NONMEM Users Guide - Part VIII

Help Guide

May 2020

by

Alison J. Boeckmann

Lewis B. Sheiner

Stuart L. Beal

ICON plc
Gaithersburg, Maryland

Contents

INTRODUCTION

This guide is intended to provide experienced users of the NONMEM system with a fast way to locate information on a given reserved word or topic.

There are four sections:

Reserved Words
Miscellaneous Topics
Detailed Descriptions
Examples

The reserved words and miscellaneous topics appear in the left column of their respective sections, with titles of corresponding and relevant detailed descriptions and examples in the right column. The detailed descriptions provide brief summaries of the subject matter and are not intended to replace the other NONMEM guides, which should be consulted if more information is needed.

Descriptions of NM-TRAN control records have a special syntax that is described fully in Guide IV, section III.A. Please note that square brackets are used to surround an option or group of options, none of which need actually appear in the record. If they surround a group of options, a vertical line is used to separate these options in the description, and at most one of the options may be selected to actually appear in the record. If none are selected to appear, then the default option, listed first and indicated in boldface (if there is an option so indicated), is understood to apply.

Some of the examples are drawn from other NONMEM guides and sources, and some are new to this guide.

All sections are arranged in alphabetic order.

The help and html directories of the NONMEM 7.4 distribution media contain essentially the same content as this printed guide, enabling the contents of this guide to be searched and displayed on-line.

HOW TO USE THIS GUIDE

Suppose one wishes to learn more about (say) nonlinear kinetics with PREDPP. The term "nonlinear" is listed among the miscellaneous topics in the second section:

nonlinear_kinetics_model	ADVAN10
	ADVAN6 ADVAN8
	ADVAN9
	Michaelis Menten example

One may then look at the detailed descriptions entitled "ADVAN6-ADVAN8", "ADVAN9", "ADVAN10" in the third section, or at the Michaelis Menten example in the fourth section.

Some of these titles also happen to be reserved words, and are therefore also listed in the first section.

ADVAN6, ADVAN8	$DES
	ADVAN6 ADVAN8
	DES
	PRCOMG
	SS6 SS9
	Control7 example
	Michaelis Menten example
	PK_PD_simultaneous_2 example
	PK_PD_simultaneous_4 example

Again, some of these titles ($DES and DES, for example) are also listed in the first section, and titles of additional detailed descriptions and examples of interest may be found.

The detailed descriptions themselves contain references to other detailed descriptions and examples in this guide, and to other NONMEM guides in which more complete information can be found. At the end of the detailed description for $DES, for example, is:

> See also **des**, **advan6 advan8**, **advan9**.
>
> REFERENCES: Guide IV, section V.C.7
> REFERENCES: Guide VI, section VI.C

`;`	Abbreviated code
`#`	$DATA
`*`	$DATA
`@`	$DATA
`:`	II data item TIME data item
`@`	Verbatim code
`-2LL`	$ESTIMATION, $ESTM $ESTIMATION record options
`-2LOGLIKELIHOOD, -2LLIKELIHOOD`	$ESTIMATION, $ESTM
`A_0`	Initial Steady State: I_SS, ISSMOD
`A_0FLG`	$PK
`A_0FLG, A_0, A_INITIAL`	Compartment Initialization: A_0 Compartment Initialization: A_0FLG Compartment Initialization Block
`A_0(n)`	Initial Steady State: I_SS, ISSMOD $PK PK
`$ABBREVIATED`	$ABBREVIATED Abbreviated Function Examples BIVARIATE FUNCTION Infn_interpolation example 2
`ABORT`	Abbreviated code $ESTIMATION, $ESTM Interoccasion_variability example $THETA
`ABS`	Abbreviated code
`ABS0, ABSZERO`	$SCATTERPLOT
`ACCEPT`	$DATA Ignore_accept example
`ACOS`	Abbreviated code
`ADDITIONAL`	Calling protocol phrase
`ADDL`	Absorption lag parameter ADDL data item $BIND Bind example Dose event record Empirical SS dose event record $INPUT SS dose event record TDM example
`ADV10`	SS6 SS9
`ADV13,`	SS6 SS9
`ADV13-ADV18`	DES Diff Eq Solver Settings

ADV14,	SS6 SS9
ADV16,	SS6 SS9
ADV18	SS6 SS9
ADV6	DES
ADV6,	Diff Eq Solver Settings SS6 SS9
ADV6, ADV8	$DES
ADV6, ADV8, ADV13-14, ADV16, ADV18	ADVAN6 ADVAN8 ADVAN13 ADVAN14 PK_PD_simultaneous_2 example PK_PD_simultaneous_4 example
ADV8	DES Diff Eq Solver Settings SS6 SS9
ADV9	DES Diff Eq Solver Settings
ADV9, ADV13-ADV18	$DES
ADVAN1	ADVAN1 ADVAN (general) Bayes Example 3 Bayes Example 4 Bayes Example 5 Bayes Example 7r Phenobarb example
ADVAN10	ADVAN10 ADVAN (general) Michaelis_Menten example
ADVAN11	ADVAN11 ADVAN (general)
ADVAN12	ADVAN12 ADVAN (general)
ADVAN13	ADVAN (general) Bayes Example 6 Bayes Example 6s Bayes Example 6sb MXSTEP
ADVAN14	ADVAN (general) MXSTEP
ADVAN15	ADVAN9_15 ADVAN (general) $AES AES $AESINITIAL CALL data item MXSTEP
ADVAN16	ADVAN (general) MXSTEP
ADVAN17	ADVAN9_15

	ADVAN (general)
	$AES
	AES
	$AESINITIAL
	CALL data item
	MXSTEP
ADVAN18	ADVAN (general)
	MXSTEP
ADVAN2	ADVAN2
	ADVAN (general)
	Bayes Example 10
	Bayes Example 10l
	Control3 example
	Control5 example
	Interoccasion_variability example
ADVAN3	ADVAN3
	ADVAN (general)
	Bayes Example 1
	Bayes Example 1B
	Bayes Example 2
	Bayes Example 8
	Bayes Example 9
	FOCE_PARALLEL example
	Multiple_dose example
ADVAN4	ADVAN4
	ADVAN (general)
ADVAN5	ADVAN5 ADVAN7
	ADVAN (general)
	PK_PD_simultaneous_3 example
	Three_compartment example
ADVAN6	ADVAN (general)
ADVAN6, ADVAN8	Control7 example
	INTERPOLATION IN $DES
	Michaelis_Menten example
ADVAN7	ADVAN5 ADVAN7
	ADVAN7: FACTOR
	ADVAN (general)
	Control6 example
	PK_PD_simultaneous_3 example
	Three_compartment example
ADVAN8	ADVAN (general)
ADVAN9	ADVAN9_15
	ADVAN (general)
	$AES
	AES
	$AESINITIAL
	CALL data item
	MXSTEP

ADVAN, ADVAN1-ADVAN18	ADVAN (general) $SUBROUTINES
AES	ADVAN9_15 $AES AES CALL data item CALLFL Calling protocol phrase DES AES: ICALL, IDEFD, IDEFA $SUBROUTINES
$AES	$AES
AES	AES
AESINITIAL, AES0	ADVAN9_15
$AESINITIAL, $AES0	$AESINITIAL
AESINITIAL, AES0	$AESINITIAL
AGH	NONMEM output files
ALAG	Moving Area Under the Curve Example
ALAG1	Absorption lag parameter Bind example PK_PD_sequential_1 example PK_PD_sequential_2 example
ALAG1-ALAG9	$BIND Dose event record $PK
ALAGN	Dose event record $PK PK
ALPHA	Exogenous supplementation example $PK TRANS5 TRANS6
AMT	AMT data item Control5 example Dose event record Exogenous supplementation example $INPUT SS dose event record
A(n)	$AES AES $AESINITIAL $DES DES $ERROR ERROR Initial Steady State example $PK PK_PD_simultaneous_4 example

	State Vector: A State Vector Time: TSTATE
AND	$MODEL
$ANNEAL	$ANNEAL CONSTRAINT
Annnnn	$TABLE
ANRD	$TOL TOL
ANRDC	$TOL TOL
AOB	Exogenous supplementation example $PK TRANS5
APPEND	$TABLE
ASIN	Abbreviated code
ATAN	Abbreviated code
ATOL	ADVAN6 ADVAN8 ADVAN13 ADVAN14 ADVAN9_15 $COVARIANCE, $COVR $ESTIMATION, $ESTM $ESTIMATION record options $SUBROUTINES $TOL TOL
ATOLC	ADVAN6 ADVAN8 ADVAN13 ADVAN14
A_UFLG, A_U	Compartment Update Block (NM75)
AUTO	$ESTIMATION, $ESTM $ESTIMATION record options
AXIS	FINEDATA NMTEMPLATE
BACKGROUND, BACKGROUND	nmfe utility program
BAYES	Bayes Example 1 Bayes Example 10 Bayes Example 10l Bayes Example 1B Bayes Example 2 Bayes Example 3 Bayes Example 4 Bayes Example 5 Bayes Example 6 Bayes Example 6s Bayes Example 6sb Bayes Example 7r Bayes Example 8 Bayes Example 9 $ESTIMATION, $ESTM $ESTIMATION record options

BAYES, BAYESIAN	Aneal $ESTIMATION, $ESTM
BAYES_PHI_STORE	$ESTIMATION, $ESTM $ESTIMATION record options
BETA	Exogenous supplementation example $PK TRANS5 TRANS6
BFM	NONMEM output files
BIND	$BIND Dose Time Non-Event: DOSTIM $PK
$BIND	$BIND
BIND	Bind example
BIONLY	$ESTIMATION, $ESTM $ESTIMATION record options
BLANKOK	$DATA
BLOCK	Control4 example Interoccasion_variability example Mixture_model example $OMEGA $OMEGAP, $OMEGAPD $SIGMA
BOOTDATA	$ESTIMATION, $ESTM
BOOTSTRAP	$NONPARAMETRIC $SIMULATION
BY	$MODEL Phenobarb example $SCATTERPLOT $TABLE
C	Correlation Across L2 Records Simulation: SIMEPS Error Code
CALL	CALL data item CALLFL Calling protocol phrase $ERROR Expectation block Initialization-Finalization block $INPUT $PK Simulation block
CALLFL	$AESINITIAL $BIND Bind example CALL data item CALLFL Calling protocol phrase

	Control3 example
	Control5 example
	Control6 example
	Control7 example
	$ERROR
	$PK
	TDM example
CALPHA	Bayes Example 1
	Bayes Example 10
	Bayes Example 10l
	Bayes Example 1B
	Bayes Example 2
	Bayes Example 3
	Bayes Example 4
	Bayes Example 7r
	Bayes Example 9
	$ESTIMATION, $ESTM
	$ESTIMATION record options
CCONTR	CCONTR
	CCONTR example
	$CONTR
	MIX CONTR: THETA
	Stubs
	$SUBROUTINES
CDATA	CCONTR: Y, DATA, N1, N2, DIM
CDF_L, CDF_LA	NPD, NPDE, NPDE_MODE, DV_LOQ, CD
CELS	CELS
CENTERING	$ESTIMATION record options
CENTERING, NOCENTERING	$ESTIMATION, $ESTM
CHAIN	$CHAIN
	$ESTIMATION, $ESTM
$CHAIN	$CHAIN
CHECKMU	$ABBREVIATED
CHECKOUT, CHECKDATA	$DATA
CHOL	CHOL
CHOLESKY	$OMEGA
	$SIGMA
CHOLROFF	$COVARIANCE, $COVR
CINTERVAL	Bayes Example 8
	$ESTIMATION, $ESTM
	$ESTIMATION record options
CIPRED	$TABLE
CIRES	$TABLE
CITER	Bayes Example 1
	Bayes Example 10
	Bayes Example 10l
	Bayes Example 1B

	Copying block
	DES AES: ISFINL
	Displayed PRED-Defined Items
COM(n)	Abbreviated code
	COM COMACT COMSAV COMRES
	Copying block
	PRED-Defined Variables
	$TABLE
COMPACT	$ABBREVIATED
	DES
	DES AES: ICALL, IDEFD, IDEFA
COMPARTMENT, COMP	$MODEL
COMPRESS	$COVARIANCE, $COVR
COMRES	$ABBREVIATED
	Abbreviated code
	COM COMACT COMSAV COMRES
	Copying block
	PRED-Defined Variables
COMSAV	$ABBREVIATED
	COMACT, COMSAV
	COM COMACT COMSAV COMRES
	Copying block
	Displayed PRED-Defined Items
CONDITIONAL	$ESTIMATION, $ESTM
	$ESTIMATION record options
	$NONPARAMETRIC
	$SCATTERPLOT
	$TABLE
CONDITIONAL, UNCONDITIONAL	$COVARIANCE, $COVR
CONSTRAIN	$ESTIMATION record options
CONSTRAIN, CONSTRAINT	$ESTIMATION, $ESTM
CONSTRAINT	CONSTRAINT
	Aneal
	$ANNEAL
CONT	CONT data item
	$INPUT
CONTR	CCONTR example
	CCONTR: Y, DATA, N1, N2, DIM
	$CONTR
	CONTR
	CONTR: KCALL
	CONTR: Y DATA NOBS
	MIX CONTR: THETA
	Stubs
	$SUBROUTINES
$CONTR	$CONTR
CONTR	CONTR

COR	$ESTIMATION, $ESTM NONMEM output files
CORRELATION	$OMEGA $SIGMA
CORRL2	Correlation Across L2 Records NONMEM modules Simulation: SIMEPS Error Code variables in modules
COS	Abbreviated code
COV	$ESTIMATION, $ESTM NONMEM output files
COVARIANCE	$OMEGA $SIGMA
$COVARIANCE	Control3 example
COVARIANCE, COVR	$COVARIANCE, $COVR Phenobarb example
$COVARIANCE, $COVR	$COVARIANCE, $COVR Phenobarb example
COVARIANCE, COVR	Phenobarb example
CPRED	$SCATTERPLOT $TABLE
CPREDI	$SCATTERPLOT $TABLE
CPU	NONMEM output files
CRES	$SCATTERPLOT $TABLE
CRESI	$SCATTERPLOT $TABLE
CRIT	CRIT Stubs $SUBROUTINES
CSV	$DATA
CTYPE	Bayes Example 1 Bayes Example 10 Bayes Example 10l Bayes Example 1B Bayes Example 2 Bayes Example 3 Bayes Example 4 Bayes Example 5 Bayes Example 6 Bayes Example 6s Bayes Example 6sb Bayes Example 7r Bayes Example 8 Bayes Example 9 $CHAIN

	$ESTIMATION, $ESTM
	$ESTIMATION record options
CWRES	$SCATTERPLOT
	$TABLE
CWRESI	$SCATTERPLOT
	$TABLE
D1	Multiple_dose example
D1-D9	Dose event record
	Duration parameter
	$PK
DA	AES
	DES
DA_0, D2A_0	Compartment Initialization: A_0
DADT(n)	Control7 example
	$DES
	DES
DAETA, D2AETA	State Vector: A
DATA	CCONTR: Y, DATA, N1, N2, DIM
	$CONTR
	CONTR: Y DATA NOBS
	$DATA
$DATA	FINISH
DATA	FINISH
	$MIX
	MIX: DATA
	RECORDS=ID Example
	RECORDS=ID Example 2
$DATA	Control3 example
	$DATA
	RECORDS=ID Example
	RECORDS=ID Example 2
DATAMAXIMUM, DMAX	$WARNINGS
DATE, DAT1, DAT2, DAT3	DATE data item
	$INPUT
DATREC	Finalization example ($PRED and $INFN)
	INFN
	Infn_interpolation example 1
	Infn_interpolation example 2
	PRED
DDE	DDEXPAND
DDEXPAND	DDEXPAND
DDOST, D2DOST	Dose Time Non-Event: DOSTIM
DEFAULT	$DEFAULT
$DEFAULT	$DEFAULT
DEFDOSE, DEFOBS, DEFOBSERVATION	$MODEL
	Three_compartment example

DELIM	$ESTIMATION, $ESTM $ESTIMATION record options FINEDATA FORMAT NMTEMPLATE
DEN_, CDEN_	Nonparametric Density: DEN_, CDEN_ Nonparametric example
DERCONT	$ESTIMATION, $ESTM $ESTIMATION record options
DERIV1	$ABBREVIATED
DERIV2	$ABBREVIATED
DES	$ABBREVIATED ADVAN10 ADVAN6 ADVAN8 ADVAN13 ADVAN14 ADVAN9_15 Control7 example $DES DES DES AES: ICALL, IDEFD, IDEFA INTERPOLATION IN $DES Michaelis_Menten example PK_PD_simultaneous_2 example PK_PD_simultaneous_4 example SS6 SS9 $SUBROUTINES
$DES	Control7 example
DES	Control7 example
$DES	$DES
DES	DES
DES_DER	Diff Eq Solver Settings
$DESIGN	$DESIGN (NM75)
DF	Bayes Example 10 Bayes Example 10l $ESTIMATION record options
DF, DFS	$CHAIN $ESTIMATION, $ESTM
DFS	$ESTIMATION record options
DIAGONAL	$OMEGA $SIGMA
DIM	CONTR: III, DIM
DIMCNS	SIZES FSIZES prsizes
DIMFN	SIZES FSIZES prsizes
DIMNEW	SIZES FSIZES prsizes
DIMQ	SIZES FSIZES prsizes
DIMTMP	SIZES FSIZES prsizes
DIMTOK	SIZES FSIZES prsizes

DIMVRB	SIZES FSIZES prsizes
DIRECT	Bayes Example 1B $ESTIMATION, $ESTM $ESTIMATION record options
DISPLAY	$PRIOR
DN	Dose event record Duration parameter $PK PK
Dnnnnn	$TABLE
DO	Abbreviated code SIMETA Transit Compartment Examples
DOE	DOEXPAND
DOEXPAND	DOEXPAND
DOPDE	DOEXPAND
DOSE	$BIND Bind example TDM example
DOSREC	$AES AES $BIND $DES DES Dose Record: DOSREC Dose Time Non-Event: DOSTIM $PK
DOSTIM	$AES AES $BIND $DES DES Dose Record: DOSREC Dose Time Non-Event: DOSTIM $PK
DOUBLE, DP, D	SUBROUTINE $SUBROUTINES
DOWHILE	$ABBREVIATED
DO-WHILE	Abbreviated code
DOWHILE	Abbreviated code
DO-WHILE	SIMETA
DOWHILE	SIMETA
DO-WHILE	Transit Compartment Examples
DOWHILE	Transit Compartment Examples
DOWHILE, ENDDO	Data_Average block Expectation block

	Initialization-Finalization block
	PASS
	SIMETA
	Simulation block
DP	AES
	DES
DPSIZE	SIZES FSIZES prsizes
DROP	$BIND
	$DATA
	DATE data item
	$INPUT
	Simulation_1 example
DT	AES
	DES
DV	Control5 example
	DATA_SET
	DV and MDV data items
	INDIVIDUALS
	$INPUT
	OBSERVATION RECORDS
DV_ITM	CONTR: Y DATA NOBS
DV_ITM2	CCONTR: Y, DATA, N1, N2, DIM
DV_LOQ, DV_LAQ	NPD, NPDE, NPDE_MODE, DV_LOQ, CI
E	$COVARIANCE, $COVR
ECWRES	$SCATTERPLOT
	$TABLE
EFIRSTDOS	Additional_Record_Counters
EFIRSTOBS	Additional_Record_Counters
EFIRSTREC	Additional_Record_Counters
ELASTDOS	Additional_Record_Counters
ELASTOBS	Additional_Record_Counters
ELASTREC	Additional_Record_Counters
ELS	ELS
EM	Aneal
	$ESTIMATION, $ESTM
EMPTY	SIZES FSIZES prsizes
E(n)	ADVAN9_15
	$AES
	AES
ENDDOE	DOEXPAND
ENDPDE	DOEXPAND
Ennnnn	$TABLE
EONLY	Bayes Example 1
	Bayes Example 10
	Bayes Example 10l
	Bayes Example 1B

	Bayes Example 2
	Bayes Example 3
	Bayes Example 4
	Bayes Example 5
	Bayes Example 7r
	$ESTIMATION, $ESTM
	$ESTIMATION record options
EPRED	$SCATTERPLOT
	$TABLE
EPS	Simulation: ETA, EPS
EPS(n)	$ERROR
	ERROR
	Intra and inter-individual
	$PRED
	PRED
	Random models
	$SIGMA
EQN	$DATA
EQUILIBRIUM	$MODEL
ERES	$SCATTERPLOT
	$TABLE
ERR(n)	$ERROR
	Intra and inter-individual
	$PRED
	Random models
ERROR	CALL data item
	CALLFL
	Calling protocol phrase
	$ERROR
	ERROR
	PK_PD_sequential_1 example
	PK_PD_sequential_2 example
	PK_PD_simultaneous_1 example
	PK_PD_simultaneous_2 example
	PK_PD_simultaneous_3 example
	PK_PD_simultaneous_4 example
	State Vector: A
	State Vector Time: TSTATE
	$SUBROUTINES
$ERROR	Control3 example
	$ERROR
ERROR	ERROR
ERRORMAXIMUM, EMAX	$WARNINGS
ESAMPLE	$TABLE
ESTIMATION, ESTM	$CHAIN
	$ESTIMATION, $ESTM
	Phenobarb example

$ESTIMATION, $ESTM	$CHAIN
ESTIMATION, ESTM	$CHAIN
$ESTIMATION, $ESTM	$ESTIMATION, $ESTM $ESTIMATION record options Phenobarb example
ESTIMATION, ESTM	Phenobarb example
ETA	Non-active ETA list for PRED Simulation: ETA, EPS Write print
ETA1-ETA9, ET10-ET70	$TABLE
ETABARCHECK	$ESTIMATION record options ETABAR
ETABARCHECK, NOETABARCHECK	$ESTIMATION, $ESTM
ETADER	$ESTIMATION, $ESTM $ESTIMATION record options
ETA, ETAS	$TABLE
ETA(n)	$ERROR ERROR $ETAS, $PHIS Intra and inter-individual $OMEGA $PK PK $PRED PRED Random models $RCOV, $RCOVI $TABLE
ETAn	$TABLE
ETAS	$NONPARAMETRIC Nonparametric example
$ETAS	$ETAS, $PHIS
ETASAMPLES	$ESTIMATION, $ESTM
ETASTYPE	$ESTIMATION, $ESTM $ESTIMATION record options ETASXI
ETASXI	$ESTIMATION, $ESTM ETASXI NONMEM modules variables in modules
ETC	$ETAS, $PHIS
ETEXT	PRED Error Message
ETS	NONMEM output files
EVENT	Calling protocol phrase
EVERY	Calling protocol phrase
EVID	EVID data item

	$INPUT
	Interoccasion_variability example
EVTREC	Dose Time Non-Event: DOSTIM
	ERROR
	EVENT RECORD: EVTREC
	PK
EWRES	$SCATTERPLOT
	$TABLE
EXAMPLE2.CTL	Bayes Example 2
EXCLUDE	$MODEL
EXCLUDE_BY	Ignore_accept example
	$TABLE
EXIT	Abbreviated code
	$ESTIMATION, $ESTM
	Interoccasion_variability example
	$THETA
EXP	Abbreviated code
EXPAND	$NONPARAMETRIC
EXPECTATION-MAXIMIZATION	Aneal
	$ESTIMATION, $ESTM
EXTRADOSE	FINEDATA
F	CCONTR: F, G, H
	CONTR: F, G, H
	$ERROR
	ERROR
	PRED
F0, FO	Output fraction parameter
	$PK
	PK
	Plasma urine example
F1	Bind example
	TDM example
F1-F9	Dose event record
	$PK
FACTOR	ADVAN7: FACTOR
FACTORIAL	Abbreviated code
FAST	$COVARIANCE, $COVR
	$ESTIMATION, $ESTM
FASTDER	$ABBREVIATED
FDATA.CSV	$DATA
F_FLAG	Bayes Example 10
	Bayes Example 10l
	F_FLAG
	MDVRES
FGH	NONMEM output files
FILE	$CHAIN

	$COVARIANCE, $COVR
	$ESTIMATION, $ESTM
	$ESTIMATION record options
	FINEDATA
	NMTEMPLATE
	$TABLE
filename	$DATA
	$ESTIMATION, $ESTM
	INCLUDE
	$MSFI
	$NONPARAMETRIC
	$RCOV, $RCOVI
	RECORDS=ID Example
	RECORDS=ID Example 2
	$TABLE
FILES	FILES
FIN	FINISH
FINAL	Nonparametric example
	$SIMULATION
FINEDATA	DDEXPAND
	Finedata example
FIRST	Calling protocol phrase
	Superproblem_2 example
FIRSTDOS	Additional_Record_Counters
FIRSTEM	Partial Derivative Indicators
FIRSTEM, IFIRSTEM	$SIMULATION
", "FIRST, "LAST, "MAIN	Finalization example ($PRED and $INFN)
FIRST, LAST, MAIN	Finalization example ($PRED and $INFN)
", "FIRST, "LAST, "MAIN	Verbatim code
FIRST, LAST, MAIN	Verbatim code
FIRSTLASTONLY	$TABLE
FIRSTOBS	Additional_Record_Counters
FIRSTONLY, FIRSTRECORDONLY	$SCATTERPLOT
	$TABLE
FIRSTREC	Additional_Record_Counters
FIXED	INITIAL ESTIMATE (INITIAL EST)
	$OMEGA
	$OMEGAP, $OMEGAPD
	PK_PD_sequential_2 example
	$SIGMA
	$SIGMAP, $SIGMAPD
	Simulation_3 example
	$THETA
	$THETAP, $THETAPV
FIXEDETAS	$TABLE
FL	SIZES FSIZES prsizes

