientific and Statistical Computing* 4:553–572.

Ochi, Y., and Prentice, R. L. (1984). "Likelihood Inference in a Correlated Probit Regression Model." *Biometrika* 71:531–543.

Owen, J. S., and Fiedler-Kelly, J. (2014). *Introduction to Population Pharmacokinetic/Pharmacodynamic Analysis with Nonlinear Mixed Effects Models*. Hoboken, NJ: John Wiley & Sons.

Pierce, D. A., and Sands, B. R. (1975). *Extra-Bernoulli Variation in Binary Data*. Technical Report 46, Department of Statistics, Oregon State University.

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

Polak, E. (1971). *Computational Methods in Optimization*. New York: Academic Press.

Powell, M. J. D. (1977). "Restart Procedures for the Conjugate Gradient Method." *Mathematical Programming* 12:241–254.

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

Rodriguez, G., and Goldman, N. (1995). "An Assessment of Estimation Procedures for Multilevel Models with Binary Response." *Journal of the Royal Statistical Society, Series A* 158:73–89.

Roe, D. J. (1997). "Comparison of Population Pharmacokinetic Modeling Methods Using Simulated Data: Results from the Population Modeling Workgroup." *Statistics in Medicine* 16:1241–1262.

Schall, R. (1991). "Estimation in Generalized Linear Models with Random Effects." *Biometrika* 78:719–727.

Schittkowski, K., and Stoer, J. (1979). "A Factorization Method for the Solution of Constrained Linear Least Squares Problems Allowing Subsequent Data Changes." *Numerische Mathematik* 31:431–463.

Self, S. G., and Liang, K.-Y. (1987). "Asymptotic Properties of Maximum Likelihood Estimators and Likelihood Ratio Tests under Nonstandard Conditions." *Journal of the American Statistical Association* 82:605–610.

Serfling, R. J. (1980). *Approximation Theorems of Mathematical Statistics*. New York: John Wiley & Sons.

Sheiner, L. B., and Beal, S. L. (1980). "Evaluation of Methods for Estimating Population Pharmacokinetic Parameters, Part I: Michaelis-Menten Model—Routine Clinical Pharmacokinetic Data." *Journal of Pharmacokinetics and Biopharmaceutics* 8:553–571.

Sheiner, L. B., and Beal, S. L. (1985). "Pharmacokinetic Parameter Estimates from Several Least Squares Procedures: Superiority of Extended Least Squares." *Journal of Pharmacokinetics and Biopharmaceutics* 13:185–201.

Smith, S. P. (1995). "Differentiation of the Cholesky Algorithm." *Journal of Computational and Graphical Statistics* 4:134–147.

Stiratelli, R., Laird, N. M., and Ware, J. H. (1984). "Random Effects Models for Serial Observations with Binary Response." *Biometrics* 40:961–971.

Vonesh, E. F. (1992). "Nonlinear Models for the Analysis of Longitudinal Data." *Statistics in Medicine* 11:1929–1954.

Vonesh, E. F. (1996). "A Note on Laplace's Approximation for Nonlinear Mixed-Effects Models." *Biometrika* 83:447–452.

Vonesh, E. F., and Chinchilli, V. M. (1997). *Linear and Nonlinear Models for the Analysis of Repeated Measurements*. New York: Marcel Dekker.

Weil, C. S. (1970). "Selection of the Valid Number of Sampling Units and Consideration of Their Combination in Toxicological Studies Involving Reproduction, Teratogenesis, or Carcinogenesis." *Food and Cosmetic Toxicology* 8:177–182.

White, H. (1982). "Maximum Likelihood Estimation of Misspecified Models." *Econometrica* 50:1–25.

Williams, D. A. (1975). "The Analysis of Binary Responses from Toxicological Experiments Involving Reproduction and Teratogenicity." *Biometrics* 31:949–952.

Wolfinger, R. D. (1993). "Laplace's Approximation for Nonlinear Mixed Models." *Biometrika* 80:791–795.

Wolfinger, R. D. (1997). "Comment: Experiences with the SAS Macro NLINMIX." *Statistics in Medicine* 16:1258–1259.

Wolfinger, R. D., and Lin, X. (1997). "Two Taylor-Series Approximation Methods for Nonlinear Mixed Models." *Computational Statistics and Data Analysis* 25:465–490.

Wolfinger, R. D., and O'Connell, M. A. (1993). "Generalized Linear Mixed Models: A Pseudo-likelihood Approach." *Journal of Statistical Computation and Simulation* 48:233–243.

Yuh, L., Beal, S. L., Davidian, M., Harrison, F., Hester, A., Kowalski, K., Vonesh, E., and Wolfinger, R. D. (1994). "Population Pharmacokinetic/Pharmacodynamic Methodology and Applications: A Bibliography." *Biometrics* 50:566–575.

Subject Index

Syntax Index