FL1	CONTR: F, G, H
FL2	CCONTR: F, G, H
FLIBR	SIZES FSIZES prsizes
FLU	FLU
FMT	WRITE Formats
FMTN	$FORMAT
FN	Bioavailability fraction parameter Dose event record $PK PK
FNLETA	$ESTIMATION, $ESTM $ESTIMATION record options FNLETA
Fnnnnn	$TABLE
FO, NOFO	$ESTIMATION, $ESTM
FOR	Non-active ETA list for PRED
FORMAT	$CHAIN $COVARIANCE, $COVR $ESTIMATION, $ESTM $ESTIMATION record options $ETAS, $PHIS $FORMAT FORMAT $TABLE
$FORMAT	$FORMAT
FORMAT	$FORMAT
FORWARD	Superproblem_1 example Superproblem_2 example $TABLE
FPARAFILE	$ESTIMATION, $ESTM $ESTIMATION record options
FPOSDEF	$COVARIANCE, $COVR
FROM	$SCATTERPLOT $TABLE
FSD	SIZES FSIZES prsizes
FSD1	SIZES FSIZES prsizes
FTL	DDEXPAND FINEDATA
FULL	$ABBREVIATED DES DES AES: ICALL, IDEFD, IDEFA
FUNC, FUNCA–FUNCI	Abbreviated Function
FUNC, FUNCA–FUNCZ	Abbreviated code Abbreviated Function Examples PHI Stubs

	Transit Compartment Examples
FUNCTION	$ABBREVIATED BIVARIATE FUNCTION
G	CCONTR: F, G, H CONTR: F, G, H ERROR PRED
G11, G21, G31	$TABLE
G2	CCONTR: F, G, H CONTR: F, G, H
GAMLN	Abbreviated code
GAMLN, GAMLND1, GAMLND2	GAMLN
GAMMA	Abbreviated code $PK TRANS6
GETETA	ERROR Finalization example ($PRED and $INFN) GETETA INFN PASS PK PRED
GG	PK TRANS (subroutine)
GGL1	CONTR: F, G, H
GGL2	CCONTR: F, G, H
GK1, HK1	$TABLE
GL1	CONTR: F, G, H
GL2	CCONTR: F, G, H
GRD	Bayes Example 2 $ESTIMATION, $ESTM $ESTIMATION record options NONMEM output files
GRDQ	$ESTIMATION, $ESTM $ESTIMATION record options
GRID	$ESTIMATION, $ESTM $ESTIMATION record options Stieltjes example
H	CCONTR: F, G, H CONTR: F, G, H ERROR PRED
H11, H21	$TABLE
HL1	CONTR: F, G, H
HL2	CCONTR: F, G, H
HYBRID	$ESTIMATION, $ESTM

IDEFD	DES AES: ICALL, IDEFD, IDEFA
IDFORMAT	FORMAT $TABLE
IDNO	MODEL
IERC, IERE	Estim Covar Error Codes Finalization example ($PRED and $INFN) Initialization-Finalization block
IERPRD	PRED Error Message PRED Exit Code
IERPRDU	PRED Exit Code
IERSQ	Simulation: SIMEPS Error Code
IETAOL, IEPSOL	Simulation: IETAOL IEPSOL
IF	Abbreviated code
IFIRSTEM	Partial Derivative Indicators
IFND	$PRIOR TNPRI TNPRI examples
IGNORE	$DATA ICALL Ignore_accept example PRED PRED_IGNORE_DATA BLOCK (NM75) Simulation_1 example Simulation_2 example
II	Bind example Dose event record Empirical SS dose event record Exogenous supplementation example II data item $INPUT Interoccasion_variability example SS dose event record TDM example
IIDX	Objective Function Value Individual Objective Function Value Individual exampl
IIDX, CNTID	Write print
III	CONTR: III, DIM
IKAPPA	$ESTIMATION record options
IMAX	Diff Eq Solver Settings
IMP	Bayes Example 1 Bayes Example 10 Bayes Example 10l Bayes Example 1B Bayes Example 2 Bayes Example 3 Bayes Example 4 Bayes Example 5

	Bayes Example 6s
	Bayes Example 6sb
	Bayes Example 7r
	$ESTIMATION, $ESTM
	NONMEM output files
IMPMAP	$ESTIMATION, $ESTM
	$ESTIMATION record options
INCLUDE	$DATA
	INCLUDE
	RECORDS=ID Example
	RECORDS=ID Example 2
	User-defined Reserved Function Example
$INDEX, $INDXS	$INDEX, $INDXS
INDEX, INDXS, INDEXES	ERROR
	$INDEX, $INDXS
	INFN
	PK
	PRED
INDIVIDUAL	Calling protocol phrase
INDR1	NINDR INDR1 INDR2
INDR2	NINDR INDR1 INDR2
INFILE	$DATA
	RECORDS=ID Example
	RECORDS=ID Example 2
$INFILE	$DATA
	RECORDS=ID Example
	RECORDS=ID Example 2
INF, INFINITY	$THETA
INFN	$INFN
	INFN
	PASS
	PASS NEWIND: NWIND
	Stubs
	$SUBROUTINES
	SUPP
$INFN	$INFN
INFN	Finalization example ($PRED and $INFN)
	Finedata example
	$INFN
	INFN
	Infn_interpolation example 1
	Infn_interpolation example 2
	Initialization-Finalization block
INFNP	INFN-defined Variables
INIT	AES
INITIAL	$SIMULATION
INITIALOFF, INITIALON	$MODEL

	Three_compartment example
INPUT	$BIND
	$INPUT
	$OMIT
$INPUT	$BIND
INPUT	Bind example
$INPUT	Control3 example
	DATA_ITEMS
	DATA_RECORDS
	DATA_SET
	$INPUT
INT	Abbreviated code
INTEGER	$ABBREVIATED
INTERACTION	Bayes Example 1
	Bayes Example 10
	Bayes Example 10l
	Bayes Example 1B
	Bayes Example 2
	Bayes Example 3
	Bayes Example 4
	Bayes Example 5
	Bayes Example 6
	Bayes Example 6s
	Bayes Example 6sb
	Bayes Example 7r
	Bayes Example 8
	Bayes Example 9
	$ESTIMATION, $ESTM
	$ESTIMATION record options
INTFLG	Diff Eq Solver Settings
IPH	NONMEM output files
IPRNV	Super Problem Print Control
IPROB	Problem Iteration Counters
IPS	Population single-Subject Indicator
IR	AES
	Calling protocol phrase
	DES
	MODEL
I_REC	Recursive PRED Indicator
IRECIDX	Additional_Record_Counters
	Moving Area Under the Curve Example
IREP	Simulation: NREP, IREP
IREV	ERROR
	PK
IRGG	PK
	TRANS (subroutine)

ISAMPEND	$ESTIMATION, $ESTM $ESTIMATION record options
ISAMPLE	Bayes Example 1 Bayes Example 10 Bayes Example 10l Bayes Example 1B Bayes Example 2 Bayes Example 3 Bayes Example 4 Bayes Example 5 Bayes Example 6 Bayes Example 6s Bayes Example 6sb Bayes Example 7r $CHAIN $ESTIMATION, $ESTM $ESTIMATION record options
ISAMPLE_M1	$ESTIMATION record options
ISAMPLE_M1A	$ESTIMATION record options
ISAMPLE_M1B	$ESTIMATION, $ESTM $ESTIMATION record options
ISAMPLE_M1, ISAMPLE_M1A	$ESTIMATION, $ESTM
ISAMPLE_M2	$ESTIMATION record options
ISAMPLE_M2, ISAMPLE_M3	$ESTIMATION, $ESTM
ISAMPLE_M3	$ESTIMATION record options
ISAMPLEMAX	SIZES FSIZES prsizes
ISCALE_MAX	$ESTIMATION record options
ISCALE_MIN	$ESTIMATION record options
ISCALE_MIN, ISCALE_MAX, IKAPPA	$ESTIMATION, $ESTM
ISFINL	$AES $DES DES AES: ISFINL
ISIZE	SIZES FSIZES prsizes
I_SS	ADVAN6 ADVAN8 ADVAN13 ADVAN14 ADVAN9_15
ISS	Initial Steady State example
I_SS	Initial Steady State example Initial Steady State: I_SS, ISSMOD $MODEL
ISS	NWPRI NWPRI examples
I_SS	$PK
ISS	$PRIOR PRIOR TNPRI TNPRI examples

ISSMOD	Initial Steady State: I_SS, ISSMOD
ISTFLG	Diff Eq Solver Settings
ITASK_	STOP_TIME, ITASK_ (NM75)
ITERATIONS	Problem Iteration Counters $SUPER Superproblem_1 example Superproblem_2 example
ITRANS	TRANS (subroutine)
ITS	Bayes Example 1 Bayes Example 10 Bayes Example 10l Bayes Example 1B Bayes Example 2 Bayes Example 3 Bayes Example 4 Bayes Example 5 Bayes Example 6 Bayes Example 6s Bayes Example 6sb Bayes Example 7r $ESTIMATION, $ESTM $ESTIMATION record options
ITYP	NWPRI NWPRI examples $PRIOR PRIOR TNPRI TNPRI examples
IVAR	$PRIOR TNPRI TNPRI examples
IWRB	SIZES FSIZES prsizes
IWRE	SIZES FSIZES prsizes
K	ADVAN1 ADVAN11 ADVAN12 ADVAN2 ADVAN3 ADVAN4 $PK
K12, K21	ADVAN11 ADVAN3 $PK
K13, K31	ADVAN11 $PK
K21, K31, K32, K42	TRANS6
K23, K32	ADVAN12

	ADVAN4
	$PK
K24, K42	ADVAN12
	$PK
KA	ADVAN12
	ADVAN2
	ADVAN4
	Exogenous supplementation example
	$PK
	TRANS2
	TRANS3
	TRANS4
	TRANS5
	TRANS6
KAPPA	$ESTIMATION, $ESTM
	$ESTIMATION record options
KCALL	CONTR: KCALL
KM	ADVAN10
KM0, KMN	ADVAN5 ADVAN7
	$PK
KNUTHSUMOFF	$COVARIANCE, $COVR
	$ESTIMATION, $ESTM
	$ESTIMATION record options
L1	ID .ID. and L1 data items
	$INPUT
	SIMETA
L1, L2	CCONTR: Y, DATA, N1, N2, DIM
L2	CCONTR: F, G, H
	CONTR: III, DIM
	Correlation Across L2 Records
	$INPUT
	L2 data item
	Plasma urine example
	SIMEPS
L2CR_DIM	CCONTR: Y, DATA, N1, N2, DIM
L2NO	CCONTR: Y, DATA, N1, N2, DIM
LABEL	$OMEGA
	$SIGMA
	$THETA
LADD	SIZES FSIZES prsizes
LAGGED	Calling protocol phrase
LAPLACE	$ESTIMATION record options
LAPLACIAN, LAPLACEAN	$ABBREVIATED
	Bayes Example 10
	Bayes Example 10l
	ERROR
	$ESTIMATION, $ESTM

	$ESTIMATION record options
	Logistic regression example
	PRED
LAST	$BIND
	$TABLE
LAST20	$DATA
LASTDOS	Additional_Record_Counters
LASTOBS	Additional_Record_Counters
LASTONLY	$TABLE
LASTREC	Additional_Record_Counters
LBSIZE	SIZES FSIZES prsizes
LCM110	SIZES FSIZES prsizes
LEVCENTER	$ESTIMATION, $ESTM
LEVEL	$LEVEL
$LEVEL	$LEVEL
LEVWT	$ESTIMATION, $ESTM
	$ESTIMATION record options
LFORMAT	FORMAT
	$TABLE
LICFILE, LICFILE	nmfe utility program
LIKE	$ESTIMATION record options
LIKELIHOOD	Expectation example
	Logistic regression example
	NCONTR
	PRED
	PRED_RES_WRES
LIM1	SIZES FSIZES prsizes
LIM10	SIZES FSIZES prsizes
LIM11	SIZES FSIZES prsizes
LIM13	SIZES FSIZES prsizes
LIM15	SIZES FSIZES prsizes
LIM16	SIZES FSIZES prsizes
LIM2	SIZES FSIZES prsizes
LIM3	SIZES FSIZES prsizes
LIM4	SIZES FSIZES prsizes
LIM5	SIZES FSIZES prsizes
LIM6	SIZES FSIZES prsizes
LIM7	SIZES FSIZES prsizes
LIM8	SIZES FSIZES prsizes
LINK	$MODEL
	MODEL
LIREC	Moving Area Under the Curve Example
	Size of Individual Record
LIST	Non-active ETA list for PRED

	$WARNINGS
LL, LOGLIKELIHOOD	$ESTIMATION, $ESTM Logistic regression example
LNP4	PRED-Defined Variables SIZES FSIZES prsizes
LNTWOPI	$ESTIMATION, $ESTM $ESTIMATION record options
LOCFILE	nmfe utility program
LOG, LOG10	Abbreviated code
LPAR	SIZES FSIZES prsizes
LPAR2	SIZES FSIZES prsizes
LPAR3	SIZES FSIZES prsizes
LRECL	$DATA
LS_ALLOC1	SIZES FSIZES prsizes
LS_ALLOC2	SIZES FSIZES prsizes
LS_DIM	CONTR: III, DIM
LSTEXT	SIZES FSIZES prsizes
LSUPP	$NONPARAMETRIC SIZES FSIZES prsizes
LTH	SIZES FSIZES prsizes
LVOUT	Non-active ETA list for PRED
LVR	SIZES FSIZES prsizes
LVR2	SIZES FSIZES prsizes
LVRM	SIZES FSIZES prsizes
LWS1	SIZES FSIZES prsizes
LWS2	SIZES FSIZES prsizes
LWS3	SIZES FSIZES prsizes
M	Active ETA list for PREDPP
M1	$ESTIMATION, $ESTM
M1A	$ESTIMATION, $ESTM
M1B	$ESTIMATION, $ESTM
M2	$ESTIMATION, $ESTM
M3	$ESTIMATION, $ESTM
MADAPT	$ESTIMATION, $ESTM $ESTIMATION record options
MAP	$ESTIMATION, $ESTM $ESTIMATION record options
MAPCOV	$ESTIMATION, $ESTM $ESTIMATION record options
MAPINTER	$ESTIMATION record options
MAPINTER, MAPITER, MAPITERS	$ESTIMATION, $ESTM
MAPITER	Bayes Example 1 Bayes Example 10 Bayes Example 10l

	Bayes Example 1B
	Bayes Example 2
	Bayes Example 3
	Bayes Example 4
	Bayes Example 5
	Bayes Example 7r
	$ESTIMATION record options
MAPITERS	$ESTIMATION record options
MARGINAL	$NONPARAMETRIC
	Nonparametric Density: DEN_, CDEN_
	Nonparametric example
MASSRESET	$ESTIMATION record options
MATRIX	Bayes Example 1
	Bayes Example 10
	Bayes Example 10l
	Bayes Example 1B
	Bayes Example 2
	Bayes Example 3
	Bayes Example 4
	Bayes Example 5
	Bayes Example 6
	Bayes Example 6s
	Bayes Example 6sb
	Bayes Example 7r
	Bayes Example 8
	Bayes Example 9
	$COVARIANCE, $COVR
MAX	Abbreviated code
MAXCAL	Diff Eq Solver Settings
MAXDIMM	SIZES FSIZES prsizes
MAXDIMZ	SIZES FSIZES prsizes
MAXETAI	SIZES FSIZES prsizes
MAXEVAL	$ESTIMATION record options
MAXEVALS	Control3 example
	$ESTIMATION, $ESTM
	MONITORING OF SEARCH
MAXEXTRA	SIZES FSIZES prsizes
MAX_EXTRA	SIZES FSIZES prsizes
MAXFCN	Diff Eq Solver Settings
	SIZES FSIZES prsizes
MAXFTEXT	SIZES FSIZES prsizes
MAXIC	SIZES FSIZES prsizes
MAXIDS	Objective Function Value Individual
	Objective Function Value Individual exampl
	SIZES FSIZES prsizes
MAXITER	SIZES FSIZES prsizes

MAXLIM	nmfe utility program
MAXNSP	SIZES FSIZES prsizes
MAXOMEG	SIZES FSIZES prsizes
MAXOMEGL	SIZES FSIZES prsizes
MAXOTHER	SIZES FSIZES prsizes
MAXPARW	SIZES FSIZES prsizes
MAXPHICOVI	SIZES FSIZES prsizes
MAXPTHETA	SIZES FSIZES prsizes
MAXRECID	SIZES FSIZES prsizes
MAXSIDL	SIZES FSIZES prsizes
MAXSIDS	SIZES FSIZES prsizes
MAXVRESN	SIZES FSIZES prsizes
MAXVRESWRES	SIZES FSIZES prsizes
MAXXNAME	SIZES FSIZES prsizes
MCETA	$ESTIMATION, $ESTM $ESTIMATION record options
MDV	DV and MDV data items $INPUT Multiple_dose example OBSERVATION RECORDS Phenobarb example
MDVI1	DV and MDV data items
MDVI2	DV and MDV data items
MDVI3	DV and MDV data items
MDVRES	MDVRES NONMEM modules variables in modules
METH	Diff Eq Solver Settings
METHOD	$ABBREVIATED $ESTIMATION, $ESTM
MFIRST	Partial Derivative Indicators
MIN	Abbreviated code
MISDAT	$DATA
MISSING	FINEDATA Finedata example NMTEMPLATE
MITER	Diff Eq Solver Settings
MIX	$CONTR $MIX MIX MIX CONTR: THETA MIX: DATA MIXNUM_MIXEST_MIXP Mixture_model example Stubs

	$SUBROUTINES
$MIX	$MIX
MIX	$MIX MIX MIXNUM_MIXEST_MIXP
MIXNUM, MIXEST, MIXP	MIXNUM_MIXEST_MIXP Mixture_model example Mixture model: MIXNUM, MIXEST Mixture model: MIXP
MIXP	$MIX
MIXPT	Mixture model: MIXPT
MMX	SIZES FSIZES prsizes
MNEXT	Enterhepatic circulation examples Examples Using MTIME to Model Periodic Model Event Time: MNOW, MPAST, MNE Model Time examples MTIME MNEXT MPAST MNOW MTDIF
MNOW	Enterhepatic circulation examples Examples Using MTIME to Model Periodic Model Event Time: MNOW, MPAST, MNE Model Time examples MTIME MNEXT MPAST MNOW MTDIF
MOD	Abbreviated code
MODE	Finalization example ($PRED and $INFN) INFN Infn_interpolation example 1 Infn_interpolation example 2 Initialization-Finalization block PASS $PRIOR TNPRI TNPRI examples
MODEL	ADVAN5 ADVAN7 ADVAN6 ADVAN8 ADVAN13 ADVAN14 ADVAN9_15 ADVAN (general) Control6 example Control7 example $MODEL MODEL $SUBROUTINES
$MODEL	Control6 example
MODEL	Control6 example
$MODEL	Control7 example
MODEL	Control7 example
$MODEL	$MODEL
MODEL	MODEL

	Bayes Example 7r
	Bayes Example 8
	Bayes Example 9
MULT	MULT
MUM	$ESTIMATION, $ESTM
	$ESTIMATION record options
MXNAME	SIZES FSIZES prsizes
MXSTEP	MXSTEP
NACTIV	Active ETA list for PREDPP
NAETA	Non-active ETA list for PRED
NAMES	$OMEGA
	$SIGMA
	$THETA
NBURN	Aneal
	Bayes Example 1
	Bayes Example 10
	Bayes Example 10l
	Bayes Example 1B
	Bayes Example 2
	Bayes Example 3
	Bayes Example 4
	Bayes Example 5
	Bayes Example 6
	Bayes Example 6s
	Bayes Example 6sb
	Bayes Example 7r
	Bayes Example 8
	Bayes Example 9
	$ESTIMATION, $ESTM
	$ESTIMATION record options
NCM	MODEL
NCM1	$AES
	$AESINITIAL
NCM, NCOMPS, NCOMPARTMENTS	$MODEL
NCONTR	NCONTR
NDREC	Additional_Record_Counters
	Record Counters: NIREC, NDREC
NEN	$DATA
NEPP	NWPRI
	NWPRI examples
	$PRIOR
	PRIOR
NEPS	NWPRI
	NWPRI examples
	$PRIOR
	PRIOR
NEQUILIBRIUM	$MODEL

NITER	Bayes Example 1 Bayes Example 10 Bayes Example 10l Bayes Example 1B Bayes Example 2 Bayes Example 3 Bayes Example 4 Bayes Example 5 Bayes Example 6 Bayes Example 6s Bayes Example 6sb Bayes Example 7r Bayes Example 8 Bayes Example 9 $ESTIMATION, $ESTM $ESTIMATION record options
NL2_OBS	CCONTR: Y, DATA, N1, N2, DIM
NL2_RECS	CONTR: III, DIM
NLUSER	SIZES FSIZES prsizes
NMPR1	Repetition_1 example Repetition_2 example
NMPR10	Repetition Variables Simulation: ETA, EPS
NMPR11	Simulation: IETAOL IEPSOL
NMPR12	YLO example
NMPR12, YLO, YUP	YLO YUP YLO YUP Probability: PR_Y
NMPR13, NMPR14, CTLO, CTUP	CTLO CTLO CTUP Probability: PR_CT CTUP
NMPR15	SKIP
NMPR16	NWPRI NWPRI examples PRIOR PRIOR Simulation: Parameters TNPRI
NMPR17	F_FLAG
NMPR50	PRIOR Simulation: ICMAX
NMPRD10	Abbreviated code
NMPRD15	Abbreviated code
NMPRD1, NMPRD2	PRED Error Message PRED Exit Code
NMPRD3	$ABBREVIATED COMACT, COMSAV COM COMACT COMSAV COMRES
NMPRD4	$ABBREVIATED Abbreviated code

	COMACT, COMSAV
	COM COMACT COMSAV COMRES
	Displayed PRED-Defined Items
	$PRED
	PRED-Defined Variables
NMPRD5	Correlation Across L2 Records
NMPRD5, NMPR6	Simulation: SIMEPS Error Code
NMPRD7	Abbreviated code
	ERROR
	Simulation: ETA, EPS
NMPRD8	Recursive PRED Indicator
NMPRD9	PASS
	PASS: PASSRC
:nnn	$TABLE
NO	$ABBREVIATED
	SIZES FSIZES prsizes
NOABORT	Abbreviated code
	$ESTIMATION, $ESTM
	$ESTIMATION record options
	Interoccasion_variability example
	$THETA
NOABORTFIRST	$THETA
NOABS0, NOABSZERO	$SCATTERPLOT
NOAPPEND	$TABLE
NOBS	CCONTR: Y, DATA, N1, N2, DIM
	CONTR: Y DATA NOBS
NOBSIND2	CONTR: Y DATA NOBS
NOBUILD	nmfe utility program
NOCHECKMU	$ABBREVIATED
NOCOMMON	$ABBREVIATED
NOCOV	$ESTIMATION, $ESTM
	$ESTIMATION record options
NOFASTDER	$ABBREVIATED
	Partial Derivative Indicators
NOFCOV	$COVARIANCE, $COVR
NOFIRSTDERCODE	$ABBREVIATED
NOFORWARD	Superproblem_1 example
	Superproblem_2 example
	$TABLE
NOHABORT	$ESTIMATION, $ESTM
	$ESTIMATION record options
NOHEADER	Simulation_1 example
	Simulation_2 example
	$TABLE
NOINTERACTION	$ESTIMATION, $ESTM

NOLABEL	$CHAIN $ESTIMATION, $ESTM $ESTIMATION record options $TABLE
NOLAPLACIAN, NOLAPLACEAN	$ESTIMATION, $ESTM
NOMSFTEST	$MSFI
NON-ACTIVE	Non-active ETA list for PRED
NONE	$WARNINGS
NON-EVENT	Calling protocol phrase
NONINFETA	$ESTIMATION, $ESTM $ESTIMATION record options
NONPARAMETRIC	RANDMT, RANDMTU RANDOM $SIMULATION
NONPARAMETRIC, NONP	COMACT, COMSAV Displayed PRED-Defined Items $NONPARAMETRIC Nonparametric Density: DEN_, CDEN_ Nonparametric example
$NONPARAMETRIC, NONP	$NONPARAMETRIC Nonparametric Density: DEN_, CDEN_ Nonparametric example
NOOFF	$MODEL
NOOPEN	$DATA Superproblem_1 example Superproblem_2 example
NOORD0, NOORDZERO	$SCATTERPLOT
NOPOSTHOC	$ESTIMATION, $ESTM
NOPREDICTION	$SIMULATION
NOPRINT	Phenobarb example Simulation_1 example Simulation_2 example $SUPER Superproblem_1 example Superproblem_2 example $TABLE
NOPRIOR	Bayes Example 1 Bayes Example 10 Bayes Example 10l Bayes Example 1B Bayes Example 2 Bayes Example 3 Bayes Example 6 Bayes Example 6s Bayes Example 6sb Bayes Example 7r Bayes Example 8

	$ESTIMATION, $ESTM
	$ESTIMATION record options
NOREPEAT, NOREPEAT1, NOREPEAT2	$ESTIMATION, $ESTM
NOREPLACE	$SIMULATION
NORESCALE	$MSFI
NORESET	$WARNINGS
NOREWIND	$DATA
	RECORDS=ID Example
	RECORDS=ID Example 2
	$SIMULATION
NORMAL	RANDMT, RANDMTU
	RANDOM
	$SIMULATION
	Simulation_2 example
NOSLOW	$COVARIANCE, $COVR
	$ESTIMATION, $ESTM
NOSUB	$ABBREVIATED
	$DEFAULT
	$ESTIMATION, $ESTM
	$ESTIMATION record options
	$SCATTERPLOT
	$TABLE
NOSUPRESET	$SIMULATION
NOTITLE	$CHAIN
	$ESTIMATION, $ESTM
	$ESTIMATION record options
	$TABLE
NO-U-TURN	$ESTIMATION, $ESTM
NOUTURN	$ESTIMATION, $ESTM
NOWIDE	$DATA
NPAR	MODEL
NPARAMETERS, NPARAMS	$MODEL
NPATSMAX	SIZES FSIZES prsizes
NPD	NONMEM output files
	$SCATTERPLOT
	$TABLE
NPDE	$SCATTERPLOT
	$TABLE
NPD, NPDE, NPDE_MODE	NPD, NPDE, NPDE_MODE, DV_LOQ, CI
NPDTYPE	$TABLE
NPE	NONMEM output files
NPESTIM	$NONPARAMETRIC
NPEXP	Bayes Example 6s
	Bayes Example 6sb
	NWPRI
	NWPRI examples

	$PRIOR
	PRIOR
NPI	NONMEM output files
NPL	NONMEM output files
NPMAXITER	$NONPARAMETRIC
NPOPETAS, POPETAS	$MSFI
NPOPEXTRA	SIZES FSIZES prsizes
NPOP_EXTSIZE	SIZES FSIZES prsizes
NPOP_EXTSIZE2	SIZES FSIZES prsizes
NPOPMIXMAX	SIZES FSIZES prsizes
NPRED	$SCATTERPLOT
	$TABLE
NPROB	Problem Iteration Counters
NPRSIZES	SIZES FSIZES prsizes
NPSUPP	$NONPARAMETRIC
NPSUPPE	$NONPARAMETRIC
NRD	$COVARIANCE, $COVR
	$ESTIMATION, $ESTM
	$SUBROUTINES
	$TOL
	TOL
NRDC	$TOL
	TOL
NREC	CCONTR: Y, DATA, N1, N2, DIM
NRECORDS, NRECS, RECORDS, RECS	$DATA
	FINISH
	RECORDS=ID Example
	RECORDS=ID Example 2
NREP	Simulation: NREP, IREP
NRES	$SCATTERPLOT
	$TABLE
NRETA	Active ETA list for PREDPP
NSAM	NWPRI
	NWPRI examples
	$PRIOR
	PRIOR
	TNPRI
	TNPRI examples
NSAMPLE	$CHAIN
	$ESTIMATION, $ESTM
	$ESTIMATION record options
NSIG	Bayes Example 1
	Bayes Example 10
	Bayes Example 10l
	Bayes Example 1B
	Bayes Example 2

	Bayes Example 3
	Bayes Example 4
	Bayes Example 5
	Bayes Example 7r
	$ESTIMATION record options
	Significant Digits from Est. Step
NSIGDIGITS, SIGDIGITS	$ESTIMATION, $ESTM
NSPOP	$MIX
	MIX
	Mixture_model example
NSTORE	SIZES FSIZES prsizes
NSUBS	CONTR: KCALL
	Simulation: NREP, IREP
NSUBS, NSUBPROBLEMS	$SIMULATION
	Simulation_2 example
	Simulation_3 example
NTHES_, NETAS_, NEPSS_	Parameter Dimensions
NTHETA	NWPRI
	NWPRI examples
	$PRIOR
	PRIOR
	$THETAI
	$THETAR
NTHP	Bayes Example 1B
	Bayes Example 6s
	Bayes Example 6sb
	NWPRI
	NWPRI examples
	$PRIOR
	PRIOR
	$THETAI
	$THETAR
NULL	$DATA
	Multiple_dose example
NUMBERPOINTS, NUMPTS	$THETA
NUMDER	$ESTIMATION, $ESTM
	$ESTIMATION record options
NUMERICAL	$ESTIMATION record options
NUMERICAL, NONUMERICAL	$ABBREVIATED
	$ESTIMATION, $ESTM
NUTS	$ESTIMATION, $ESTM
NUTS_BASE	$ESTIMATION, $ESTM
	$ESTIMATION record options
NUTS_DELTA	$ESTIMATION, $ESTM
	$ESTIMATION record options
NUTS_EPARAM	$ESTIMATION, $ESTM
	$ESTIMATION record options

NUTS_GAMMA	$ESTIMATION, $ESTM $ESTIMATION record options
NUTS_INIT	$ESTIMATION, $ESTM $ESTIMATION record options
NUTS_MASS	$ESTIMATION, $ESTM $ESTIMATION record options
NUTS_MAXDEPTH	$ESTIMATION, $ESTM $ESTIMATION record options
NUTS_OPARAM	$ESTIMATION, $ESTM $ESTIMATION record options
NUTS_REG	$ESTIMATION, $ESTM $ESTIMATION record options
NUTS_SPARAM	$ESTIMATION, $ESTM $ESTIMATION record options
NUTS_STEPINTER	$ESTIMATION, $ESTM $ESTIMATION record options
NUTS_STEPITER	$ESTIMATION, $ESTM $ESTIMATION record options
NUTS_TERM	$ESTIMATION, $ESTM $ESTIMATION record options
NUTS_TEST	$ESTIMATION, $ESTM $ESTIMATION record options
NUTS_TRANSFORM	$ESTIMATION, $ESTM $ESTIMATION record options
NVECX	SIZES FSIZES prsizes
NVNT	ERROR
NWIND	INFN PASS NEWIND: NWIND
NWPRI	Bayes Example 1 Bayes Example 10 Bayes Example 10l Bayes Example 1B Bayes Example 2 Bayes Example 3 Bayes Example 4 Bayes Example 6 Bayes Example 6s Bayes Example 6sb Bayes Example 7r Bayes Example 8 Bayes Example 9 PRIOR PRIOR Simulation: Parameters NWPRI NWPRI examples $PRIOR
NWRES	$SCATTERPLOT

	$TABLE
OACCEPT	$ESTIMATION, $ESTM $ESTIMATION record options
OBJ	OBJ
OBJECT	Finalization example ($PRED and $INFN) Objective Function Value Objective Function Value Individual Objective Function Value Individual exampl
OBJI	$SCATTERPLOT $TABLE
OBSERVATION	Calling protocol phrase
OBSONLY	$SCATTERPLOT
OCC	FINEDATA NMTEMPLATE
OLKJDF	$ESTIMATION, $ESTM $ESTIMATION record options $OLKJDF
$OLKJDF	$OLKJDF
OLNTWOPI	$ESTIMATION, $ESTM $ESTIMATION record options
OMEGA	FINAL PARAMETER ESTIMATE INITIAL ESTIMATE (INITIAL EST) Intra and inter-individual $OMEGA $OMEGAP, $OMEGAPD Parameters OMEGA SIGMA: Current Write print
$OMEGA	Control4 example $OMEGA
OMEGAF	Finalization example ($PRED and $INFN) Parameter Values: Initial and Final
OMEGAP	Bayes Example 1 Bayes Example 10 Bayes Example 10l Bayes Example 1B Bayes Example 2 Bayes Example 3 Bayes Example 4 Bayes Example 6 Bayes Example 7r Bayes Example 8 NWPRI $OMEGAP, $OMEGAPD
$OMEGAP	Bayes Example 1 Bayes Example 10 Bayes Example 10l Bayes Example 1B

	Bayes Example 2
	Bayes Example 3
	Bayes Example 4
	Bayes Example 6
	Bayes Example 7r
	Bayes Example 8
	NWPRI
	$OMEGAP, $OMEGAPD
OMEGAPD	Bayes Example 1
	Bayes Example 10
	Bayes Example 10l
	Bayes Example 1B
	Bayes Example 2
	Bayes Example 3
	Bayes Example 4
	Bayes Example 6
	Bayes Example 7r
	Bayes Example 8
	NWPRI
	$OMEGAP, $OMEGAPD
$OMEGAPD	Bayes Example 1
	Bayes Example 10
	Bayes Example 10l
	Bayes Example 1B
	Bayes Example 2
	Bayes Example 3
	Bayes Example 4
	Bayes Example 6
	Bayes Example 7r
	Bayes Example 8
	NWPRI
	$OMEGAP, $OMEGAPD
OMIT	Data_average example
	$OMIT
$OMIT	$OMIT
	RAW_ data item
OMITTED	$COVARIANCE, $COVR
	$ESTIMATION, $ESTM
	$ESTIMATION record options
	$SCATTERPLOT
	$TABLE
OMSIMP	NWPRI
	NWPRI examples
	PRIOR
	PRIOR Simulation: Parameters
	TNPRI
	Write print
ONCE	Calling protocol phrase

ONEHEADER	Phenobarb example
ONEHEADERALL	$TABLE
ONEHEADERPERFILE	$TABLE
ONLYREAD	$MSFI
ONLYSIMULATION, ONLY	Displayed PRED-Defined Items Plasma urine example $SIMULATION Simulation_1 example Simulation_2 example Simulation: ETA, EPS
OPEN	Superproblem_1 example Superproblem_2 example Write print
OPSCRS	CONTR SPTWO
OPTMAP	$ESTIMATION, $ESTM $ESTIMATION record options
ORD0, ORDZERO	Phenobarb example $SCATTERPLOT
ORDER	$CHAIN $ESTIMATION, $ESTM $ESTIMATION record options
OSAMPLE_M1	$ESTIMATION record options
OSAMPLE_M2	$ESTIMATION record options
OSAMPLE, OSAMPLE_M1, OSAMPLE_M2	$ESTIMATION, $ESTM
OTHER	Abbreviated Function Abbreviated Function Examples BIVARIATE FUNCTION $SUBROUTINES
OVARF	$ESTIMATION, $ESTM $ESTIMATION record options $ovarf
$OVARF	$ovarf
P	$MIX MIX Mixture_model example
PACCEPT	$ESTIMATION, $ESTM $ESTIMATION record options
PAL	SIZES FSIZES prsizes
PARAFILE	$COVARIANCE, $COVR $ESTIMATION, $ESTM $ESTIMATION record options $NONPARAMETRIC $SIMULATION $TABLE
PARAFILE, PARAFILE.PNM	FOCE_PARALLEL example

	State Vector Time: TSTATE
	$SUBROUTINES
$PK	Control3 example
	$PK
PK	PK
PL	SIZES FSIZES prsizes
PLEV	NWPRI
	NWPRI examples
	$PRIOR
	PRIOR
	TNPRI
	TNPRI examples
PLOG	PROTECT functions
PLOG10	PROTECT functions
P(n)	ADVAN6 ADVAN8 ADVAN13 ADVAN14
	ADVAN9_15
	$AES
	AES
	$AESINITIAL
	$DES
	DES
	$PK
PNG	PROTECT functions
Pnnnnn	$TABLE
PNP	PROTECT functions
POPETAS, NPOPETAS	$MSFI
POSDEF	$COVARIANCE, $COVR
POSTHOC	CCONTR
	Displayed PRED-Defined Items
	$ESTIMATION, $ESTM
	$ESTIMATION record options
	Finalization example ($PRED and $INFN)
	FNLETA
	Interoccasion_variability example
	Phenobarb example
	$TABLE
PPR	SIZES FSIZES prsizes
PRCOMG	Diff Eq Solver Settings
	PREDPP modules
PRCOMU	Bioavailability Behavior
	PREDPP modules
PRCOMW	ADVAN7: FACTOR
	PREDPP modules
PR_CT	CTLO
	CTLO CTUP Probability: PR_CT
	CTUP

PRDDE1	DES AES: ICALL, IDEFD, IDEFA
PRDDE1, PRDPK1, PRDPK2, PRDPK3, PRINFN	PREDPP modules
PRDFL	Abbreviated code Repetition_2 example
PRDPK1	Model Event Time: MTIME MTIME MNEXT MPAST MNOW MTDIFI
PRDPK2	Model Event Time: MTDIFF MTIME MNEXT MPAST MNOW MTDIFI
PRDPK3	Compartment Initialization: A_0 Compartment Initialization: A_0FLG Compartment Initialization Block
PRDPK4	Initial Steady State: I_SS, ISSMOD
PRDPK4, I_SS	PK
PRECOND	$COVARIANCE, $COVR
PRECONDS	$COVARIANCE, $COVR
PRED	Displayed PRED-Defined Items Non-active ETA list for PRED $PRED PRED PRED_RES_WRES $SCATTERPLOT $SUBROUTINES $TABLE
$PRED	Control4 example
PRED	Control4 example
$PRED	$PRED
PRED	PRED
PREDI	$SCATTERPLOT $TABLE
PREDICTION	$ESTIMATION record options $SIMULATION
PRETYPE	$COVARIANCE, $COVR
PRINFN	INFN-defined Variables
PRINT	Abbreviated code Control3 example $COVARIANCE, $COVR $ESTIMATION, $ESTM $ESTIMATION record options MONITORING OF SEARCH $SUPER Superproblem_1 example Superproblem_2 example $TABLE Write print
PRIOR	Bayes Example 1 Bayes Example 10

	Bayes Example 10l
	Bayes Example 1B
	Bayes Example 2
	Bayes Example 3
	Bayes Example 4
	Bayes Example 6
	Bayes Example 6s
	Bayes Example 6sb
	Bayes Example 7r
	Bayes Example 8
	Bayes Example 9
	PRIOR
	PRIOR Simulation: Parameters
	$SIMULATION
	Stubs
	$SUBROUTINES
$PRIOR	NWPRI
	NWPRI examples
	$PRIOR
	TNPRI
	TNPRI examples
PRIOR	NWPRI
	NWPRI examples
	$PRIOR
	PRIOR
	TNPRI
	TNPRI examples
PRIORC	$ESTIMATION, $ESTM
	$ESTIMATION record options
PROBLEM	$PROBLEM
	$SIZES
	$WARNINGS
$PROBLEM	Control3 example
	$PROBLEM
PROCM1	New Individual Indicator: NEWIND
PROCM1-PROCM13, PROCMA-PROCMC	PREDPP modules
PROCM2	Dose Time Non-Event: DOSTIM
PROCM3	Dose Record: DOSREC
PROCM4	State Vector: A
	State Vector Time: TSTATE
PROCM5	Active ETA list for PREDPP
PROCM6	DES AES: THETA
PROCM7	EVENT RECORD: EVTREC
PROCM8	WRITE Formats
PROCM9	State Vector: A
	State Vector Time: TSTATE
PROCMA	Model Event Time: MNOW, MPAST, MNE

	MTIME MNEXT MPAST MNOW MTDIFI
PROCMB	DES AES: ISFINL
PROCMC	Compartment Initialization: A_0 Compartment Initialization: A_0FLG Compartment Initialization Block
PROTECT	$ABBREVIATED PROTECT functions
PRRES	PRRES
PRSEQ	nmfe utility program
PR_Y	YLO example YLO YUP YLO YUP Probability: PR_Y
PSAMPLE_M1	Bayes Example 6s $ESTIMATION record options
PSAMPLE_M2	Bayes Example 6s $ESTIMATION record options
PSAMPLE_M3	Bayes Example 6s $ESTIMATION record options
PSAMPLE, PSAMPLE_M1, PSAMPLE_M2, PSAMPLE_M3	$ESTIMATION, $ESTM
PSCALE_MAX	$ESTIMATION record options
PSCALE_MIN	$ESTIMATION record options
PSCALE_MIN, PSCALE_MAX	$ESTIMATION, $ESTM
PSQRT	PROTECT functions
PW	SIZES FSIZES prsizes
PZR	PROTECT functions
Q	$PK TRANS3
Q2, Q3, Q4	$PK
Q, Q2, Q3, Q4	TRANS4
R	$COVARIANCE, $COVR Expectation block Simulation block
R1-R9	Dose event record $PK Rate parameter
RANDMTU, RANDMT	RANDMT, RANDMTU
RANDOM	RANDOM $SIMULATION Simulation_2 example Expectation block Initialization-Finalization block Simulation block
RANMETHOD	$CHAIN $COVARIANCE, $COVR $ESTIMATION, $ESTM $ESTIMATION record options

	$SIMULATION
	$TABLE
RATE	Dose event record
	Exogenous supplementation example
	$INPUT
	Multiple_dose example
	RATE data item
	Rate parameter
	SS dose event record
RAW_	Data_Average block
	Data_average example
	Displayed PRED-Defined Items
	Expectation example
	$INPUT
	$OMIT
	RAW_ data item
$RCOV	$RCOV, $RCOVI
$RCOVI	$RCOV, $RCOVI
RDATA	CONTR: Y DATA NOBS
	MIX: DATA
RECOMPUTE	$NONPARAMETRIC
RECORDS=ID	$DATA
	RECORDS=ID Example
	RECORDS=ID Example 2
RECORDS, RECS, NRECORDS, NRECS	$DATA
	FINISH
	RECORDS=ID Example
	RECORDS=ID Example 2
REPEAT	$ESTIMATION record options
REPEAT1	$ESTIMATION record options
REPEAT2	$ESTIMATION record options
REPEAT, REPEAT1, REPEAT2	$ESTIMATION, $ESTM
REPL	$DATA
REPL_	$INPUT
	REPL_ data item (NM75)
REPLACE	Bayes Example 7r
	$SIMULATION
REPLACE, REPLACEMENT	$ABBREVIATED
	Abbreviated Function Examples
	Infn_interpolation example 2
REPS	DOEXPAND
REQUESTFIRST	$SIMULATION
REQUESTSECOND	$SIMULATION
RES	Data_average example
	Displayed PRED-Defined Items
	Expectation example

	PRED_RES_WRES
	$SCATTERPLOT
	$TABLE
RESCALE	$MSFI
RESET	$WARNINGS
RESI	$SCATTERPLOT
	$TABLE
RESUME	$COVARIANCE, $COVR
RETURN	Data_Average block
	Expectation block
	Initialization-Finalization block
	PRED_IGNORE_DATA BLOCK (NM75)
	Simulation block
REWIND	$DATA
	RECORDS=ID Example
	RECORDS=ID Example 2
	$SIMULATION
	Write print
RFORMAT	FORMAT
	$TABLE
R-MATRIX	Bayes Example 1
	Bayes Example 10
	Bayes Example 10l
	Bayes Example 1B
	Bayes Example 2
	Bayes Example 3
	Bayes Example 4
	Bayes Example 5
	Bayes Example 6
	Bayes Example 6s
	Bayes Example 6sb
	Bayes Example 7r
	Bayes Example 8
	Bayes Example 9
RMT	$ESTIMATION, $ESTM
	NONMEM output files
RN	Dose event record
	$PK
	PK
	Rate parameter
ROCM0	MIX
	MIX CONTR: THETA
ROCM1	$CONTR
	CONTR
	CONTR: Y DATA NOBS
	MIX
	MIX: DATA

	SPTWO
ROCM10	Initialization-Finalization block Simulation: NREP, IREP
ROCM11	Mixture model: MIXNUM, MIXEST
ROCM12	Partial Derivative Indicators
ROCM13	CONTR: KCALL
ROCM14	Initialization-Finalization block Problem Iteration Counters
ROCM15	Non-active ETA list for PRED
ROCM16	Significant Digits from Est. Step
ROCM17	Initialization-Finalization block PASS New L2 record: NEWL2
ROCM18	Nonparametric Density: DEN_, CDEN_ Nonparametric example
ROCM1C	CONTR: Y DATA NOBS
ROCM2	CONTR: III, DIM
ROCM22	Parameters OMEGA SIGMA: Current
ROCM25	Mixture model: MIXP
ROCM28	Super Problem Print Control
ROCM29	Population single-Subject Indicator
ROCM3	CONTR: F, G, H
ROCM31	Data_Average block MIX CONTR: TEMPLT
ROCM32	Initialization-Finalization block Record Counters: NIREC, NDREC
ROCM34	Initialization-Finalization block PASS PASS NEWIND: NWIND
ROCM35	Parameter Dimensions Parameters OMEGA SIGMA: Current
ROCM4	CCONTR: Y, DATA, N1, N2, DIM $CONTR CONTR SPTWO
ROCM44	YLO example YLO YUP YLO YUP Probability: PR_Y
ROCM45	CTLO CTLO CTUP Probability: PR_CT CTUP
ROCM46	Initialization-Finalization block NINDR INDR1 INDR2
ROCM47	Mixture model: MIXPT
ROCM48	Initialization-Finalization block Size of Individual Record

ROCM49	Initialization-Finalization block PRED, RES, WRES
ROCM5	CCONTR: F, G, H
ROCM50	Objective Function Value Individual Objective Function Value Individual exampl
ROCM6	Finalization example ($PRED and $INFN) Initialization-Finalization block Parameters OMEGA SIGMA: Current Parameter Values: Initial and Final
ROCM7	Finalization example ($PRED and $INFN) Initialization-Finalization block Standard Errors
ROCM8	Finalization example ($PRED and $INFN) Initialization-Finalization block Objective Function Value
ROCM9	Estim Covar Error Codes Finalization example ($PRED and $INFN) Initialization-Finalization block
RPT_	$INPUT Repetition_1 example Repetition Variables RPT_ data item
RPTI	Repetition_2 example
RPTI, RPTO, RPTON, PRDFL	Repetition Variables
RPTO	Abbreviated code Repetition_2 example
RPTON	Abbreviated code Repetition_1 example
RPTO, RPTI	Repetition_1 example
RS_ALLOC1	SIZES FSIZES prsizes
RS_ALLOC2	SIZES FSIZES prsizes
RUNDIR, RUNPDIR, NMEXEC	nmfe utility program
RV_ALLOC1	SIZES FSIZES prsizes
RV_ALLOC2	SIZES FSIZES prsizes
S	$COVARIANCE, $COVR
S0	Scaling parameter
S0, S1-S10, SC	$PK
S1NUM, S2NUM, S1NIT, S2NIT, S1IT, S2IT	Problem Iteration Counters
SADDLE_HESS	$ESTIMATION, $ESTM $ESTIMATION record options
SADDLE_RESET	$ESTIMATION, $ESTM $ESTIMATION record options
SAEM	Aneal Bayes Example 1 Bayes Example 10 Bayes Example 10l

	Bayes Example 1B
	Bayes Example 2
	Bayes Example 3
	Bayes Example 4
	Bayes Example 5
	Bayes Example 7r
	Bayes Example 9
	$ESTIMATION, $ESTM
	$ESTIMATION record options
SAME	Interoccasion_variability example
	Mixture_model example
	$OMEGA
	$SIGMA
SAME(m)	$OMEGA
	$SIGMA
SCATTERPLOTS	COMACT, COMSAV
	COM COMACT COMSAV COMRES
	Displayed PRED-Defined Items
	Phenobarb example
	PRED-Defined Variables
	$SCATTERPLOT
$SCATTERPLOTS	Control3 example
	$SCATTERPLOT
SCO	SIZES FSIZES prsizes
SCOPE	Problem Iteration Counters
	$SUPER
	Superproblem_1 example
	Superproblem_2 example
SD	$OMEGA
	$SIGMA
	SIZES FSIZES prsizes
SDF	SIZES FSIZES prsizes
SEED	Bayes Example 1
	Bayes Example 1B
	Bayes Example 2
	$CHAIN
	$ESTIMATION, $ESTM
	$ESTIMATION record options
	$TABLE
SELECT	$CHAIN
	$ESTIMATION, $ESTM
SEOMEG	Write print
SEOMEG, SESIGM, SETHET, SETHETR	Standard Errors
SEOM, SESIG, SETH	Finalization example ($PRED and $INFN)
SESIGM	Write print
SETHET	Write print
SETHETR	Write print

SETHR	Finalization example ($PRED and $INFN)
SGSIMP	NWPRI NWPRI examples PRIOR PRIOR Simulation: Parameters Write print
SGSIMP, THSIMPR	TNPRI
SHK	$ESTIMATION, $ESTM NONMEM output files
SHM	$ESTIMATION, $ESTM ETASXI NONMEM output files
SIGDIGITS	$ESTIMATION record options
SIGDIGITS, NSIGDIGITS	$ESTIMATION, $ESTM $FORMAT Phenobarb example
SIGL	Bayes Example 1 Bayes Example 10 Bayes Example 10l Bayes Example 1B Bayes Example 2 Bayes Example 3 Bayes Example 4 Bayes Example 5 Bayes Example 6 Bayes Example 6s Bayes Example 6sb Bayes Example 7r Bayes Example 9 $COVARIANCE, $COVR $ESTIMATION record options
SIGLO	$COVARIANCE, $COVR $ESTIMATION record options
SIGL, SIGLO	$ESTIMATION, $ESTM
SIGMA	FINAL PARAMETER ESTIMATE INITIAL ESTIMATE (INITIAL EST) Intra and inter-individual Parameters OMEGA SIGMA: Current $SIGMA $SIGMAP, $SIGMAPD Write print
$SIGMA	Control4 example $SIGMA
SIGMAF	Finalization example ($PRED and $INFN) Parameter Values: Initial and Final
SIGMAP	Bayes Example 1 NWPRI

	$SIGMAP, $SIGMAPD
$SIGMAP	Bayes Example 1 NWPRI $SIGMAP, $SIGMAPD
SIGMAPD	Bayes Example 1 NWPRI $SIGMAP, $SIGMAPD
$SIGMAPD	Bayes Example 1 NWPRI $SIGMAP, $SIGMAPD
SIGNAL	SIGNAL
SIMEPS	Correlation Across L2 Records ERROR PRED SIMEPS $SIMULATION Simulation: ETA, EPS Simulation: IETAOL IEPSOL Simulation: SIMEPS Error Code Simulation block
SIMETA	ERROR PK PRED SIMETA $SIMULATION Simulation: ETA, EPS Simulation: IETAOL IEPSOL Simulation block
SIMPARON	nmfe utility program
SIMULATION	Simulation: IETAOL IEPSOL
SIMULATION, SIMULATE	Compartment Initialization Block CONTR: KCALL Logistic regression example Nonparametric example Plasma urine example RANDMT, RANDMTU RANDOM SIMEPS SIMETA $SIMULATION Simulation_1 example Simulation_2 example Simulation_3 example Simulation: ETA, EPS Simulation: NREP, IREP Superproblem_1 example Superproblem_2 example Verbatim code

$SIMULATION, $SIMULATE	Nonparametric example Plasma urine example
SIMULATION, SIMULATE	Plasma urine example
$SIMULATION, $SIMULATE	$SIMULATION Simulation_1 example
SIMULATION, SIMULATE	Simulation_1 example
$SIMULATION, $SIMULATE	Simulation_2 example
SIMULATION, SIMULATE	Simulation_2 example
$SIMULATION, $SIMULATE	Simulation_3 example
SIMULATION, SIMULATE	Simulation_3 example
SIN	Abbreviated code
SIRCENTER	$COVARIANCE, $COVR
SIRCLOCKSEED	$COVARIANCE, $COVR
SIRDF	$COVARIANCE, $COVR
SIRMAXWT	$COVARIANCE, $COVR
SIRMINWT	$COVARIANCE, $COVR
SIRNITER	$COVARIANCE, $COVR
SIRPARAFPRINT	$COVARIANCE, $COVR
SIRPRINT	$COVARIANCE, $COVR
SIRSAMPLE	$COVARIANCE, $COVR
SIRSEED	$COVARIANCE, $COVR
SIRTHBND	$COVARIANCE, $COVR
SIZES	$PROBLEM $SIZES $WARNINGS
$SIZES	$SIZES
SKIP	Abbreviated code $BIND $DATA DATE data item $INPUT SKIP
SLKJDF	$ESTIMATION, $ESTM $ESTIMATION record options $SLKJDF
$SLKJDF	$SLKJDF
SLOW	$COVARIANCE, $COVR $ESTIMATION, $ESTM $ESTIMATION record options
SMT	$ESTIMATION, $ESTM NONMEM output files
SN	$PK PK Scaling parameter
SORT	$ESTIMATION record options

SORT, NOSORT	$ESTIMATION, $ESTM
SOURCE_EPS	$SIMULATION
SPECIAL	$COVARIANCE, $COVR
SPTWO	PRED_RES_WRES SPTWO Stubs $SUBROUTINES
SQRT	Abbreviated code
SS	Absorption lag parameter Dose event record Exogenous supplementation example $INPUT Interoccasion_variability example SS data item SS dose event record
SS6	Diff Eq Solver Settings
SS6, SS9	SS6 SS9
SS9	Diff Eq Solver Settings
SSATOL	ADVAN6 ADVAN8 ADVAN13 ADVAN14 $SUBROUTINES $TOL TOL
SSATOLC	ADVAN6 ADVAN8 ADVAN13 ADVAN14
SS, SS1-SS7, SS9	$SUBROUTINES
SS, SS1-SS7, SS9, SS11-12	SS option
SSS, SSS0	SS option
SSTOL	$SUBROUTINES $TOL TOL
STANDARD	$OMEGA $SIGMA
STDOBJ	$ESTIMATION, $ESTM $ESTIMATION record options
STEP	DOEXPAND
STIELTJES	$ESTIMATION, $ESTM $ESTIMATION record options Stieltjes example
STIELTJ_SIZE	SIZES FSIZES prsizes
STOP_TIME,	STOP_TIME, ITASK_ (NM75)
STRAT, STRATF	$NONPARAMETRIC $SIMULATION
SUBPROBLEMS	$SIMULATION Simulation_2 example Simulation_3 example
SUBROUTINE	Simulation: IETAOL IEPSOL $SUBROUTINES

$SUBROUTINES	Control3 example $SUBROUTINES
SUBS, SUBROUTINES	SUBROUTINE $SUBROUTINES
SUPER	$PROBLEM $SIZES $WARNINGS
$SUPER	Problem Iteration Counters $SUPER Superproblem_1 example Superproblem_2 example
SUPP	SUPP Initialization-Finalization block
SUPRESET	$SIMULATION
SVARF	$ESTIMATION, $ESTM $ESTIMATION record options $svarf
$SVARF	$svarf
T	$AES AES $AESINITIAL $DES DES
TABLE	COMACT, COMSAV COM COMACT COMSAV COMRES Displayed PRED-Defined Items Phenobarb example PRED-Defined Variables Simulation_1 example Simulation_2 example $TABLE
$TABLE	Control3 example Phenobarb example
TABLE	Phenobarb example
$TABLE	Simulation_1 example
TABLE	Simulation_1 example
$TABLE	Simulation_2 example
TABLE	Simulation_2 example
$TABLE	$TABLE
TAN	Abbreviated code
TBLN	$CHAIN $ESTIMATION, $ESTM $ESTIMATION record options $ETAS, $PHIS $RCOV, $RCOVI
TDELTA	FINEDATA

	NMTEMPLATE
T-DISTRIBUTION	$SIMULATION
TEMPLT	Data_Average block MIX CONTR: TEMPLT Mixture model: MIXPT $OMIT
THBND	$COVARIANCE, $COVR
THETA	$AES AES $DES DES DES AES: ICALL, IDEFD, IDEFA DES AES: THETA FINAL PARAMETER ESTIMATE INITIAL ESTIMATE (INITIAL EST) MIX CONTR: THETA $THETA $THETAI $THETAP, $THETAPV $THETAR
$THETA	Control3 example Control4 example INITIAL ESTIMATE (INITIAL EST) $THETA
THETAC	MIX CONTR: THETA
THETAF	Finalization example ($PRED and $INFN) Parameter Values: Initial and Final
THETAFR	Finalization example ($PRED and $INFN) Parameter Values: Initial and Final
THETAI	Stubs $THETAI
$THETAI	$THETAI
THETA(n)	$ERROR ERROR INFN $PK PK $PRED PRED
THETAP	Bayes Example 1 Bayes Example 1B Bayes Example 8 NWPRI $THETAI $THETAP, $THETAPV $THETAR
$THETAP	Bayes Example 1

	Bayes Example 1B
	Bayes Example 8
	NWPRI
	$THETAP, $THETAPV
THETAPI	$THETAI
	$THETAR
THETAPV	Bayes Example 1
	Bayes Example 1B
	Bayes Example 8
	NWPRI
	$THETAP, $THETAPV
$THETAPV	Bayes Example 1
	Bayes Example 1B
	Bayes Example 8
	NWPRI
	$THETAP, $THETAPV
THETAR	Stubs
	$THETAR
$THETAR	$THETAR
THIN	$ESTIMATION, $ESTM
	$ESTIMATION record options
THSIMP	NWPRI
	NWPRI examples
	PRIOR
	PRIOR Simulation: Parameters
	TNPRI
	Write print
THSIMPR	NWPRI
	NWPRI examples
	PRIOR
	PRIOR Simulation: Parameters
	Write print
TIME	Calling protocol phrase
	Control5 example
	DATE data item
	$INPUT
	TIME data item
TNPRI	PRIOR
	PRIOR Simulation: Parameters
	$PRIOR
	TNPRI
	TNPRI examples
TO	$MODEL
	$SCATTERPLOT
	$TABLE
TOL	ADVAN10
	ADVAN6 ADVAN8 ADVAN13 ADVAN14

	ADVAN9_15
	ADVAN (general)
	Control7 example
	$COVARIANCE, $COVR
	$SUBROUTINES
	$TOL
	TOL
$TOL	$TOL
TOL	$TOL
	TOL
TOLC	ADVAN6 ADVAN8 ADVAN13 ADVAN14
TOS	$COVARIANCE, $COVR
TPRDEFAULT	nmfe utility program
TPU	$ESTIMATION record options
TRANS	TRANS (subroutine)
TRANS1	Bayes Example 3
	Bayes Example 4
	Bayes Example 5
	Bayes Example 6
	Bayes Example 6s
	Bayes Example 6sb
	TRANS1
TRANS2	ADVAN1
	ADVAN2
	Bayes Example 10
	Bayes Example 10l
	Bayes Example 7r
	Interoccasion_variability example
	Phenobarb example
	TRANS2
TRANS3	ADVAN3
	ADVAN4
	TRANS3
TRANS4	ADVAN11
	ADVAN12
	ADVAN3
	ADVAN4
	Bayes Example 1
	Bayes Example 1B
	Bayes Example 2
	Bayes Example 8
	Bayes Example 9
	FOCE_PARALLEL example
	Multiple_dose example
	TRANS4
TRANS5	ADVAN3
	ADVAN4

	Exogenous supplementation example TRANS5
TRANS6	ADVAN11 ADVAN12 ADVAN3 ADVAN4 TRANS6
TRANSLATE	$DATA
TRANS, TRANS1-TRANS6	$SUBROUTINES TRANS (option)
TRSKIP, PRSAME, PRDEFAULT, PRCOMPILE	nmfe utility program
TRUE	Nonparametric example $SIMULATION
TSCALE, XSCALE	$PK PK Time scale parameter
TSTART	FINEDATA NMTEMPLATE
TSTATE	$PK State Vector: A State Vector Time: TSTATE
TSTOP	FINEDATA NMTEMPLATE
TTDF	$ESTIMATION, $ESTM $ESTIMATION record options $SIMULATION $TTDF (NM75)
$TTDF	$TTDF (NM75)
UNCONDITIONAL	$NONPARAMETRIC $SCATTERPLOT $TABLE
UNIFORM	RANDMT, RANDMTU RANDOM $SIMULATION
UNINT, UNINTERESTING	$OMEGA $SIGMA $THETA
UNIT	Control3 example $SCATTERPLOT
USMETA	$SUBROUTINES
U-TURN	$ESTIMATION, $ESTM
UTURN	$ESTIMATION, $ESTM
V	$PK TRANS2 TRANS3
V1, V2, V3, V4	$PK

	TRANS4
VALUES	$OMEGAP, $OMEGAPD
VALUE, VALUES	$OMEGA $SIGMA
VARCALC	$TABLE
VARIANCE	$OMEGA $SIGMA
VECTOR	$ABBREVIATED
VECTR	$TABLE
VECTR, VECTRA-VECTRZ	Abbreviated code Abbreviated Function Abbreviated Function Examples Transit Compartment Examples
VERSION	$MSFI
VM	ADVAN10
VPD	$ESTIMATION, $ESTM NONMEM output files
VPT	$ESTIMATION, $ESTM NONMEM output files
VRBL	PRED-Defined Variables
VS	$SCATTERPLOT
VSS	$PK TRANS3
WARNINGMAXIMUM, WMAX	$WARNINGS
WARNINGS	$WARNINGS
WHILE	Abbreviated code SIMETA Transit Compartment Examples
WIDE	$DATA
WRES	Displayed PRED-Defined Items PRED_RES_WRES $SCATTERPLOT $TABLE
WRESCHOL	$TABLE
WRESI	$SCATTERPLOT $TABLE
WRITE	Abbreviated code Write print
XMLOFF, XMLOFF, FLUSHTIME	nmfe utility program
XN	$OMEGA $SIGMA $THETA
XSCALE, TSCALE	$PK PK Time scale parameter

XVID1-XVID5	EVID data item $INPUT
Y	CCONTR: Y, DATA, N1, N2, DIM CONTR: Y DATA NOBS $ERROR $PRED
YLO	YLO example
ZERO	$ESTIMATION, $ESTM $ESTIMATION record options

****	NONMEM file control stream
-1000000000	raw_output_file
1000000000	raw_output_file
-1000000001	raw_output_file
1000000001	raw_output_file
-1000000002	raw_output_file
1000000002	raw_output_file
-1000000003	raw_output_file
1000000003	raw_output_file
-1000000004	raw_output_file
1000000004	raw_output_file
-1000000005	raw_output_file
1000000005	raw_output_file
-1000000006	raw_output_file
1000000006	raw_output_file
-1000000007	raw_output_file
1000000007	raw_output_file
-1000000008	raw_output_file
1000000008	raw_output_file
1.0E+06	$ESTIMATION,$ESTM INDIVIDUALS $OMEGA RECORDS=ID Example RECORDS=ID Example 2 $SIGMA
160	$INPUT
2000	$DATA
2000000000	raw_output_file
2000000001	raw_output_file
2000000002	raw_output_file
2000000003	raw_output_file
-2ll	F_FLAG MDVRES
-2loglikelihood,-2llikelihood	F_FLAG MDVRES
48	$DES
67000	$INPUT nmfe utility program
68	Enterhepatic circulation examples MINIMUM VALUE OF OBJ. FUNCTION
80	$INPUT
a	TRANS5
abbreviated_code,abbreviated	$ABBREVIATED

numder	additional_output_files
obj	NONMEM error messages
objective_function	CCONTR CELS CONTR CRIT ELS $ESTIMATION,$ESTM MULT NCONTR NWPRI OBJ Objective Function Value Objective Function Value Individual Objective Function Value Individual example parallel computing PRIOR SPTWO TNPRI MINIMUM VALUE OF OBJ. FUNCTION MONITORING OF SEARCH
Objective_function_value_individual	Objective Function Value Individual
#OBJN	NONMEM output files
OBJN	NONMEM output files
objquick	$ESTIMATION,$ESTM
#OBJS	NONMEM output files
OBJS	NONMEM output files
#OBJT	NONMEM output files
OBJT	NONMEM output files
#OBJT	raw_output_file
OBJT	raw_output_file
#OBJV	NONMEM output files
OBJV	NONMEM output files
observation_compartment	CMT PCMT data item Default compartment
observation_event,observation	EVID data item
observed_value,observation	DV and MDV data items INDIVIDUALS
obs_recs,observation_record,observations	OBSERVATION RECORDS
occasion	Interoccasion_variability example
odd-type_data	ERROR $ESTIMATION,$ESTM Logistic regression example PRED PRED_RES_WRES $SIMULATION

users_guide	USERS GUIDE
user_supplied_routine	CCONTR
	CONTR
	CRIT
	INFN
	$MIX
	MIX
	NONMEM modules
	PREDPP modules
	PRIOR
	SPTWO
	Stubs
	variables in modules
utility_routine	Abbreviated Function
	Abbreviated Function Examples
	CELS
	CHOL
	CONSTRAINT
	DDEXPAND
	DOEXPAND
	ELS
	FILES
	FINEDATA
	Finedata example
	FLU
	GAMLN
	GETETA
	LBLR
	LPCC
	MULT
	NCONTR
	NEFF,NEFFI
	nmfe utility program
	NMTEMPLATE
	NWPRI
	PASS
	PHI
	priorget
	Probability Density Functions
	RANDMT,RANDMTU
	RANDOM
	SIGNAL
	SIMEPS
	SIMETA
	SUPP
	TABLE_COMPARE
	TABLE_QUANT
	TABLE_RESAMPLE
	TABLE_TO_XML

$ABBREVIATED

MEANING: Provides instructions about abbreviated code
CONTEXT: NM-TRAN Control Record

USAGE:

```
$ABBREVIATED [COMRES=n1] [COMSAV=n2]
             [DERIV2=NO] [DERIV2=NOCOMMON] [DERIV1=NO]
             [FASTDER | NOFASTDER]
             [CHECKMU | NOCHECKMU]
             [DES=COMPACT|DES=FULL]
             [REPLACE left_string = right_string ] ...
             [DECLARE [type] [DOWHILE] name [(dimension [,dimension])] ...
             [PROTECT]
             [FUNCTION function_name(input_vector_name,dimension[,usage])]
             [VECTOR input_vector_name(dimension)]
```

SAMPLE:

```
$ABBREVIATED    COMRES=2
```

DISCUSSION:
Optional. May be used when $PK, $ERROR, or $PRED abbreviated code is present. Must precede all blocks of abbreviated code. With NONMEM 7.4, may also be used when there is no abbreviated code. For example, $ABBR REPLACE may be used for label substitution in NONMEM report files.

OPTIONS:

`COMRES=`n_1 ('common reserve')

COMRES gives instructions to NM-TRAN.
Values of n_1:

-1 Do not store any variables in the global variable array, VRBL, in the NMPRD4 module.

0 Store variables in NMPRD4 with no reserved positions (the default)

n_1 Store variables in NMPRD4, but reserve the first n1 positions

With abbreviated code, the Ith position in NMPRD4 is referenced by COM(I).

This option is intended for advanced users of NONMEM, e.g., when abbreviated code is combined with user-supplied subroutines or verbatim code. A user-supplied subroutine may reserve the first n_1 positions in NMPRD4 for its use, in which case the option `COMRES` should be set to n_1 to instruct NM-TRAN to skip these positions; the first position used by NM-TRAN for storing variables defined in abbreviated code will be position $n_1 + 1$.

$TABLE and $SCATTER may explicitly reference variables in reserved positions 1 through n_1 by `COM(1)` through `COM(`n_1`)`, respectively, in addition to listing variables defined in abbreviated code by name.

An individual block of abbreviated code (e.g. $PK) may include the pseudo-statement `COMRES=-1`, which prevents any variable defined in that particular block from being stored in NMPRD4.

`COMSAV=` n_2 ('common save')

Values of variables displayed in tables and scatterplots are obtained from

NMPRD4. There are particular times when data records are passed to PRED for the purpose of obtaining these values; these are called copying passes. The SAVE region of NMPRD4 is the initial part of this array. If a variable is stored in the SAVE region, then the value of the variable computed with a given data record during a copying pass will be found in NMPRD4 when the same record is passed during the next copying pass, i.e. it will have been saved from the previous copying pass. This is in contrast to the usual behaviour, where with a given data record, the value in NMPRD4 is the value computed with the previous data record.
n_2 is the initial size of the SAVE region, i.e. the number of positions in this region. $n_2 = 0$ is the default value. n_2 may not exceed n_1.
The SAVE region has size n_2 initially, but NM-TRAN may extend it if SAVE variables are used. However, if $n_2 = -1$, the SAVE region is not to be extended, and there is to be no SAVE region altogether.
(See **copying block**).

When PREDPP is used, and a $PK block is present, NM-TRAN inserts code into the PK routine that stores the value of COMSAV at ICALL=1. If no $PK block is present, and a $ERROR block is present, the code is inserted into the ERROR routine. When PREDPP is not used, and a $PRED block is present, the generated or library PRED routine stores the value of COMSAV at ICALL<=1.

`DERIV1=NO` (NM74)
Prevents the computation of first derivatives. For example, may be used when only SAEM or BAYES is performed, or IMPMAP/ITS/FOCE are performed using OPTMAP>0 and ETADER>0. The global variable NOFIRSTDERCODE is set to 1.

`DERIV2=NO`
Prevents the computation of second derivatives, which are needed only for the Laplacian method.

`DERIV2=NOCOMMON`
Permits the computation of these derivatives, but prevents them from being stored in the global variable NMPRD4.
`$ESTIMATION METHOD=COND LAPLACIAN` may be specified, but variables representing second derivatives are not stored in NMPRD4. Therefore, they cannot be displayed in tables and scatterplots. In addition, no variables computed in the $PK block may be referenced in the $ERROR block. This is true whether or not these variables happen to have second derivatives, and whether or not the Laplacian method is used.

`FASTDER`, `NOFASTDER`
With NONMEM 7.2 and higher, NM-TRAN collects statements that compute first-partial eta-derivatives together in FSUBS, and they are performed only when NONMEM sets `IFIRSTEM`=1. `NOFASTDER` prevents NM-TRAN from doing this, and restores the order of statements in FSUBS to what it was in previous versions.

`FASTDER` requests that the statements be collected, and is the default.
(See **Partial Derivative Indicators**).

`CHECKMU`, `NOCHECKMU`
With NONMEM 7.2 and higher, NM-TRAN checks the MU model statements in abbreviated code and issues warning messages if they appear to contain mistakes. This can take a long time for large control streams. Also, in the examples directory, there are control streams for which the check is too difficult for NMTRAN

(tdist6_sim.ctl and tdist7.ctl), and some for which the warnings are inappropriate (the "superid" control streams that use $LEVEL). NOCHECKMU can be used to prevent NM-TRAN from attempting to check the MU model statements.

CHECKMU requests that MU model statements be checked, and is the default. Neither option affects the generated code.

DES=FULL

Arrays of the DES routine are stored in non-compact form.
With $ESTIMATION METHOD=COND LAPLACIAN, the option NUMERICAL is also required.
DES=FULL is the default with ADVAN9 and ADVAN15 and ADVAN17. (Prior to NONMEM 7.4, FULL was required with ADVAN13.)

DES=COMPACT

Arrays of the DES routine are stored in compact form.
Required with Laplacian method; optional otherwise. This is the default, except with ADVAN9 and ADVAN15 and ADVAN17.

REPLACE left_string = right_string

There are several different forms of the REPLACE option. They do not affect verbatim code. Case is ignored.

(1) Simple replacement
REPLACE left_string = right_string

May be used in all blocks of abbreviated code and $TABLE and $SCATTER records. Left_string is replaced by right_string. The search is "anchored" by a Fortran identifier. That is, left_string is parsed for the first Fortran identifier and the abbreviated code is searched for the same identifier. Following the identifier, the entire left_string must match the abbreviated code. Single or double quotes around a string are optional. They are necessary if the string contains punctuation such as a comma. A given line of abbreviated code may have multiple replacements; they are applied in the order that they appear in the NM-TRAN control stream.

Examples:

```
$ABBR REPLACE PI=3.14159265
$ABBR REPLACE THETA(CL)=THETA(4)
$ABBR REPLACE ETA(CL)=ETA(5)
$ABBR REPLACE K34="3,4"
 ...
CL=PI*THETA(CL)*EXP(ETA(CL))
PRINT *,OMEGA(K34)
```

The post-replacement code is

```
CL=3.14159265*THETA(4)*EXP(ETA(5))
PRINT *,OMEGA(3,4)
```

With this feature, subscripts of THETA, ETA, and EPS and ERR may be given symbolic names in abbreviated code, and $ABBR REPLACE is used to replace them with integer subscripts. With NONMEM 7.4, this feature also requests *label substitution* in NONMEM report files and table files. For example, the following will cause all appearances of "ETA(CL)" to be replaced by "ETA(3)" in the generated subroutine, and will cause "ETA(3)" to be replaced by "ETA(CL)" in the NONMEM report and any table file in which "ETA(3)" is listed.

```
$ABBR REPLACE ETA(CL)=ETA(3)
```

Label substitution can be turned off for an entire problem with NOSUB=1 option of the $DEFAULT record. Label substitution can be turned off for a specific NONMEM task (and all subsequent tasks) with the NOSUB=1 option of $TABLE, $SCAT, and $ESTIMATION.

Substitutions will never be made in the additional output files *.ext, ,phi, etc., to maintain their third party software readability.

With this feature, compartment names may be used instead of compartment numbers. For example,

```
$ABBR REPLACE A(DEPOT)=A(1)
$ABBR REPLACE DADT(DEPOT)=DADT(1)
  ...
$DES
 DADT(DEPOT)=-KA*A(DEPOT)
```

This is called "explicit" compartment name substitution. With NONMEM 7.5, compare the "implicit" compartment name substitution feature of the $MODEL record.

See **INTRODUCTION TO NONMEM 7, Symbolic Label Substitutions of Model Compartments**

(2) Replacement with selection by data item

$ABBR REPLACE VAR(d)=VAR($n_1$,$n_2$,..,$n_k$)

May be used in $PK, $ERROR, $PRED blocks only. VAR must one of ETA, EPS, THETA. (ERR is not permitted.) d must be a data item. The integer value of d (i.e., INT(d)) is used to select one of n_1,n_2,..,n_k. If any of the n_i is 0, that position is skipped. The effective code is:

```
IF (d.eq.1) VAR(d)=VAR(n1)
IF (d.eq.2) VAR(d)=VAR(n2)
 ...
```

The actual generated code uses Q-type indicator variables, and variables such as THETA_OCC_ in place of THETA(OCC).

Example: Suppose OCC is a data item that takes values 1 and 2.

```
$ABBR REPLACE THETA(OCC)=THETA(4,7)
  ...
$PK
TVCL=THETA(OCC)
```

The effective code after replacement is

```
IF (OCC==1) TVCL=THETA(4)
IF (OCC==2) TVCL=THETA(7)
```

A user-defined variable can be given a default value in case the implied selection is not satisfied. E.g., suppose there are 3 doses per subject (DOSN=1,2,3) and the model is:

```
$PK ...
   IF(DOSN.EQ.1) F1=1
   IF(DOSN.EQ.2) F1=1*EXP(ETA(2))
   IF(DOSN.EQ.3) F1=1*EXP(ETA(3))
```

The model can be implemented using implied selection as follows:

```
$ABBR REPLACE ETA(DOSN)=ETA(0,2,3)
```

```
$PK ....
  F1=1
  IF (DOSN>1) F1=1*EXP(ETA(DOSN))
```

(3) Replacement with selection by data item and parameter

`$ABBR REPLACE VAR(p_d)=VAR(`n_1`,`n_2`,..,`n_k`)`

p is a user-defined variable in the abbreviated code. The order may be p_d or d_p This allows a given data item d to be used as a selection variable for more than one parameter, with a different choice of elements of VAR.

Example 1:

```
$ABBR REPLACE THETA(SID_KA)=THETA(4,6)
$ABBR REPLACE THETA(SID_CL)=THETA(5,7)
   ...
$PK
KA=THETA(SID_KA)
CL=THETA(SID_CL)
```

which is equivalent to

```
$PK
IF (SID==1) KA=THETA(4)
IF (SID==2) KA=THETA(6)
IF (SID==1) CL=THETA(5)
IF (SID==2) CL=THETA(7)
```

Example 2:

```
$ABBR REPLACE ETA(OCC_CL)=ETA(5,3)
$ABBR REPLACE ETA(OCC_V) =ETA(6,4)
   ...
$PK
CL=TVCL*EXP(ETA(1)+ETA(OCC_CL))
V =TVV *EXP(ETA(2)+ETA(OCC_V))
```

which is equivalent to

```
$PK
IF (OCC==1) CL=TVCL(EXP(ETA(1)+ETA(5))
IF (OCC==2) CL=TVCL(EXP(ETA(1)+ETA(3))
IF (OCC==1) V=TVV(EXP(ETA(2)+ETA(6))
IF (OCC==2) V=TVV(EXP(ETA(1)+ETA(4))
```

The number of values specified for the selection data item must be consistent for all parameters in which it is used.

(4) Replacement for multiple variables

`$ABBR REPLACE VAR(`p_1`,`p_2`,...,`p_k`)=VAR(`n_1`,`n_2`,...,`n_k`)`

This form is new to NONMEM 7.4. The p_i are character strings separated by commas. A character string may not contain an embedded space. The lists on both left and right sides must be of the same length. VAR(p_i) is replaced by VAR(n_i). For example,

```
$ABBR REPLACE THETA(CL,V1,Q,V2)=THETA(1,2,3,4)
```

This is equivalent to:

```
$ABBR REPLACE THETA(CL)=THETA(1)
$ABBR REPLACE THETA(V1)=THETA(2)
$ABBR REPLACE THETA(Q)=THETA(3)
$ABBR REPLACE THETA(V2)=THETA(4)
```

Label substition occurs with this form, as with Simple replacement (1).

Short-hand notation

A short-hand notation may be used to describe a series of values of n_i. An ascending sequence of n_i can be described as
,start TO end [BY interval]

BY is optional and must be > 0. Default is 1. TO is required. If end < start, the sequence is ignored. At least one comma must appear, so NMTRAN knows it is a number list, not a variable name. With NONMEM 7.4, the character : may be used instead of TO, and the value of BY may be negative.

EXAMPLES:

```
$ABBR REPLACE THETA(SID_KA)=THETA(10:4 by 3) ; order: 10,7,4
$ABBR REPLACE THETA(SID_KA)=THETA(4 to 10 by -3) ; order: 10,7,4
$ABBR REPLACE THETA(SID_KA)=THETA(10 to 4) ; order: 10,9,8,7,6,5,4

$ABBR REPLACE THETA(SID_KA)=THETA(,4 to 13 by 3,25 to 37 by 4)
```

is identical to

```
$ABBR REPLACE THETA(SID_KA)=THETA(4,7,10,13,25,29,33,37)
```

Files FORIG and FREPL

When $ABBR REPLACE is coded, NM-TRAN produces two files:
FORIG Contains the original (pre-replacement) abbreviated code,
and task specification records $TABLE and $SCATTER.
FREPL Contains the new (post-replacement) abbreviated code
and task specification records.
These may be helpful for debugging.

`DECLARE` [INTEGER] [DOWHILE] name [(dimension [,dimension])] ...

One or names may be coded. They are referred to as "declared variables." IF INTEGER or DOWHILE is coded, the type of the variable is integer. Otherwise, the type of the variable is double precision. If one or two dimensions are declared, the variable being declared is an array. Each dimension may be a number or a variable or an expression. Constants defined in SIZES (e.g., NO, LVR) may be used. Multiple DECLARE options may be coded. The characters "DECLARE" are optional after the first. Commas are ignored, and type and dimensions must be respecified as needed. No other options of $ABBR may appear after the DECLARE option(s). Declared variables are global, i.e., are defined in all blocks of abbreviated code. The number of subscripts must agree with the number of dimensions in the declaration. Declared variables that are not INTEGER or DOWHILE will be random variables if they are assigned in a statement whose right-side involves ETA's or EPS's.

EXAMPLE:

```
$ABBR DECLARE A,B(10),C(1,NO-1),INTEGER I J
```

Only I is INTEGER.

Variables may be declared as INTEGER or DOWHILE for use as subscripts of declared arrays or reserved variables that are arrays (but not of ETA, ERR, or EPS).

Variables used as looping indices in DOWHILE statements must be declared as DOWHILE variables.

Declared variables are automatically initialized to 0.

Elements of a declared array may be displayed in WRITE/PRINT statement, but not the entire array. E.g., the following is permitted

```
PRINT *,B(1),B(2)
```

but not

```
PRINT *,B
```

Neither individual elements nor the entire array may be listed in $TABLE. The workaround is to code, e.g.,

```
      ... code assigning values to B(1) etc. ...
   B1=B(1)
 $TABLE B1
```

(See **dowhile block**).

`PROTECT`

With NONMEM 7.4, a series of routines are available that protect against domain violations, divide by zero, and floating point overflows. Each of these routines start with the letter P, followed by the name of the mathematical operation they are to perform. For example, PLOG is the protective code routine that performs the LOG operation. With $ABBR PROTECT, NMTRAN will automatically replace all relevant function names with the P name.

(See **protect functions**).

`FUNCTION function_name(input_vector_name,dimension[,usage])`

With all versions of NONMEM since NONMEM VI, user-supplied functions are permitted in abbreviated code with reserved names such as FUNCA, FUNCB, ..., and argument vectors with reserved names such as VECTRA, VECTRB, ... With NONMEM 7.4, extended reserved names are recognized. These are FUNCxy and FUNCxyz, where each of x, y, z stands for an alphabetic character A-Z, e.g., FUNCAB or FUNCABC. Similar extended reserved names for vectors are also recognized: e.g, VECTRAB or VECTRABC. Reserved argument vectors and arrays have dimension 9 and each reserved function may appear in abbreviated code at most 9 times.

(See **Abbreviated Function**).

In NONMEM 7.4 the $ABBR FUNCTION option allows user-defined function names and user-defined argument vector names. The dimensions of the argument vector and the maximum number of times a given function name may appear in abbreviated code is user-specified.

A user-defined function may be declared as follows:

```
$ABBR FUNCTION function_name(input_vector_name,dimension,usage)
```

function_name

is the name of the function.

input_vector_name

is the name of an input vector that may be used to pass arguments to the function.

dimension

specifies how many input arguments function_name will use, and defines input_vector_name as a vector with this length. "Dimension" is a property of both the function and of the input vector.

usage

is the maximum number of times the function may appear in the abbreviated code, that is, the maximum number of occurances of function_name. It is not an error if

there are fewer occurances. If usage is omitted, NMTRAN will supply the exact number. If usage is coded, NMTRAN will generate an error message if function_name appears in abbreviated code more than "usage" number of times.

A vector and its length may be declared independently of a function,

```
$ABBR VECTOR input_vector_name (dimension)
```

input_vector_name
is the name of an input vector.

dimension
The length of the vector. The dimension of a vector should be no less than the dimension of all the functions which which it is used.

Example:

```
$ABBR FUNCTION BIVARIATE(VBI,5,3)
```

A vector VBI is defined of length 5. There is a function called BIVARIATE. When BIVARIATE is used, the value 5 is passed to it as argument NDIM. BIVARIATE uses 5 elements from the input vector. Function BIVARIATE may appear in abbreviated code at most 3 times. BIVARIATE should be present in a source code file listed in the $SUBROUTINES record:

```
$SUBROUTINES OTHER=filename
```

Instructions for coding both reserved functions and user_defined functions are in **Abbreviated Function**.

Here is an example of abbreviated code that uses BVI and BIVARIATE:

```
$PK ...
VBI(1)=RHO
VBI(2)=5
VBI(3)=6
VBI(4)=1 ;***0 = Upper tail as in Drezner & Wesolowsky
         ; 1 = Bottom tail***;
VBI(5)=1 ;***0 = 3 pt approximation
          ; 1 = 5 point approximation***;
BV=BIVARIATE(VBI)
```

There is a fully worked out example.
(See **bivariate function**).
Files are in the examples directory: bivariate.ctl, bivariate.csv, bivariate.f90.

By default, there may be up to 100 different user-defined functions, which includes functions with reserved names and user-defined names. There may also be up to 100 different vectors of input arguments, which includes vectors with reserved names and user-defined names. Constants NFUNCX and NVECX in SIZES may be used to change these numbers.

It is preferable that a unique vector should be associated each function to assure that each vector-function pair are set up with comparable dimensions. This is not necessary. Any user-defined vector or function may be used with any reserved vector or reserved function, subject to the same restriction. For example:

```
$ABBR FUNCTION BIVARIATE(VBI,5)
$ABBR FUNCTION BIVARIATEQ(VQI,10)
```

In abbreviated code, any of the following are permitted:

```
BVI=BIVARIATE(VBI)
```

```
BVQ=BIVARIATE(VQI)
BVQI=BIVARIATEQ(VBI)
BVQQ=BIVARIATEQ(VQI)
```

NM-TRAN gives a warning:

```
(WARNING  133) DIMENSION OF VECTOR IS LESS THAN
               WHAT IS EXPECTED BY FUNCTION
BVQI=BIVARIATEQ(VBI(001),FNC002_1(1,001),FNC002_2(1,1,001),10)
```

This is because vector VBI has dimension 5 and function BIVARIATEQ was declared with dimension 10. Note that the results will be correct if BIVARIATEQ does not use more than 5 positions in the argument vector.

A vector and its length may be declared independently of a function, and vice versa. The asterisk is used as a placeholder, e.g.,

```
$ABBR FUNCTION BIVARIATE(*,5)
; when BIVARIATE is called, NDIM will be 5
$ABBR FUNCTION BIVARIATEQ(*,10)
; when BIVARIATEQ is called, NDIM will be 10
$ABBR VECTOR VQI(15)
```

The $ABBREV FUNCTION record may be used to override the default settings for any of the reserved function or vectors. For example,

```
$ABBR FUNCTION FUNCA(VECTRA,25,5)
```

VECTRA will be defined with length 25, not 9 as for a reserved function. The code for the function should have NDIM as an argument:

```
FUNCTION VECTRA(X,X1,X2,NDIM)
```

NDIM should be used instead of 9 for vector and array dimensions, as shown in the **Abbreviated Function** help item.

REFERENCES: Guide IV, section III.B.7 , IV
REFERENCES: Guide Introduction_7

$AES

MEANING: Marks the beginning of abbreviated code for the AES routine
CONTEXT: NM-TRAN Control Record

USAGE:
`$AES`
abbreviated code

DISCUSSION:
The $AES record is used to compute algebraic expressions which can be regarded as specifying equilibrium kinetics. It is used with PREDPP's ADVAN9 and ADVAN15 and ADVAN17.
(See **$AESINITIAL**).
General rules for abbreviated code are documented elsewhere
(See **abbreviated code**).
Specific rules follow.

ASSIGNMENT AND CONDITIONAL STATEMENTS

Left-hand quantities in assignment statements:

`E(ncm1+1)`, `E(ncm1+2)`, ... (Required. Expressions which, when set to 0, constitute the system of algebraic expressions describing the equilibrium kinetics. ncm1 is the number of nonequilibrium compartments.) The indices "(...)" may be omitted, in which case NM-TRAN will supply them according to the order in which the expressions appear. Indices are required when the expressions are defined conditionally (i.e., using an IF statement).

AES-defined (i.e., PRED-defined) items.

Right-hand quantities in assignment statement and in conditions:

`A(1)`, `A(2)`, ... (Current compartment amounts; may be random variables.)

`P(1)`, `P(2)`, ... (Post-translation basic PK parameters; may be random variables.)

PK-defined items (Implicit basic PK parameters; may be random variables.)

`T` (Time; may be random variable. `T` takes values continuously over an integration interval.)

AES-defined variables that appeared earlier as left-hand quantities in $AES, and similarly from the $AESINITIAL record. (Caution: AESINITIAL-defined variables that depend on compartment amounts will depend on the initial values of these compartment amounts, not the current values.)

Data item labels specified on the $INPUT statement.

`THETA(n)`.

Global Variables in Modules

Certain variables in FORTRAN modules can be used.
(See **Variables_in_Modules**)
The following are of particular interest.

DOSTIM
DOSTIM is the time of a lagged dose or additional dose to which the system is being advanced. Abbreviated code in $AES may test DOSTIM. It may use DOSTIM on the right, unless DOSTIM is a random variable. However, it may be used on the right in a $PK block to define a random variable which may in turn be used on the right in the $AES block.

DOSREC
DOSREC is the dose record corresponding to the dose entering at DOSTIM. Abbreviated code in $AES may test items in DOSREC in a logical condition, and DOSREC may always be used on the right.

ISFINL
During simulation or a copying pass, and during the advance to a particular time (event or non-event time), ISFINL=1 at a final call to AES at that time. Otherwise, ISFINL=0.

Forbidden Variable Names:

```
IR DA DP DT ETA(n) EPS(n) ERR(n)
```

RECORD ORDER:

Follows $SUBROUTINES $INPUT $MODEL $PK
Follows $AESINITIAL

(See **aes**, **advan9_15**, **advan9_17**).

REFERENCES: Guide IV, section V.C.9
REFERENCES: Guide VI, section VI.E

$AESINITIAL

MEANING: Marks the beginning of abbreviated code for the AES routine
CONTEXT: NM-TRAN Control Record

USAGE:
`$AESINITIAL`
abbreviated code

DISCUSSION:
The $AESINITIAL record is used to compute the amounts in the equilibrium compartments at the beginning of an integration interval. It is used with PREDPP's ADVAN9, ADVAN15, and ADVAN17. May also be coded `$AES0`.
(See **$AES**).
General rules for abbreviated code are documented elsewhere
(See **abbreviated code**).
Specific rules follow.

ASSIGNMENT AND CONDITIONAL STATEMENTS

Left-hand quantities in assignment statements:

`A(ncm1+1)`, `A(ncm1+2)`, ... (Required. The amounts in the equilibrium compartments at the beginning of the integration interval. ncm1 is the number of nonequilibrium compartments.)

`INIT` (Initialization flag.)
`INIT=0`: The A(n) are approximate.
`INIT=1`: The A(n) are exact (the default).

AESINITIAL-defined (i.e., PRED-defined) items.

Right-hand quantities in assignment statement and in conditions:

`A(1)`, `A(2)`, ... (Current compartment amounts; may be random variables.)

`P(1)`, `P(2)`, ... (Post-translation basic PK parameters; may be random variables.)

PK-defined items (Implicit basic PK parameters; may be random variables.)

`T` (Time at the beginning of the integration interval; may be random variables.)

AESINITIAL-defined items that appeared earlier as left-hand quantities.

Data item labels specified on the $INPUT statement.

`THETA(n)`.

Global Variables in Modules
Certain variables in FORTRAN Modules can be used.
(See **Variables_in_modules**)

Forbidden Variable Names:

```
  IR DA DP DT E(n) ETA(n) EPS(n) ERR(n)
```

PSEUDO ASSIGNMENT STATEMENTS

```
COMRES=-1
```

`CALLFL=-1`: Call ADVAN and AES with every event record (default).
`CALLFL=1`: Call ADVAN and AES once per individual record.

(CALLFL may be used only when the TIME data item is not defined.) The pseudo assignments statements may be enclosed in parentheses. If both are present within the same set of parentheses, separate them with a semicolon. Within parentheses, a calling protocol phrase may be used instead of CALLFL, and either upper or lower case may be used. E.g.,

```
$AESINIT (ONCE PER IR)      ; same as CALLFL=1
$AESINIT (EVERY EVENT)      ; same as CALLFL=-1 (default)
```

(See **calling protocol**).

RECORD ORDER:

Follows $SUBROUTINES $INPUT $MODEL $PK
Precedes $AES

REFERENCES: Guide IV, section V.C.8
REFERENCES: Guide VI, section VI.E

$ANNEAL

MEANING: Sets starting diagonal Omega values to facilitate EM search methods
CONTEXT: NM-TRAN Control Record

USAGE:
`$ANNEAL` number-list1:value1 number-list2:value2 ...

SAMPLE:
```
$ANNEAL 1-3,5:0.3 6,7:1.0
```

DISCUSSION:

Sets starting diagonal Omega values for purposes of simulated annealing by NONMEM subroutine CONSTRAINT.

In the above example, initial values of OMEGA(1,1), OMEGA(2,2), OMEGA(3,3), and OMEGA(5,5) are set to 0.3, while initial OMEGA(6,6) and OMEGA(7,7) are set to 1.0.

A number-list may contain a single integer, a range of integers (with -), or a series of integers and ranges separated by comma. Required.

A value may be any numeric value. Optional; default is 0.

When $EST CONSTRAIN>=4, an algorithm in subroutine CONSTRAINT will initially set the omegas to these values, and then shrink these OMEGA values more and more with each iteration, and eventually shrinks the OMEGA's to 0, the intended target value for that Omega. This is a technique that may be used especially with SAEM, to provide an annealing method for moving thetas that have 0 omega values associated with them. The default is the use of gradient methods, which are good for problems starting near the solution, whereas the annealing method is more suitable for problems starting far from the solution.

Subroutine CONSTRAINT obtains values entered via $ANNEAL record in the array OMEGANNL. Any value that is set to 0 in $ANNEAL is given the default value of .3.

This record is optional. If omitted, the starting values of Omega are those specified in the $OMEGA record.

NONMEM's default CONSTRAINT.f90 is identical to Subroutine source/CONSTRAINT.f90.

See **INTRODUCTION TO NONMEM 7, $ANNEAL to facilitate EM search methods**

REFERENCES: Guide Introduction_7

$BIND

MEANING: Define data values used by $PK, $DES, and $AES
CONTEXT: NM-TRAN Control Record

USAGE:
`$BIND` [$value_1$] [$value_2$] ...

DISCUSSION:
$BIND is optional. It may be used when $PK abbreviated code is present *and* this code requests that the PK subroutine be called at additional or lagged dose times (CALLFL=-2). It is used to override the default values of user-defined variables used in the code when the PK routine is called at these particular dose times. It is also used to override the default values of user-defined variables used in $DES and $AES abbreviated code during an advance to an additional or lagged dose time.

$BIND has no effect when PK is called at a model event time (MTIME).

Let t be a time at which an additional or lagged dose enters the system. If t1 is the greatest event time not exceeding the time t, then the "last event record" is the last event record with event time t1, and the "next event record" is the event record following the last event record.
(The term "last" is similar to the word "previous" in this context.)
The event time on the next event record will exceed time t.

OPTIONS:

The positions of the values correspond to the positions of data items in the $INPUT record in a 1-to-1 manner. Each value is one of:

DOSE (Use the value from the dose record.)
NEXT (Use the value from the next event record.)
LAST (Use the value from the last event record.)
SKIP (Ignore this data item.)
DROP (Ignore this data item.)
\- (Default.)

For user (concomitant) data items, the default is NEXT but any of DOSE, NEXT, LAST may be specified.

For the PREDPP data item TIME, the default is NEXT. Only - or NEXT may be specified.

For all other PREDPP or NONMEM data items, the default is DOSE. Only - or DOSE may be specified.

Option DROP in the $BIND record is optional. It is ignored by NM-TRAN, but helps the user maintain a 1-to-1 relationship between positions in $BIND and positions in $INPUT.

A $BIND record with all defaults specified, such as
```
$BIND - - - - - - -
```
has the same effect as if no $BIND record were present.

EXAMPLE:

```
$INPUT ID TIME DATE=DROP AMT DV WGT  PREP  X   HGT
$BIND   -   -    DROP     -   - NEXT DOSE LAST  -
```

ID, AMT, DV have the values from the initiating dose record.
TIME has the value from the next event record.

WGT and HGT have values from the next event record.
PREP has the value from the initiating dose record.
X has the value from the last event record. This record will be the dose record if there is no other event record between the dose record and the next event record.

$INPUT and $BIND records can be interleaved to help maintain a visual relationship. The above example could have been coded:

```
$INPUT ID TIME DATE=DROP
$BIND   -    -    DROP
$INPUT AMT DV  WGT PREP X    HGT
$BIND  -    -  NEXT DOSE LAST  -
```

$BIND may not specify a position beyond the last position defined via $INPUT. It may specify fewer positions, in which case defaults apply to the remaining data items.

Changes to $BIND, like changes to $INPUT, cause changes to generated code. Thus, an existing NONMEM executable cannot be re-used when the $BIND and/or $INPUT records are changed.

The $BIND record only applies under the following circumstances:

(1) There exists a dose that is either one or both of the two following types of doses:
   a) An additional dose, subsequent to the initiating dose (ADDL>0, II>0 in the dose event record). Such a dose enters the system at the "additional dose time."
   b) A lagged dose. (with the corresponding dose event record, PK specifies ALAGi>0, where i is the index of the dose compartment). Such a dose enters the system at the "lagged dose time."

(2) The PK subroutine computes parameters that depend on values in the data record which are not constant for the individual, i.e., parameters depend on time-varying data items.

(3) The PK subroutine is called when the dose enters the system.
That is, $PK contains the pseudo-statement CALLFL=-2, requesting that the PK routine be called to compute values of the PK parameters at additional and lagged dose times. When $PK does not contain this pseudo-statement, there is no such call, and at an additional or lagged dose time, the PK parameters have those values computed with the next event record.

When the $PK code is implemented with an event record, the variable DOSTIM is 0. When it is implemented at an additional or lagged dose time, the value of this variable is the time in question. By default, data items used in abbreviated code have values from either the initiating dose record (DOSREC) or the next event record (EVTREC), according to this rule:

NONMEM data items and PREDPP data items (other than TIME) have values from the original dose event record (DOSREC). These are:

DV MDV ID L2 MRG_ RAW_ REPL_ EVID AMT RATE SS II CMT PCMT CALL CONT ADDL DATE DAT1 DAT2 DAT3

TIME and all user-defined data items have the values from the next event record (EVTREC). This default can be overridden using the $BIND record.

EXAMPLE:

The $BIND record is a convenience; it does nothing that cannot be accomplished with abbreviated code. Suppose X is a user-defined data item. The following three fragments of code create a variable XB with the same values at calls with additional and lagged doses that X would have if there were a $BIND record specifying the following for X.

DOSE:

```
IF (DOSTIM.EQ.0) THEN
XB=X
ELSE
XB=DOSREC(X)
ENDIF
```

NEXT:

```
XB=X
```

LAST:

```
IF (DOSTIM.EQ.0) THEN
XB=X
ENDIF
```

In this case, when DOSTIM>0, XB retains its value from the previous call to PK.

(See **bind example**).

REFERENCES: Guide IV, section V.C.2 , V.C.5

$CHAIN

MEANING: Supplies initial estimates for an entire problem
CONTEXT: NM-TRAN Control Record

USAGE:

```
$CHAIN [FILE=filename]
       [FORMAT=s] [ORDER=xxxf]
       [NOTITLE=[0|1]][NOLABEL=[0|1]]
       [ISAMPLE=n][NSAMPLE=n]
       [SEED=n]
       [CLOCKSEED=[0|1]]
       [SELECT=n]
       [RANMETHOD=[n|S|m] ]
       [CTYPE=[0|1|2|3|4]]
       [DF=n]
       [DFS=n]
       [IACCEPT=x]
       [TBLN=n]
```

SAMPLE:

```
$CHAIN FILE=example1_previous.txt NSAMPLE=0 ISAMPLE=-1000000000
```

DISCUSSION:
Option METHOD=CHAIN of the $ESTIMATION record will only set thetas, omegas, and sigmas for initial values of the estimation process. Its scope is therefore limited in that it will not impact the parameters used in simulating data for the Simulation step. To introduce initial THETAs omegas and sigmas that will cover the entire scope of a given problem, use the $CHAIN record.

The options have the same meanings as for the $ESTIMATION record. Setting SEED or CLOCKSEED or RANMETHOD in a $CHAIN record does not propagate to $EST METHOD=CHAIN or any other $EST record.
(See **$ESTIMATION_record_options**).
(See **$ESTIMATION_record**).

REFERENCES: Guide Introduction_7

$CONTR

MEANING: Defines values for certain user-supplied routines
CONTEXT: NM-TRAN Control Record

USAGE:
`$CONTR DATA=([`*label*$_1$`|0] [`*label*$_2$`|0] [`*label*$_3$`|0])`

The data item with the Jth label (J=1,2,3) and from the Ith observation record of an individual record is available in DATA(I,J). If 0 is used instead of a label, then a zero appears in DATA(I,J).

SAMPLE:
```
$CONTR    DATA=(0,TYPE)
```

DISCUSSION:
Optional. Used only with user-supplied routines such as MIX and CONTR and CCONTR that use data items stored in the DATA array. This record gives labels (or synonyms) defined in the $INPUT record of one to three types of data items to be made available to the subroutine(s) in the DATA array. These routines are called with individual records. An array DATA is available in NONMEM module ROCM_REAL and changes value with each individual record.

With the above sample $CONTR record, the following code might be present in a double precision MIX routine. The code loops through the observation records of the NREC'th individual record. For each of the NOBS observation records, the local variable TYPE is given the value of the TYPE data item from that data record. The 0 in the sample is a place-holder which causes the first column in the DATA array to be skipped. The value of TYPE for the Ith observation record is therefore available in DATA(I,2). The DATA array is found in ROCM_REAL. NO is a constant giving the maximum number of observations per individual record. NOBS is the number of observations in the current individual record. (See **sizes**).

```
      USE SIZES, ONLY: NO,DPSIZE
      USE ROCM_REAL, ONLY: DATA=>RDATA
      USE ROCM_INT, ONLY: NOBS=>NOBSIND2
      ...
      INTEGER I
      REAL(KIND=DPSIZE) :: TYPE
      ...
      DO 100 I=1,NOBS
      TYPE=DATA(I,2)
      ...
  100 CONTINUE
      ...
```

REFERENCES: Guide IV, section III.B.4

$COVARIANCE,$COVR

MEANING: Instructions for NONMEM Covariance Step
CONTEXT: NM-TRAN Control Record

USAGE:

```
$COVARIANCE [SPECIAL] [MATRIX=] [PRINT=[E][R][S]
            [COMPRESS]
            [SLOW|NOSLOW|FAST]
            [TOL=n] [ATOL=n]
            [SIGL=n] [SIGLO=n]
            [NOFCOV]
            [PARAFILE=[filename|ON|OFF] [PARAFPRINT=n]
            [THBND=n]
            [SIRSAMPLE=[numberlist] [SIRNITER=n] [SIRCENTER=n]
            [IACCEPT=x] [IACCEPTL=x]
            [SIRDF=n] [RANMETHOD=[n|S|m|P] ]
            [SIRPRINT=n] [FILE=filename] [FORMAT=s]
            [SIRTHBND=n]
            [SIRMINWT=x] [SIRMAXWT=x] [SIR_CAPCORR=x]
            [SIRSEED=n] [SIRCLOCKSEED=[0|1]]
            [SIRPARAFILE=[filename|ON|OFF] [SIRPARAFPRINT=n]
            [PRECOND=n] [PRECONDS=TOS] [PFCOND=n] [PRETYPE=n]
            [FPOSDEF=n]
            [CHOLROFF=n] [KNUTHSUMOFF=n]
            [POSDEF=n]
            [RESUME]
            [CONDITIONAL|UNCONDITIONAL]
            [OMITTED]
```

SAMPLE:

```
$COVARIANCE
```

DISCUSSION:

Optional. Requests that the NONMEM Covariance Step be implemented. This step outputs: standard errors, covariance matrix, inverse covariance matrix, and the correlation form of the covariance matrix. May also be coded `$COVR`.

If $COV is specified, then for IMP, IMPMAP, and ITS methods, standard error information will be supplied for every $EST statement. Standard error information for the classical methods (METHOD=0, METHOD=1) will be given only if they are the last estimation method, and only if NOFCOV is not specified.

OPTIONS:

`SPECIAL`

The special computation will be used in the Covariance Step with a recursive PRED subroutine. A recursive PRED subroutine is such that, with single-subject data, the PRED computation with a data record depends on information passed to it with *any* of the previous data records. This is the default when PREDPP is used.

`MATRIX=c`

Specifies that the covariance matrix will be different from the default ($R^{-1}SR^{-1}$). MATRIX=R requests that 2 times the inverse R matrix be used. MATRIX=S

requests that 4 times the inverse S matrix be used. (R and S are two matrices from statistical theory, the Hessian and Cross-Product Gradient matrices, respectively.) With MATRIX=R the standard errors will be more consistent with other nonlinear regression software implementations. MATRIX=R should not be used with option SPECIAL.

MATRIX has no relevance to the Covariance step results for a BAYES method.

`PRINT=[E][R][S]`

Additional outputs will be printed besides the defaults. E: print the eigenvalues of the correlation matrix. R: print the matrix .5*R. S: print the matrix .25*S. PRINT=R (or S) is not needed with MATRIX=R (or S).

`COMPRESS`

Covariance Step arrays are printed in compressed format, even if their size is such that NONMEM would normally print them in the usual format.

`SLOW`

Requests a slower method of computation. Required when either a mixture model was used along with `CENTERING` on the $ESTIMATION record, or `NUMERICAL` was used on the $ESTIMATION record. If not present, the option will be automatically supplied in these two cases.

`NOSLOW`

Requests a faster method of computation. This is the default (but see `SLOW`).

`FAST` (NM74)

This is equivalent to FAST for the $EST record. record. If $EST FAST is set, then $COV will be set to FAST, unless SLOW or NOSLOW is specified.

`TOL=`*n* (NM72)

Tolerance. Used only with General Nonlinear (differential equation solver) ADVAN's. Sets NRD=TOL. By default, the value set on $SUBROUTINES record (or $TOL or TOL subroutine) is used. If TOL is coded on $COVARIANCE, it overides the default. With NONMEM 74, this feature is deprecated. A user-supplied TOL subroutine should test NM_STEP and set NRD accordingly.

TOL has no relevance to the Covariance step results for a BAYES method.

`ATOL=`*n* (NM72)

Absolute tolerance. Used only with ADVAN9, ADVAN13, ADVAN14, ADVAN15, ADVAN16, ADVAN17, ADVAN18, for which TOL is a relative tolerance. Sets ANRD=ATOL. The default is 12 (that is, accuracy is 10**(-12)). Usually the problem runs quickly when using this setting. On occasion, however, you may want to reduce ATOL (usually set it equal to that of TOL), and improve speeds of up to 3 to 4 fold.

By default, the value set on $SUBROUTINES record (or $TOL or TOL subroutine) is used. If ATOL is coded on $ESTIMATION, it overrides the default for that step. If ATOL is coded on $COVARIANCE, it overrides $ESTIMATION and/or the default for that step. With NONMEM 74, this feature is deprecated. A user-supplied TOL subroutine should test NM_STEP and set ANRD accordingly.

`SIGL=`*n* `SIGLO=`*n* (NM72)

These options may be used to specify different values for the Covariance step. The step size for evaluating the R matrix (central difference second derivative) is set to SIGL/4. If only the S matrix is evaluated (central difference first derivative), the step size is set to SIGL/3. SIGLO is the precision to which individual etas are optimized. Classical NONMEM methods in particular often require a different significant digits

level of evaluation (usually more stringent) during the Covariance step than during Estimation Step. For example, during the Estimation step, NSIG=2, SIGL=6, TOL=6 may be sufficient, but during the Covariance step, you may need SIGL=12 TOL=12 to avoid positive definiteness issues.
By default, values specified on the $ESTIMATION record are used.
(See **$ESTIMATION**).
SIGL and SIGLO have no relevance to the Covariance step results for a BAYES method.

`NOFCOV` (NM72)
No $COV step for any classical estimation steps. This would be useful if you wanted EM estimation analyses performed, and a final FOCE analysis performed, but did not want the program to spend time on standard error assessments for FOCE, which can take a long time relative to the other methods.

`PARAFILE`=filename
Name of the "parallel file" (the parallelization profile) that controls parallelization (distributed computing). Default file name if not specified: parallel.pnm or parafile name specified on nmfe command.
`PARAFILE=ON` turns on parallelization for this $COVARIANCE record.
`PARAFILE=OFF` turns off parallelization for this $COVARIANCE record.

`PARAFPRINT`=*n* (NM74)
The print iteration intervals to the parallelization log file can be controlled by this option during parallelization of the $COV step. See also $ESTIMATION record and nmfe7 command. Default is `PARAFPRINT=1`.

`THBND`=*n* (NM74)
If THBND=1, any theta boundaries specified in the $THETA record causes NONMEM to impose a non-linear transformation of the theta parameters so that the transformed parameters may vary from -infinity to infinity. It does this with logistic transformations. This is suitable during the estimation step, but may be desirable have this off (THBND=0) for covariance assessment, and assess partial derivatives of the objective function with respect to the thetas themselves, or some linear transformation of these thetas. By default THBND=1, in keeping with the behavior of earlier NONMEM versions, which effectively has THBND=1. Usually boundaries that are fairly wide will not impact how the variance-covariance is assessed, such as when a lower bound of 0 is given, but if you have very narrow boundaries set, then this can impact the assessment of the variance-covariance of the estimates considerably, and you may wish to set THBND=0. If no lower or upper bounds are given to thetas in $THETA record, this option has no impact.

`SIRSAMPLE`=[numberlist] (NM74)
Option SIRSAMPLE requests the Sampling-Importance-Resampling algorithm (SIR)
(See Guide Introduction_7, "Importance Sampling of the Variance-Covariance of the Parameter Estimates").
By default SIRSAMPLE=-1, so SIR process does not occur. SIRSAMPLE should be set between 300 and 10000, to indicate the number of random samples to be generated for each of the SIRNITER iterations. This will produce SIRSAMPLE importance samples, each of which will be placed in the raw output file. As of nm75, SIRSAMPLE may contain a list of integers, one for each of the SRNITER iterations (use double quotes around the list): SIRSAMPLE="1000,200,500" If there are fewer than SIRNITER values, the last value in the list will be used as the SIRSAMPLE

value for the remaining iterations. Utility programs table_quant (frequency and quantile sorting) and table_resample (resampling) may be used to analyze $COV Sampling-Importance-Resampling data.

`SIRNITER=`*n* (NM74)
The number of times SIR sampling should occur. Default is 1.

`SIRCENTER=`*n* (NM74)
Where the sampling (proposal) density is to be centered. On the first iteration, the mean of the sampling density is at the estimate. On subsequent iterations, the mean of the sampling density is at the estimate (SIRCENTER=0) or at the mean of the (transformed) samples of the previous iteration (SIRSAMPLE=1). Default is 0.

`SIR_CAPCORR=`*x* (NM75)
Limit the correlation of omegas and sigmas generated form the proposal density to have |correlation| of SIR_CAPCORR or less. This is similar to PSN's cap_correlation option.

`SIRMINWT=`*x* (NM75)

`SIRMAXWT=`*x* (NM75)
Limit the weight range that a sample may relative to the proposal density, to avoid extreme values. Default values are SIRMINWT=0.001, SIRMAXWT=1000.0

`SIRSEED=`*n* (NM75)
Starting seed for SIR analysis (default=11456).

`SIRCLOCKSEED=[0|1]` (NM75)
If SIRCLOCKSEED=1 (default is 0), actual starting seed will be 10000*(seconds after midnight)+SEED. This allows a control stream to produce different stochastic results for automated replications, without the need to modify the seed value in the control stream file in each replication.

`IACCEPT=`*x* (NM74)
The acceptance ratio acts similarly to importance sampling in EM analysis. Default is 1.

`IACCEPTL=`*x* (NM74)
IACCEPTL=0 (NM74) Default is 1. The IACCEPTL option performs the same as listed for IACCEPTL in "Monte Carlo Importance Sampling EM". Default is 0.

`SIRDF=`*n* (NM74)
The proposal density is to be a t distribution with n degrees of freedom. Default is 0, a normal density.

`RANMETHOD=`[n|S|m|P] (NM74)
See the corresponding option of the $ESTIMATION record for importance sampling in EM analysis.

`SIRPRINT=`*n* (NM74)
Set the console print iterations interval. This does not impact the iterations listed in the raw output file. Default SIRPRINT=0.

`FILE=`filename (NM74)
By default, the raw output file is whatever was listed in the $EST step, or root.ext, where root is the root name of the control stream file. You can re-direct SIR sample listings to an alternative file with this option.

`FORMAT=`*s* (NM75)
By default, the format for the raw output file and all additional output files is

whatever FORMAT was specified for the $EST step. You can change its format with the FORMAT option. *s* defines the delimiter [,|s(pace)|t(ab)] followed by a Fortran format specification. The default is s1PE12.5. For more details, see the format help item:
(See **format**).
See **INTRODUCTION TO NONMEM 7, FORMAT=s1PE11.4**

`SIRTHBND=`*n* (NM74)

As with the deterministic covariance assessment step, by default the transformed parameters are sampled, so that no sample is below the $THETA lower bound specification, and no higher than the $THETA upper bound specification. To allow a boundariless search in the original theta domain, set SIRTHBND=1. You should also set THBND=1, so that the determinstc covariance matrix used as the proposal density is also not hindered or contorted by the boundaries. Default is the value of THBND, which in turn is 0 by default.

`SIRPARAFILE=`filename

Name of the "parallel file" (the parallelization profile) that controls parallelization during SIR sampling (distributed computing). Default file name if not specified: parallel.pnm or parafile name specified on nmfe command.
`SIRPARAFILE=ON` turns on parallelization for this $COVARIANCE record during SIR sampling.
`SIRPARAFILE=OFF` turns off parallelization for this $COVARIANCE record during SIR sampling.

`SIRPARAFPRINT=`*n* (NM74)

The print iteration intervals to the parallelization log file can be controlled by this option during parallelization of the SIR sampling portion of the $COV step. See also $ESTIMATION record and nmfe7 command. Default is `SIRPARAFPRINT=1`.

`PRECOND=`*n* (NM74)

Options PRECOND through PRETYPE affect the preconditioning of the R Matrix.
(See Guide Introduction_7, "Preconditioning the R Matrix to Improve Precision and Success Rate of $COV Step").
By default, PRECOND (Preconditioning cycles") is 0, and no preconditioning of the R matrix is performed. When PRECOND=n, then up to n preconditioning cycles are performed. This is used in combination with the PFCOND setting.

`PRECONDS=TOS` (NM74)

By default, if preconditioning is performed, it is done on Thetas (T), Omegas (O), and Sigmas(S). Specify PRECONDS (Preconditioning types)=T to do only thetas, PRECONDS=TO to do only thetas and omegas, etc.

`PFCON=`*n* (NM74)

PFCOND means 'forced' precondition cycles. Preconditioning occurs exactly PFCOND times, without testing if the R matrix is positive definite or not on each preconditioning cycle. On the remaining PRECOND-PFCOND cycles, the R matrix is tested for positive definiteness, and upon success, will terminate the preconditioning cycles. Default is PFCOND=0.

`PRETYPE=`*n* (NM74)

By default PRETYPE(preconditioning type)=0 and the R matrix corrector is V*square_root(eigenvalue). If you set PRETYPE=1, then the corrector is V*square_root(eigenvalue)*Vtranspose. If you set PRETYPE=2, then the corrector

is the correlation version of PRETYPE=1.

FPOSDEF=*n* (NM74)

By default FPOSDEF(forced positive definite)=0. If FPOSDEF=1, then if the Rmatrix is not positive definite, it will be forced positive definite. If PRECOND>0, this will occur after the PRECONDth try.

CHOLROFF=*n* (NM74)

If CHOLROFF is set to 1, then one part of the R matrix evaluation (before inversion) will be evaluated in the manner of earlier versions of NONMEM. This is strictly for comparison with earlier versions for diagnostic purposes. If CHOLROFF=0, then a part of the R matrix evaluation will undergo cholesky decomposition, and if not positive definite, only then will it undergo evaluation according to earlier versions of NONMEM. If CHOLROFF=2, then if cholesky decomposition fails, it will be slightly adjusted to be positive definite. The CHOLROFF=0 is the safest option, in that the Cholesky decomposition (if positive definite) provides more precision in evaluating the R matrix. The CHOLROFF=2 setting is useful if you wish to increase the rate of success in obtaining an R matrix (followed by its inverse) that could be suitable for SIR sampling. If you use CHOLROFF=2, then positive definiteness has been corrected before the inverse occurs, so POSDEF will have no additional effect. If you use POSDEF, then CHOLROFF should be 0 or 1. Default is 0.

KNUTHSUMOFF=*n* (NM74)

In NONMEM 7.4, the Knuth summing method is used to allow the most accurate summation of individual objective function values, even with large variations in values of the individual objective function. To turn this off, and allow a standard summation (not recommended except for comparison purposes from earlier versions), set KNUTHSUMOFF=1. If KNUTHSUMOFF was set in the $EST step, but not in the $COV step, the KNUTHSUMOFF value of the last $EST record will be used.
Default is 0.

POSDEF=*n* (NM75)

If POSDEF=1, then if the R matrix is not positive definite, it will be forced positive definite, by method of shifting the eigenvalues by an amount slightly greater than the most negative eigenvalue. If POSDEF=2, then R matrix is made positive definite by making the negative valued eigenvalues approximately 1/100 of the least positive eigenvalue. If POSDEF=3, then R matrix is made positive definite by using the absolute values of the eigenvalues POSDEF is 0 (no correction) by default for classical covariance step, and 3 for EM methods.
The preconditioning method is a more sophisticated, but time-consuming method of making the R matrix positive definite (See PRECOND).
Default is 0.

CONDITIONAL

The Covariance Step is implemented, but only when the Estimation Step terminates successfully (in this run or in a run continued via $MSFI). This is the default.

UNCONDITIONAL

The Covariance Step is implemented regardless of how the Estimation Step terminates (in this run or in a run continued via $MSFI).

RESUME (NM73)

If MSFO=msfile was specified in the Estimation Step for the FO/FOCE/Laplace method and analysis was interrupted during the Covariance Step, then the Covariance Step may be resumed where it was interrupted in a subsequent problem. Use

the $MSFI record to specify the MSFO file of the interrupted analysis, and the RESUME option of the $COV record:

```
$MSFI=msfile
  ...
$COV RESUME
```

OMITTED

The Covariance Step is not implemented.

EXAMPLE:

```
$COV UNCONDITIONAL TOL=10 SIGL=10 SIGLO=11
```

REFERENCES: Guide IV, section III.B.15
REFERENCES: Guide V, section 9.4.2 , 10.6

$DATA

MEANING: Describes the NM-TRAN data set
CONTEXT: NM-TRAN Control Record

USAGE:

```
$DATA [filename|*] [(format)] [IGNORE=c1] [NULL=c2]
      [IGNORE= (list) ...|ACCEPT= (list) ...]
      [PRED_IGNORE_DATA]
      [NOWIDE|WIDE] [CHECKOUT]
      [RECORDS=n1|RECORDS=label]
      [LRECL=n2] [NOREWIND|REWIND]
      [NOOPEN] [LAST20=n3] [TRANSLATE= (list) ]
      [BLANKOK]
      [MISDAT=r...]
      [REPL=n...]
```

SAMPLE:

```
$DATA        DATAFILE
```

DISCUSSION:

This record specifies the data set to be used. It is required with the first problem specification. It must precede any other NM-TRAN control record that refers to specific data item types. May also be coded `$INFILE`.

Optional with the second or subsequent problem specifications. If omitted, NONMEM re-uses the data set from the previous problem (which will include any modifications made via transgeneration, e.g., via use of NONMEM's PASS, or via simulation).

OPTIONS:

`filename`

Name of the file containing the data set. Must be the first option. If it contains commas, semicolons, or parentheses, then it must be surrounded by single quotes ' or double quotes ". Filename may also contain equal signs if it is enclosed in quotes. If the file is opened by NM-TRAN, filename may contain embedded spaces if it is enclosed in quotes, and may contain at most 80 characters. If the file is opened by NONMEM, the filename may not contain embedded spaces, and may contain at most 71 characters. If filename is the same as any option of the $DATA record, it must be enclosed in quotes.

* may be coded in a problem subsequent to the first. This has the same effect as omitting the $DATA record (NONMEM is told to re-use the previous data set), but allows the CHECKOUT option to be included. With *, no other option may be included.

`(format)`

FORTRAN format specification to be used to read the data. Format codes F, E, and X may be used, but not I. When a format is provided, the label DROP cannot be used on the $INPUT record and options WIDE and NULL may not be coded. If omitted, NM-TRAN will generate a suitable FORMAT specification.

`RECORDS=`n_1

The number of records to be read from the NM-TRAN data set. Comment records

are not counted. If NM-TRAN does not drop any records from its data set (see IGNORE list and ACCEPT list), then n_1 is also the number of records written to the NONMEM data set. If NM-TRAN drops records, then the total number of records written to the NONMEM data set is n_1 minus the number of dropped records.
With NONMEM 7.5, records may also be dropped using the PRED_IGNORE_DATA block of abbreviated code. The same total applies to these dropped record. See PRED_IGNORE_DATA, below.

If omitted, the records written to the NONMEM data set are all the records in the NM-TRAN data set up to the end-of-file (or up to a NONMEM FINISH record) minus the number of comment and dropped records. May also be coded `NRECORDS`, `RECS`, or `NRECS`.

If the option is coded as `RECORDS`=label, where label is a data item label, NM-TRAN understands the data records for the problem to start with the first data record of the NM-TRAN data set (at the place where the file is positioned before data records are read; see the NOREWIND option), and to include as well, those and only those subsequent contiguous data records having the same value of the data item as does the first record. It counts the total number of these data records, minus any comment or dropped records, and puts this number in the NONMEM control file.

In particular, the ID label may be used (or alternatively, the option may be coded `RECORDS=IR`, `RECORDS=INDREC`, or `RECORDS=INDIVIDUALRECORD`). If a label other than ID is used, the $INPUT record must precede the $DATA record. If the data are single-subject data, the ID data items used to determine the data records for the problem are those labeled ID (not .ID.).

If there is more than one problem specification with a $DATA record that includes an option of the form `RECORDS`=label, then either none of these $DATA records may also include a format specification, or all of them must include the same format specification. (See records=id).

`LRECL`=n_2

The number of characters in a logical record. Needed for certain operating systems (e.g., IBM/CMS).

`IGNORE`=c_1

[Note: The following two options, `IGNORE` and `ACCEPT` allow the user to drop records from the NM-TRAN data set prior to the run. With NONMEM 7.5, it is also possible to drop records from the NONMEM data set using abbreviated code, which is more flexible than the criteria that can be specified in the $DATA record.
See PRED_IGNORE_DATA, below.]

Specifies that any data record having the character c_1 in column 1 should be ignored, i.e., these records are not included in the NONMEM data set. This allows comment records to be included in the NM-TRAN data set. In general, records having the character c_1 in column 1 will be called "comment records".
Also permitted: `IGNORE='c'` or `IGNORE="c"`, where `c` may be any character except space. `IGNORE=#` is the default. That is, in the absence of IGNORE option, any record whose first character is # is treated as a comment record.

`IGNORE=@` signifies that any data record having an alphabetic character or @ as its first non-blank character (not just in column 1) should be ignored. Alphabetic characters are the letters A-Z and a-z. This permits a table file having header lines to be used as an NM-TRAN data set.

`IGNORE=(list)`

"List" is a list of one or more data item labels, with logical operators and values, of the form "label=value", "label.EQ.value", "label.NE.value", "label.GT.value", "label.GE.value", "label.LT.value", and "label.LE.value". (Fortran 90 logical operators such as '==' '/=' '<' '<=' '>' '>=' " may also be used.) Thus, the following are identical: "label=value","label==value","label.EQ.value". With NONMEM 7.3, "label.NEN.value" and "label.EQN.value" are permitted. (There is no Fortran 90 operator for this comparison.) If the logical operator is omitted, the default is "=". With each data record, the value of the data item with the given label and the value in the list are compared according to the logical operator, and if result is "true", the record is ignored, i.e. it is not included in the NONMEM data set (see example below). Such records are called "dropped records". With "=", "==", '/=', ".EQ." and ".NE.", the value in the data record and the value in the list are compared as character strings. Otherwise, they are converted to numeric and compared numerically. (This is the case with .NEN. and .EQN.) This comparison is made prior to time translation. Hence, the TIME item cannot be compared numerically if it contains non-numeric characters such as ":".

Note: if the data file is a table file from a previous NONMEM run, values that had been integers (0,1,..) in the original data file will be real values (0.0000E+00, 1.0000E+00, ...) in the table file. A comparison for equality or inequality should now be for the real value. E.g.

```
IGNORE=(OCC==1.0000E+00).
```

With NONMEM 7 the default format for the table file is PE11.4, as in the examples above. The FORMAT option of $TABLE may be used to change this and the values used in a subsequent IGNORE will have to be changed accordingly.

The .NEN. and .EQN. operators that are described above will always work.

A data item label along with a logical operator and value is called a condition. A list may contain several conditions; these should be separated by commas, and the list should be enclosed in parentheses. Up to 100 different conditions altogether can be specified. IGNORE=(list) may be used with IGNORE=c, where c is a character. Multiple IGNORE options with different lists may be used. A list may span one or more NM-TRAN records. The use of "=" after IGNORE is optional, but parentheses are required with this form of IGNORE. Values may be alphabetic or numeric, and may optionally be surrounded by single quotes ' or double quotes ". Quotes are required if a value contains special characters such as =. However, a value may not contain spaces or commas. No format specification is permitted with this form of IGNORE.

A data item type may be dropped from the NONMEM data set by means of the DROP or SKIP synonym on the $INPUT record, after records are dropped due to a condition based on the data item type. E.g.,

```
$INPUT ... GEN=SKIP ...
$DATA file IGNORE=(GEN='M')
```

Records having GEN equal to 'M' will be dropped, and the GEN data item type will then be omitted from the NONMEM data set. A dropped data item may be any alphanumeric string (without a data item delimiter - a blank or a comma).

If there is more than one condition, then records satisfying at least one of these conditions will be dropped. In effect, the conditions for dropping a record are connected by the implied conjunction ".OR.". E.g.

```
IGNORE=(GEN.EQ.1,AGE.GT.60).
```

Records having GEN equal to 1 or AGE greater than 60 are dropped. All others are accepted.

`ACCEPT=` (list)

The ACCEPT list option is identical to the IGNORE list option, except that it specifies conditions for acceptance of records. An ACCEPT list cannot be used together with an IGNORE list. However, it may be used with IGNORE=c, where c is a character.

E.g.

```
ACCEPT=(GEN.EQ.1,AGE.GT.60).
```

Records having GEN equal to 1 or AGE greater than 60 are accepted. All others are dropped.

Suppose it is desired that records be dropped that satisfy the logical ".AND." of several conditions. This can be implemented by using an ACCEPT list with the negations of the conditions. For example, suppose that records to be ignored are those having GEN=1 .AND. AGE > 60. This may be done as follows:

```
ACCEPT=(GEN.NE.1,AGE.LE.60)
```

`PRED_IGNORE_DATA` (NM75)

Informs NONMEM that a PRED_IGNORE_DATA pass through the data set is required. If the abbreviated code uses variables PRED_IGNORE_DATA or PRED_IGNORE_DATA_TEST, this option is supplied. The explicit option is needed if the use of the PRED_IGNORE_DATA variables occurs in user-written or verbatim code, so that NMTRAN is unaware of it.

(See Guide Introduction_7 "Extension to $DATA IGNORE=st filtering")

`NULL=`c_2

Specifies that null data items in the NM-TRAN data set are to be replaced in the NONMEM data set by the character c_2. E.g., `NULL=0` or `NULL=.`.

Null data items consist of a single dot (.) or consecutive commas or consecutive tab characters. c_2 may be any character except space (" ") or semicolon (";").

Also permitted: `NULL='c'` or `NULL="c"`, where `c` may be any character.

If this option is omitted, NM-TRAN replaces each null with a space.

`NOWIDE`

Requests that NM-TRAN attempt to limit FDATA to 80-character records. Space between adjacent columns may be suppressed and multi-line records may be generated. This is the default.

`WIDE`

Requests that FDATA contain single-line records, and that at least one space separate columns. (Records in FDATA will never be wider than 300 characters.) With this option, there will be no FINISH (FIN) record in the NONMEM data set.

`NOREWIND` | `REWIND`

With the first problem specification in a control stream, the file is positioned at its initial point so that the first NM-TRAN data set in the file is used. The options REWIND and NOREWIND apply only with a $DATA record in a subsequent problem specification, and only when the file named on the record is the same as that specified for the previous problem. When the file named on the record is different from that specified for the previous problem, the file is (re)positioned at its initial point so that the first NM-TRAN data set in the file is used.

REWIND: Reposition the file at its intial point so that the first NM-TRAN data set in the file is re-used.

(Whether the file input to NONMEM itself will be repositioned depends on whether this file is FDATA or is one named in the $DATA record; see NONMEM Users Guide, Part IV for a complete explanation.)

NOREWIND: Leave the file at its current position so that the next NM-TRAN data set in the file is used. The $DATA record with the previous problem specification must have included the RECORDS option (or a FINISH record must have terminated the data set used in the previous problem), so that NM-TRAN did not read to a physical end-of-file. This is the default.

`CHECKOUT`

Requests that NONMEM implement the data checkout mode, in which the PRED routine is not called and predictions, residuals, weighted residuals and the objective function are not computed.

May also be coded `CHECKDATA`. No tasks other than $TABLE or $SCAT can be specified.

With NONMEM 7.5, an additional file, FDATA.csv is produced that outputs the contents of its input data file (typically FDATA) in a comma delimited file format, so you can check how NONMEM interprets the input data. If the REPL option of $DATA or the REPL_ data item is used, the replicated form of the data will appear in FDATA.csv.

`NOOPEN`

NM-TRAN will not open the named data file. This permits the data file to be created by one problem and used in a subsequent problem of the same run. May not be used with options `IGNORE`, `DROP`, or when data items `ID`, `MDV`, or `EVID` must be generated by NM-TRAN. With `NOOPEN`, a format specification is required. No day-time translation takes place.

`LAST20=`n_3

Override the LAST20 constant in resource/TRGLOBAL.f90 (default: 50).

One or two digit years > LAST20 are assumed to be in the 1900's,

One or two digit years <= LAST20 are assumed to be in the 2000's.

E.g,. suppose LAST20=50. Then two digit years are interpreted as follows:

00-50 = 2000-2050

51-99 = 1951-1999

LAST20=-1 can be used when two digit years span the year 2051. All two digit years will be assumed to be in the same century. If year is recorded with four digits, it is always processed correctly and the value of LAST20 is of no consequence.

`TRANSLATE=`(list)

"list" describes modifications to be made to the contents of the data file. It may contain one of:

`TIME/F, TIME/F/D`

and/or one of

`II/F, II/F/D`

F ("factor") may be an integer or a real value. If F is a real number, the translated value in FDATA will have the same number of digits after the decimal point. If F is an integer and D is omitted, there will be 2 digits. Alternately, the number of digits may be specified explicitly by D ("digits"). If D is a real number, it is truncated to integer. If D is specified as 0, it defaults to 2. The maximum value of D is 12. The number of digits that may be requested in F is limited by the precision of the computer.

For example, either of the following can be used to request values of TIME in FDATA that have 4 digits to the right of the decimal point:

```
TIME/1.0000
TIME/1/4
```

Another example is

```
II/0.01/6
```

which divides II values by 0.01, and writes 6 digits to the right of the decimal point.

If F is specified as "24" (or 24.0..), the options involving TIME (II) can be used to convert the units of time (of the steady-state interval) from hours to days. The TIME (II) data item is first processed as if the option were not present. Then the resulting value is divided by F.

Note: The value of TIME is divided by F, whether or not day-time translation occurs (i.e., whether or not relative times are being computed by NM-TRAN). Similarly, the value of II is divided by F whether or not ":" appears in any II data value.

`BLANKOK`

Specifies that blank lines are permitted in the NM-TRAN data set. With all versions prior to NONMEM VI 2.0 a blank line was permitted, and was copied to the NONMEM data set. A warning message was issued. With later versions, NM-TRAN stops with an error message when there is a blank line in the NM-TRAN data set. Option BLANKOK restores the previous behavior. There is no abbreviation. BLANKOK must be coded in full.

`MISDAT=`*r* (NM74)

A numerical value indicating a missing data value in the data set, which is displayed on $TABLE table outputs, but is safely interpreted as 0 by other steps of NONMEM. May be used up to 20 times.

Example: `$DATA mydatafile MISDAT=1.0E-99 MISDAT=1.0E-102`.

`REPL=`*n* (NM75)

When the REPL=n option of $DATA is coded, the NONMEM data set is considered to be a template data set. NONMEM itself replicates the template data set n times at the start of the problem to create an expanded NONMEM data set.

(Note that the NONMEM data set is typically the file FDATA generated by NM-TRAN. If there is nothing for NM-TRAN to change and the format is supplied on the $DATA record, the file named on the $DATA record is the NONMEM data set. See NONMEM Users Guide, Part IV.)

The REPL option may be used with the REPL_ data item. If both are used the REPL_ data item applies first, and the REPL option applies second.

See REPL_ data item.

The REPL option and REPL_ data item are meant to be used with $SIMULATION or $DESIGN.

For important information, see Guide Introduction_7 Section "$DATA REPL".

(See Guide Introduction_7 "Extension to $DATA IGNORE=st filtering")

Another change in NONMEM VI 2.0 is that tab characters (and other characters that are smaller than blank in the computer's collating sequence, such as carriage return ^M) are permitted in NM-TRAN input files. In the NM-TRAN data set they are treated like commas, i.e., as field delimiters. In the NM-TRAN control stream they are converted to spaces. They are left unchanged in verbatim code. With NONMEM 7, the last non-blank character on the line is replaced by a space if it is a low-value character.

Note: The character ":" in TIME or II data items requests day-time translation of TIME or II values. These values must have the form hh:mm (i.e., hours:minutes). With NONMEM 7.3, values may also have the form hh:mm:ss (i.e., hours:minutes:seconds).
(See **Guide IV, section II.C.2**)

REFERENCES: Guide IV, section III.B.5
REFERENCES: Guide V, section 6.4
REFERENCES: Guide Introduction_7

$DEFAULT

MEANING: Specifies certain defaults for NONMEM
CONTEXT: NM-TRAN Control Record

USAGE:
`$DEFAULT [NOSUB=[-1|0|1]]`

SAMPLE:
`$DEFAULT NOSUB=1`

DISCUSSION:
$DEFAULT is optional. If present, it must appear following $PROB

Specifies certain defaults for NONMEM. If more than one $DEFAULT record is present in a given problem, the one used by NONMEM is the last one in the problem.

OPTIONS:

`NOSUB=[-1|0|1]`

With `NOSUB=0`, label substitution will be performed for all tasks in the problem. This is the default.
(See **$ABBREVIATED**).
With `NOSUB=1`, label substitution will not be performed.
With `NOSUB=-1`, revert to NONMEM default, which is to treat -1 as a 0.

If the `NOSUB` option is also specified on a task specification record ($TABLE, $SCATTER), then this value of NOSUB applies only for the current task. When speficied on a $EST record, the usual rule for options apply, in which the option varies carries into the next $EST record in the problem unless otherwise re-specified.

May also be coded `$DEFAULTS`.

REFERENCES: Guide Introduction_7

$DES

MEANING: Marks the beginning of abbreviated code for the DES routine
CONTEXT: NM-TRAN Control Record

USAGE:
`$DES`
abbreviated code

SAMPLE:

Suppose differential equations were used for ADVAN2, rather than the analytic solution. The $DES block would be as follows:

```
$DES
DADT(1)=-P(3)*A(1)
DADT(2)=P(3)*A(1)-P(1)*A(2)
```

DISCUSSION:
The $DES record is used to compute differential equations. It is used with PREDPP's general non-linear models (ADVAN6, ADVAN8, ADVAN9, ADVAN13, ADVAN14, ADVAN15, ADVAN16, ADVAN17,ADVAN18). General rules for abbreviated code are documented elsewhere
(See **abbreviated code**).
Specific rules follow.

ASSIGNMENT AND CONDITIONAL STATEMENTS

Left-hand quantities in assignment statements:

`DADT(1)`, `DADT(2)`, ... (Required. Derivative of each compartment amount with respect to time.)

It is important to note that PREDPP itself adds in the rates for any infusions that may be active. It is possible to introduce endogenous drug into a compartment by explicit terms in a differential equation, rather than by PREDPP dose event records. Drug introduced in this manner is not included by PREDPP in the computation of the output compartment.

DES-defined (i.e., PRED-defined) items.

Right-hand quantities in assignment statement and in conditions:

`A(1)`, `A(2)`, ... (Current compartment amounts; may be random variables.)

`P(1)`, `P(2)`, ... (Post-translation explicit basic PK parameters; may be random variables.)

PK-defined items (Post-translation implicit basic PK parameters; may be random variables.)

`T` (Time; may be random variable. `T` takes values continuously over an integration interval.) (Note that `T` may take values that are larger than the ending time of the interval.)

DES-defined items that appeared earlier as left-hand quantities in $DES.

Data item labels specified on the $INPUT statement may be used explicitly in DES and $DES abbbreviated code (rather than obtained as PK parameters). This may improve run time. As with $PK, values are those of the current event record (the event record to which the system is being advanced).

`THETA(n).`

Fortran functions that are continuous, such as SIN and COS, may be used. The INT, MOD, and ABS functions (or other functions that introduce a discontinuity) should not be used in the $DES block. Instead, use model event times MTIME in the $PK block, and either set flag variables in $PK that can be tested in $DES, or use MPAST in the $DES block. See Appendix 3 of Guide VI PREDPP. Several examples are available for the use of MTIME with $DES.
(See **mtime**, **model time examples**).
(See **Examples Using MTIME to Model Periodic Discontinuities in $DES**).
See step_circexa.ctl in the NONMEM examples directory.
See idr_circexa.ctl in the NONMEM examples directory.

Global Variables in Modules
Certain variables in FORTRAN Modules can be used.
(See **Variables_in_Modules**)
The following are of particular interest.

DOSTIM
DOSTIM is the time of a lagged dose or additional dose to which the system is being advanced. Abbreviated code in $DES may test DOSTIM. It may use DOSTIM on the right, unless DOSTIM is a random variable. In this case, it may be used on the right in a $PK block to define a random variable which may in turn be used on the right in the $DES block.

DOSREC
DOSREC is the dose record corresponding to the dose entering at DOSTIM. Abbreviated code in $AES may test items in DOSREC in a logical condition, and DOSREC may always be used on the right.

ISFINL
During simulation or a copying pass, and during the advance to a particular time (event or non-event time), ISFINL=1 at a final call to DES at that time. Otherwise, ISFINL=0.

DES_DER, MITER, METH
See Diff Eq Solver Settings.

Forbidden Variable Names:

`IR DA DP E(n) ETA(n) EPS(n) ERR(n)`

PSEUDO ASSIGNMENT STATEMENTS

`COMRES=-1`

RECORD ORDER:

Follows $SUBROUTINES $INPUT $MODEL $PK

DISPLAYED VALUES

DES-defined variables may be listed for display in $TABLE or $SCATTER. The values displayed for the first event record of the data set are 0's. The values displayed for the first event record of subsequent individuals are those from the last event record of the previous individual. This is because there is no call to DES with the first event record of the individual record, or with any other event record with the same value of TIME. NMTRAN provides a warning message.

```
(WARNING  48) DES-DEFINED ITEMS ARE COMPUTED ONLY WHEN EVENT TIME
 INCREASES. E.G., DISPLAYED VALUES ASSOCIATED WITH THE FIRST EVENT RECORD
 OF AN INDIVIDUAL RECORD ARE COMPUTED WITH (THE LAST ADVANCE TO) AN EVENT
 TIME OF THE PRIOR INDIVIDUAL RECORD.
```

The values displayed for event records other than the first are evaluated at the event time (as are all other displayed items such as PK-defined values and data record items).

Note that multiple calls to DES are associated with each event record. Some (including the final call during integration) may have a value of T that is larger than the end time of the integration interval. When NONMEM is performing simulation or a copying pass (COMACT>0), DES is called immediately after the advance to an event time or non-event time, with a value of T equal to this time, so that displayed values are the final value of T. Global variable ISFINL is set by PREDPP to 1 on this final call to DES. ISFINL can be tested on a WRITE statement if such statements are used in DES.

(See **des**, **advan6**, **advan8**, **advan9**).

REFERENCES: Guide IV, section V.C.7
REFERENCES: Guide VI, section VI.C

$DESIGN (NM75)

MEANING: Instructions for Clinical Trial Design Evaluation and Optimization
CONTEXT: NM-TRAN Control Record

DISCUSSION:
The optimal design process can help you in designing or evaluating a clinical trial. It may be desired to evaluate specified time points, or find the optimal time points, dose levels, and number of time points appropriate for a particular sub-design, etc. The design algorithms have been modeled after POPED by Hooker et al., and PFIM by Mentre et al. (see references [22-27]) in Introduction to NONMEM 7.

USAGE:

`$DESIGN` Record Options

[APPROX=[**FO**|FOI|FOCE|LAPLACE|LAPLACEI]]
[OFVTYPE=[**0**|1|2|3|4|5|6|7|8]]
[GROUPSIZE=*n*]
[FIMTYPE=[**0**|1|2|3]]
[FIMDIAG=[**0**|1|2|3]]
[VARCROSS=[**0**|1]]
[EOPTD=1]
[SEED=*n*]
[CLOCKSEED=[0|1]]
[MODE=[**0**|1|2|3]]
[DATASIM=[**0**|1]]

Additional Control Options for $DESIGN

[SIGL=*n*]
[SIGLO=*n*]
[**ABORT**|NOABORT|NOHABORT]
[MAXEVALS=*n*]
[PRINT=*n*]
[NUMERICAL|**NONUMERICAL**]
[LIKELIHOOD|-2LOGLIKELIHOOD|-2LL]
[SLOW|**NOSLOW**|FAST]
[POSTHOC|**NOPOSTHOC**]
[NOPRIOR=[0|1]]
[FILE=filename]
[FORMAT|DELIM=s_3]
[FNLETA=*n*]

Options for Setting up Types of Optimal Design

[NELDER]
[FEDOROV]
[RS]
[STGR]
[DISCRETE]
[DISCRETE_RS]
[DISCRETE_SG]
[DESEL=label]
[DESELSTRAT=label]
[DESELMIN=label]

[DESELMAX=label]
[NMIN=label]
[NMAX=label]
[STRAT=label]
[STRATF=label]

SAMPLE:

```
$DESIGN APPROX=FOCEI MODE=1 NELDER FIMDIAG=0
        DATASIM=1 GROUPSIZE=32 OFVTYPE=0
```

will produce the most empirical, "clinical trial simulation" (CTS) style covariances, complete with simulated etas and eps, and standard FIM is assessed. If FIMDIAG>0, then a y-expectation covariance will be evaluated, but mode will be evaluated with the simulated data.

DISCUSSION:

The record name can be shortened to $DESI. Another name for the record is $OPTDESIGN, which can be shortened to $OPT.

The following is a brief summary of the options. See Introduction to NONMEM 7 "Clinical Trial Design Evaluation and Optimization" for more detail and more examples. The options are described in the following sections:

(1) $DESIGN Record Options
(2) Additional Control Options for $DESIGN
(3) Options for Setting up Types of Optimal Design

(1) $DESIGN Record Options

APPROX=FO (default)

The nlme approximation method is specified here. First order (no interaction) is the default, and the appropriate type of covariance matrix is evaluated or used in optimization of the design. Other options are:

FOI

First order interaction with interaction"

FOCE

first order conditional estimation

FOCEI

first order conditional estimation with interaction

LAPLACE

Laplace conditional estimation

LAPLACEI

Laplace conditional estimation with interaction

OFVTYPE=1 (default)

The objective function types are comparable to those of PopED:

0,1,3,4,5: design type: d-optimality, -log(det(FIM)), where FIM=Fisher Information Matrix (inverse of variance-covariance).
2: design type: a-optimality, -1/tr(1/FIM)
6: design type: ds-optimality, -log(det(FIM))+log(det(FIMuninterstingly))
To identify parameters as uninteresting, place UNINT at the parameter in the same manner you would place FIX.
7: design type: r-optimality (relative standard error),-1/tr(sqrt(1/FIM)/Paramer))
8: optimal design type: Individual Bayesian FIM, -log(det(Bayes

FIM)), as described in the PFIM 4.0 manual [26].

9: Same as option 2, and using the UNINT filter.
10: Same as option 7, and using the UNINT filter.

GROUPSIZE=1 (default)

The GROUPSIZE is comparable to that of POPED, in which the FIM is multiplied by this number to provide the subject number size of the dataset template. For a template of one subject, GROUPSIZE would then offer the variance-covariance expected from GROUPSIZE number of subjects.

FIMTYPE or FIMDIAG=0

FIMTYPE or FIMDIAG may be set to the following, and corresponds to fim-calc.type in POPED:

0 (default): Full information matrix

Uses finite difference assessment for Theta, Omega, and Sigma variances and covariances.

1 Create a block diagonal information matrix of the estimates

2 Create a block information matrix

3 Create a full information matrix

VARCROSS=0 (default)

Standard NONMEM residual variance modeling.

VARCROSS=1

Residual Standard Deviation Modeling
VARCROSS=1 means to treat the residual variance model in the manner of PFIM 4.0, as described in the manual.

FIMTYPE=1 VARCROSS=1 is equivalent to fim.calc.type=4 in POPED, and diagonal option in PFIM.

EOPTD=1

For each iteration, this creates a random sample of thetas, omegas, or sigmas, using the prior information. $PRIOR NWPRI prior information is required, and PLEV=0.999 must be specified. Best used with STGR. See example opt-design16.

SEED=223345

Set the starting seed for any random samples to be created, whether for EOPTD=1, or for FOCE type FIM in creating random etas (see below).

CLOCKSEED=0 (default)

As of nm75, the actual starting seed will be 10000*(seconds after mid-night)+SEED (SEED may be set to 0 for this purpose), if CLOCKSEED=1. This allows a control stream to produce different stochastic results for auto-mated replications, without the need to modify the seed value in the control stream file in each replication.

MODE=0 (default), 1, 2, or 3

Used for specifying data and prediction value type when specifying APPROX=FOCEI. In NONMEM report:
0 EVALUATE AT F(ETAsim) DURING CONDITIONAL DESIGN ASSESSMENT
1 EVALUATE AT THE MODE F(ETAhat) DURING CONDITIONAL DESIGN ASSESSMENT

2 EVALUATE AT F(ETAsim)-G*ETAsim DURING CONDITIONAL DESIGN ASSESSMENT
3 EVALUATE AT F(0) DURING CONDITIONAL DESIGN ASSESSMENT
In other words,
MODE=0 means FOCEI with data at F(ETAsim), and predicted function evalauted at f(ETAsim), is to be used. This method works well.
MODE=1 means FOCEI with data at F(ETAsim), and predicted function evaluated at the mode, f(ETAhat), is to be used. The results are not satisfactory.
MODE=2 means linearized FOCEI, with data at F(ETAsim)-G*ETAsim, and predicted function at F(ETAsim). Works well.

DATASIM=0 (default)

Normally, y-expectation evaluation of the FIM is performed. To actually simulate data, set DATASIM=1, and along with APPROX=FOCEI, this will produce simulated etas as well.

(2) Additional Control Options for $DESIGN

The following options may be set within the $DESIGN record, and they operate exactly as their equivalents in the $EST record. Thus, the following are equivalent:

```
$DESIGN FIMDIAG=1 OFVTYPE=6 APPROX=FO
        MAXEVAL=9999 NOHABORT PRINT=20 SIGL=10 POSTHOC

$DESIGN FIMDIAG=1 OFVTYPE=6 APPROX=FO
$EST    MAXEVAL=9999 NOHABORT PRINT=20 SIGL=10 POSTHOC
```

The options are as follows.

SIGL
SIGLO
SIGL
SIGLO
ABORT/NOABORT/NOHABORT
MAXEVAL
MAXEVAL=0 indicates design evaluation (the default)
MAXEVAL>0 indicates design optimization
PRINT control iteration printing during optimal design
NUMERICAL/NONUMERICAL
-2LL/LIKELIHOOD/LLIKELIHOOD
SLOW/NOSLOW/FAST
POSTHOC
NOPRIOR
FORMAT
FILE
FNLETA

(3) Options for Setting up Types of Optimal Design

The additional options for $DESIGN listed below are for optimizing parts of the design components. For example, the DESEL, DESELSTRAT, DESELMIN, DESELMAX can be specified for all the various design elements that you want optimized. It might be TIME for time samples, AMT for dose, or some type of covariate specific for the problem. Certainly, any combination of covariates can be requested to be optimized.

NELDER

Use Nelder method to search for optimal continuous parameters

FEDOROV
: Use to find ideal set of discrete time points from a larger set of possible time points.

RS Random Search method to find optimal continuous parameters

STGR
: Stochastic Gradient method to find optimal continuous parameters

DISCRETE
: Find optimal number of time points for each sub-design (subject template), and use NELDER method to find optimal continuous parameters.

DISCRETE_RS
: Find optimal number of time points for each sub-design (subject template), and use RS method to find optimal continuous parameters.

DISCRETE_SG
: Find optimal number of time points for each sub-design (subject template), and use STGR method to find optimal continuous parameters.

Specific parameters must be specified to be optimized.

This is done using the following options:

DESEL=label
: The data item (column) that contains the design element (DESEL) values that are to be modified and optimized. For example, TIME column would indicate that you want times to be estimated.

DESELSTRAT=label
: The data item (column) indicating stratification. The DESELSTRAT data item should contain integer indices to indicate what values are to be shared, and estimated together. If a record contains a value of 0 in the DESELSTRAT column, then this record is not included in the estimation process, and its value (say its time value in DESEL=TIME column) will not be changed. If the record contains a value >0 in DESELSTRAT, let us suppose a 1, then all records with the value of 1 in DESELSTRAT will share the same time value (or whatever DESEL selected), extimated together. Those records with value 2 will be another set of records which will share a time value, etc. Thus, within a subject, there may be a PK record and a PD record which should share the same time value. Also, a group of subjects may share the same time values. Within a subject, times will be automatically sorted as they are changed, so that NONMEM's time ordering policy is not violated.

DESELMIN=label
: The data item (column) containing the minimal value

DESELMAX=label
: The data item (column) containing the maximal value

  You must impose boundaries on the values that are being optimized. That is done with these two data items. Only those records with a stratification value >0 in DESELSTRAT column will require a min and max value, and only those records that define that stratification value for the first time.

  All four DESEL items must be specified: DESEL,DESELSTRAT,DESELMIN,DESELMAX. They may be repeated for as many design elements are to be optimized.

For example for times and amounts:

```
DESEL=TIME DESELMMIN=TMIN  DESELMAX=TMAX   DESELSTRAT=TSTRAT
DESEL=AMT  DESELMIN=AMTMIN DESELMAX=AMTMAX DESELSTRAT=AMTSTRAT
```

NMIN=label

The name of the data item (column) containing minimal number of time points to the subject.

NMAX=label

The name of the data item (column) containing maximal number of time points to the subject.

If NMIN<0 or NMAX<0, then most previous non-negative value is used. The NMIN and NMAX column are only used for the DISCRETE* analyses, to bound the number of time points that may be permitted for a given subject. With DISCRETE*, the total N of time points among all subjects is determined by the total number of time points whose MDV=0 in the starting data set.

STRAT=label

The data item (column) containing grouping or stratification number pertaining to that subject.

STRATF=label

The data item (column) containing starting fraction representation for the stratification value in column STRAT.

If STRAT and STRATF are specified, and there is at least one STRAT value >0, then the SRATF values are optimized, and represent the weight to the contribution of that subject to the Information matrix. For STRAT<=0, then their STRATF values are not optimized, and remain fixed at their initial values, but are still used as weights to the information matrix. It is up to the user to ensure that the sum of STRATF values among unique STRAT values sum to 1. If value of STRATF<0.0, then that subject is not included in the assessment.

Additional $COVARIANCE Control Options for $DESIGN

In addition, $DESIGN sets up the covariance step as $COV MATRIX=R UNCONDITIONAL without the user requiring this record entered in the control stream. If you wish to specify additional control for the covariance, you can add these in a $COV record, such as:

```
$DESIGN FIMDIAG=1 OFVTYPE=6 APPROX=FO MAXEVAL=9999 NOHABORT PRINT=20 SIGL=10
$COV PRINT=E
```

RECORD ORDER:

Same as $ESTIMATION or $SIMULATION

REFERENCES: Guide Introduction_7

$ERROR

MEANING: Marks the beginning of abbreviated code for the ERROR routine
CONTEXT: NM-TRAN Control Record

USAGE:
`$ERROR`
abbreviated code

DISCUSSION:
The $ERROR record is used to model intra-individual error in observed values. It is used with PREDPP. It can also be used to to convert predictions from PREDPP, i.e., scaled drug amounts, to other types of predictions (for example, to obtain the prediction of a drug effect as a function of concentration, in a pharmacodynamic study). General rules for abbreviated code are documented elsewhere (See **abbreviated code**). Specific rules follow.

ASSIGNMENT AND CONDITIONAL STATEMENTS

Left-hand quantities in assignment statements:

`Y` (Required. The modeled value for the dependent variable under the statistical model.)

ERROR-defined (i.e., PRED-defined) items.

Left-hand quantities from the $PK block, if they are not random variables **and** neither $PK nor $ERROR include the `COMRES=-1` pseudo-assignment statement.

Right-hand quantities in assignment statement and in conditions:

`F` (Required. The value of the scaled drug amount in the observation compartment.)

Data item labels specified on the $INPUT statement.

`THETA(n)`

`ETA(n)` (Required if the data are single-subject, and can be coded ERR(n). Optional if the data are population.)

`EPS(n)` (Required if the data are population, and can be coded ERR(n).)

ERROR-defined items that appeared earlier as left-hand quantities. This includes `Y`.

Left-hand quantities from the $PK block, if neither $PK nor $ERROR include the `COMRES=-1` pseudo-assignment statement.

`NEWIND`
Same as the NEWIND argument passed by NONMEM to PREDPP.
`NEWIND=0`: First record of the data set. THETA value may differ from value at last call with this record.

NEWIND=1: First record of the data set, THETA value does not differ from value at last call with this record, and PRED is nonrecursive (see **I_REC**), or,
First record of a subsequent individual record.
NEWIND=2: Subsequent data record of an individual record.

NEWL2
NEWL2=1: First record of an L2 record.
NEWL2=2: Otherwise.

ICALL
Same as the ICALL argument passed by NONMEM to PREDPP.
ICALL=1: Initialization.
ICALL=2: Normal call.
ICALL=3: Finalization.
ICALL=4: Simulation.
ICALL=5: Expectation.
ICALL=6: Data Average.
Special rules apply to blocks of abbreviated code that are executed when ICALL is not 2.
(See **Initialization-Finalization block**, **Simulation block**).
(See **Expectation block**, **Data_Average block**).

Global Variables in Modules
Certain variables in FORTRAN Modules can be used. (See **Variables_in_modules**) The following are of particular interest.

A(n) (Amount in compartment n.) (See **State Vector A**)
Note: If there is no verbatim code and no explicit use of a subscripted variable A in the $ERROR block, then the symbol A can be used as a data item label or as a name of an ERROR-defined item.

Forbidden Variable Names:
IDEF IREV EVTREC NVNT INDXS G HH DADT(n) E(n) P(n)

PSEUDO ASSIGNMENT STATEMENTS

COMRES=-1

CALLFL=-1: Call with every event record.
CALLFL=0: Call with every observation record.
CALLFL=1: Call once per individual record.

Of the last three, CALLFL=-1 is the default, except when the abbreviated code consists of one line specifying a simple additive or proportional error model, and no verbatim code is present.
CALLFL=2: Call once per problem.
In effect, this is supplied by NM-TRAN for the simple error models, but it cannot be specified explicitly in the $ERROR record.
(See **callfl**).

Pseudo-assignments statements may be enclosed in parentheses. If two of them are present in the same set of parentheses, separate them with a semicolon. A calling protocol phrase may be used within parentheses instead of a pseudo-assignment statement, and then either upper or lower case may be used . E.g.,

```
$ERROR (ONCE PER IR)        ; same as CALLFL=1
```

```
$ERROR (ONLY OBSERVATIONS) ; same as CALLFL=0
$ERROR (EVERY EVENT)       ; same as CALLFL=-1 (default)
```

RECORD ORDER:

Follows $SUBROUTINES and $INPUT
Follows $MODEL (if a general model such as ADVAN6 is used)
Follows $PK (if present)

REFERENCES: Guide IV, section V.C.6
REFERENCES: Guide V, section 8
REFERENCES: Guide VI, section IV

$ESTIMATION RECORD OPTIONS

MEANING: Instructions for the NONMEM Estimation Step
CONTEXT: NM-TRAN Control Record

List of $ESTIMATION Record Options and Their Relevance to Various Methods

Classical methods are the First order method (METHOD=0 or METHOD=FO), the Conditional method (METHOD=1 or METHOD=FOCE) and the Hybrid method (METHOD=HYBRID). All other methods are EM (Expectation-Maximization) / Bayesian methods and are new to NONMEM 7.

Option	Classical	ITS	DIRECT	IMP	IMPMAP	SAEM	BAYES
-2LL	X	X	X	X	X	X	X
ATOL +	X	X	X	X	X	X	X
AUTO		X	X	X	X	X	X
BAYES_PHI_STORE							X
BIONLY							X
BOOTDATA	X	X	X	X	X	X	
CALPHA		X	X	X	X	X	X
CENTERING	X						
CINTERVAL		X	X	X	X	X	X
CITER/CNSAMP		X	X	X	X	X	X
CLOCKSEED			X	X	X	X	X
CONDITIONAL	X	X	X	X	X	X	X
CONSTRAIN		X	X	X	X	X	X
CTYPE	(CTYPE 4)	X	X	X	X	X	X
DERCONT			X	X	X	X	
DF				X	X		
DFS(CHAIN only)							
EONLY			X	X	X	X	
ETABARCHECK	X						
ETADER	X	X		X	X		
ETASAMPLES						X	X
ETASTYPE	X	X	X	X	X	X	
FAST	X	X					
FILE	X	X	X	X	X	X	X
FNLETA	X	X	X	X	X	X	X
FORMAT/DELIM	X	X	X	X	X	X	X
FPARAFILE	X	X	X	X	X	X	X
Option	Classical	ITS	DIRECT	IMP	IMPMAP	SAEM	BAYES
GRD		X	X	X	X	X	X
GRDQ				X	X		
GRID(Stieltjes)	X						
HYBRID	X						
IACCEPT				X	X	X	X
IACCEPTL				X	X		
INTERACTION	X	X	X	X	X	X	X
IKAPPA							
ISAMPEND			X	X	X	X	

ISAMPLE						X	X
ISAMPLE_M1						X	X
ISAMPLE_M1A						X	X
ISAMPLE_M1B						X	X
ISAMPLE_M2						X	X
ISAMPLE_M3						X	X
ISCALE_MAX				X	X	X	X
ISCALE_MIN				X	X	X	X
KAPPA							
KNUTHSUMOFF	X	X	X	X	X	X	X
LAPLACE	X	X	*	*	X	*	*
LEVCENTER(for $LEVEL)	X	X	X	X	X	X	X
LEVWT (for $LEVEL)	X	X	X	X	X	X	X
LIKE	X	X	X	X	X	X	X
LNTWOPI	X	X	X	X	X	X	X
Option	Classical	ITS	DIRECT	IMP	IMPMAP	SAEM	BAYES
MADAPT							
MAPCOV				X	X		
MAPINTER				X	X		
MAPITER				X	X		
MAPITERS						X	X
MASSRESET	***						X
MAXEVAL	X						
MCETA	X	X		X	X		
MSFO	X	X	X	X	X	X	X
MUM		X	X	X	X	X	X
NBURN						X	X
NITER/NSAMPLE		X	X	X	X	X	X
NOABORT	X	X	X	X	X	X	X
NOCOV**		X	X	X	X	X	X
NOHABORT	X	X	X	X	X	X	X
NOLABEL	X	X	X	X	X	X	X
NONINFETA	X						
NOOMEGABOUNDTEST	X						
NOPRIOR	X	X	X	X	X	X	X
NOSIGMABOUNDTEST	X						
NOSLOW	X	X					
NOSUB	X	X	X	X	X	X	X
NOTHETABOUNDTEST	X						
NOTITLE	X	X	X	X	X	X	X
NSIG/SIGDIGITS	X	X					
NUMDER	X	X	X	X	X	X	X
NUMERICAL	X	X	*	*	X	*	*
Option	Classical	ITS	DIRECT	IMP	IMPMAP	SAEM	BAYES
NUTS_BASE							
NUTS_DELTA							
NUTS_EPARAM	***						X
NUTS_GAMMA							
NUTS_INIT							

NUTS_MASS	***						X
NUTS_MAXDEPTH							
NUTS_OPARAM	***						X
NUTS_REG	***						X
NUTS_SPARAM	***						X
NUTS_STEPINTER							
NUTS_STEPITER							
NUTS_TERM							
NUTS_TEST							
NUTS_TRANSFORM							
Option	Classical	ITS	DIRECT	IMP	IMPMAP	SAEM	BAYES
OACCEPT							X
OLKJDF							
OLNTWOPI						X	X
OMEGABOUNDTEST	X						
OMITTED	X	X	X	X	X	X	X
OPTMAP	X	X		X	X		
ORDER	X	X	X	X	X	X	X
OSAMPLE_M1							X
OSAMPLE_M2							X
OSAMPLE_M3							X
OVARF							
PACCEPT							X
PARAFILE	X	X	X	X	X	X	X
PARAFPRINT	X	X	X	X	X	X	X
PHITYPE		X	X	X	X	X	X
POSTHOC	X	X	X	X	X	X	X
PREDICTION	X	X	X	X	X	X	X
PRINT	X	X	X	X	X	X	X
PRIORC	X	X	X	X	X	X	X
PSAMPLE_M1							X
PSAMPLE_M2							X
PSAMPLE_M3							X
PSCALE_MAX							X
PSCALE_MIN							X
RANMETHOD=nSmP			X	X	X	X	X
Option	Classical	ITS	DIRECT	IMP	IMPMAP	SAEM	BAYES
REPEAT	X						
REPEAT1	X						
REPEAT2	X						
SADDLE_HESS	X						
SADDLE_RESET	X						
SEED			X	X	X	X	X
SIGL	X	X	X	X	X	X	
SIGLO	X	X		X	X		
SIGMABOUNDTEST	X						
SLKJDF							
SLOW	X	X					
SORT	X						

STDOBJ				X	X			
STIELTJES	X							
SVARF								
TBLN(CHAIN only)								
THETABOUNDTEST	X							
THIN								X
TPU								
TTDF								
ZERO	X							
Option	Classical	ITS	DIRECT	IMP	IMPMAP	SAEM	BAYES	

+ ADVAN9,ADVAN13,ADVAN14,ADVAN15,ADVAN16,ADVAN17,ADVAN18
*May be needed to suppress error messages from NMTRAN or NONMEM.
**When last estimation step
***In prep for NUTS

(See **$ESTIMATION**).

REFERENCES: Guide Introduction_7

$ESTIMATION,$ESTM

MEANING: Instructions for the NONMEM Estimation Step
CONTEXT: NM-TRAN Control Record

USAGE:

```
$ESTIMATION
            [ABORT|NOABORT|NOHABORT]
            [ATOL=n]
            [AUTO=[0|1|2|3]]
            [BAYES_PHI_STORE=[0|1]]
            [BIONLY=[0|1]]
            [BOOTDATA=[0|1]]
            [CALPHA=n]
            [CENTERING|NOCENTERING]
            [CINTERVAL=n]
            [CITER=n | CNSAMP=n]
            [CONSTRAIN=n]
            [CTYPE=[0|1|2|3|4]]
            [DERCONT=[0|1]]
            [DF=n]
            [DFS=n]
            [EONLY=[0|1]]
            [ETABARCHECK|NOETABARCHECK]
            [ETADER=n]
            [ETASAMPLES=0 | ETASAMPLES=1]
            [ETASTYPE=0 | ETASTYPE=1]
            [FILE=filename]
            [FNLETA=n]
            [FORMAT|DELIM=s]
            [FO|NOFO]
            [FPARAFILE=[filename|ON|OFF]
            [GRD=s]
            [GRDQ=s]
            [GRID=(nr,ns,r0,r1)]
            [IACCEPT=x]
            [IACCEPTL=x]
            [IKAPPA=x]
            [INTERACTION|NOINTERACTION]
            [ISAMPEND=n]
            [ISAMPLE=n]
            [ISAMPLE_M1=n]
            [ISAMPLE_M1A=n]
            [ISAMPLE_M1B=n]
            [ISAMPLE_M2=n]
            [ISAMPLE_M3=n]
            [ISCALE_MAX=x]
            [ISCALE_MIN=x]
            [KAPPA=x]
            [KNUTHSUMOFF=n]
```

[LAPLACIAN|**NOLAPLACIAN**]
[LEVCENTER=[0|1]]
[LEVWT=*n*]
[LIKELIHOOD|-2LOGLIKELIHOOD]
[LNTWOPI]
[MADAPT=*n*]
[MAPCOV=*n*]
[MAPINTER=*n*]
[MAPITER=*n*]
[MAPITERS=[0|1]]
[MASSRESET=*n*]
[MAXEVALS=*n*]
[MCETA=*n*]
[METHOD=kind]
[MSFO=filename]
[MUM=*s*]
[NBURN=*n*]
[NITER=*n*]
[NOCOV=[0|1]]
[NOLABEL=[0|1]]
[NONINFETA=[0|1]]
[NOPRIOR=[0|1]]
[NOSUB=[0|1]]
[NOTITLE=[0|1]]
[NUMDER=[0|1|2|3]]
[NUMERICAL|**NONUMERICAL**]
[NUTS_BASE=*x*]
[NUTS_DELTA=*x*]
[NUTS_EPARAM=*n*]
[NUTS_GAMMA=*x*]
[NUTS_INIT=*x*]
[NUTS_MASS=[B|F|D|BD|DB|BBD|BBB]]
[NUTS_MAXDEPTH=*n*]
[NUTS_OPARAM=*n*]
[NUTS_REG=*x*]
[NUTS_SPARAM=*x*]
[NUTS_STEPINTER=*n*]
[NUTS_STEPITER=*n*]
[NUTS_TERM=*x*]
[NUTS_TEST=*n*]
[NUTS_TRANSFORM=*n*]
[OACCEPT=*n*]
[OLKJDF=*n*]
[OLNTWOPI]
[**OMEGABOUNDTEST**|NOOMEGABOUNDTEST]
[OMITTED]
[OPTMAP=*n*]
[ORDER=xxxf]
[OSAMPLE_M1=*n*]
[OSAMPLE_M2=*n*]

[OVARF=*x*]
[PACCEPT=*n*]
[PARAFILE=[filename|ON|OFF]
[PARAFPRINT=*n*]
[PHITYPE=*n*]
[POSTHOC|**NOPOSTHOC**]
[**PREDICTION**]
[PRINT=*n*]
[PRIORC]
[PSAMPLE_M1=*n*]
[PSAMPLE_M2=*n*]
[PSAMPLE_M3=*n*]
[PSCALE_MAX=*n*]
[PSCALE_MIN=*n*]
[RANMETHOD=[n|S|m|P]]
[REPEAT|**NOREPEAT**]
[REPEAT1|**NOREPEAT1**]
[REPEAT2|**NOREPEAT2**]
[SADDLE_HESS=*n*]
[SADDLE_RESET=*n*]
[SEED=*n*]
[CLOCKSEED=[0|1]]
[SELECT=*n*]
[SIGDIGITS|NSIGDIGITS=*n*]
[SIGL=*n*]
[SIGLO=*n*]
[**SIGMABOUNDTEST**|NOSIGMABOUNDTEST]
[SLKJDF=*x*]
[SLOW=1|SLOW=2]
[SLOW|**NOSLOW**|FAST]
[SORT|**NOSORT**]
[STDOBJ=*x*]
[STIELTJES]
[SVARF=*x*]
[TBLN=*n*]
[**THETABOUNDTEST**|NOTHETABOUNDTEST]
[THIN=*n*]
[TPU=*n*]
[TTDF=*n*]
[ZERO=list]

SAMPLE:

```
$ESTIMATION     MAXEVAL=450  PRINT=5
```

DISCUSSION:

Optional. Requests that the NONMEM Estimation Step be implemented. May also be coded $ESTM or $ESTIMATE. The Estimation Step obtains parameter estimates.

With NONMEM 7, multiple Estimation Steps can be implemented in a single problem. A sequence of two or more Estimation Steps will result in the sequential execution of each. Options specified in an $ESTIMATION record will carry over to the next $ESTIMATION record unless a new option is specified. If a particular option is not used by the method then the option will be ignored. The final parameter estimates from an

Estimation Step will be passed on as the initial estimates for the next Estimation Step. (See **$ESTIMATION_record_options**).

See also "Reserved Variables that are of Interest During the Estimation Step", at the end of this help item.

OPTIONS:

ABORT

During the Estimation Step, NONMEM does not implement theta-recovery when PRED sets the error return code to 1. (The PRED error return code n is set by the statement "EXIT n [k]" in abbreviated code, or by the statement IERPRD=n in user-supplied code, or by PREDPP when it detects errors.) This is the default.

NOABORT

During the Estimation Step, NONMEM implements theta-recovery, i.e., attempt to avoid values of theta which result in PRED error return code 1. In addition, most non-positive Hessian matrices will be forced to be positive definite, allowing the program to continue, and abnormal termination of the estimation step will occur less often.

NOHABORT

Perform positive definite correction at all levels of the estimation. This can hide a serious ill-posed problem, so use with care.

ATOL=*n* (NM72)

Absolute tolerance. Used only with ADVAN9, ADVAN13, ADVAN14, ADVAN15, ADAN16, and ADVAN17, ADVAN18, for which TOL is a relative tolerance. Sets ANRD=ATOL. The default is 12 (that is, accuracy is 10**(-12)). Usually the problem runs quickly when using this setting. On occasion, however, you may want to reduce ATOL (usually set it equal to that of TOL), and improve speeds of up to 3 to 4 fold.

By default, the value set on $SUBROUTINES record (or $TOL or TOL subroutine) is used. If ATOL is coded on $ESTIMATION, it overrides the default for that step. If ATOL is coded on $COVARIANCE, it overrides $ESTIMATION and/or the default for that step. With NONMEM 74, this feature is deprecated. A user-supplied TOL subroutine should test NM_STEP and set ANRD accordingly.

AUTO=0 (NM73)

NONMEM does not provide best settings of certain options. This is the default.

AUTO=1 (NM73)

Several options will be set by NONMEM that will allow best settings to be determined. User may still over-ride those options set by auto, by specifying them on the same $EST record. The AUTO option is ignored by the FO/FOCE/Laplace methods.

AUTO=2 (NM74)

AUTO=2 may be user with NUTS estimation to setup the alternative sampling strategy "Matt trick".

AUTO=2 (NM74)

AUTO=3 may be user with NUTS estimation to setup the alternative sampling strategy of eta sampling.

See **INTRODUCTION TO NONMEM 7** for more information.

BAYES_PHI_STORE=[0|1] (NM75)

If BAYES_PHI_STORE=1 then phi and eta values from each BAYES iteration will

be stored in root.iph.

`BIONLY=[0|1]` (NM75)

BIONLY stands for Bayesian individual parameters only, and when set to 1, will create new samples of individual parameters only, but will keep the population parameters fixed. May be used with BAYES_PHI_STORE=1 or write statements capturing the individual parameters samples. See **INTRODUCTION TO NONMEM 7** for more information.

`BOOTDATA=[0|1]` (NM75)

By default (BOOTDATA=0), when data are selected based on $SIML BOOSTRAP, the randomly selected subjects are analyzed during the subsequent estimation method. If BOOTDATA=1, then the subjects not selected are analyzed.
See **INTRODUCTION TO NONMEM 7** for more information.

`CALPHA=`*n*

n is a value between 0.01 and 0.05. Alpha error rate to use for linear regression test to assess statistical significance. Default is 0.05.

`CENTERING`

Requests that the average conditional estimates of each eta be constrained to be close to 0. May only be used with `METHOD=1`. Not permitted with `INTERACTION`.

`NOCENTERING`

Requests that the average conditional estimates of each eta not be constrained. This is the default.

`CINTERVAL=`*n*

Every *n* iterations is submitted to the convergence test system. If CINTERVAL=0, then a best CINTERVAL will be found, then used.

`CITER=`*n*

n is the number of latest PRINT or CINTERVAL iterations on which to perform a linear regression test (where independent variable is iteration number, dependent variable is parameter value). If CITER=10, then 10 of the most recent PRINTed or CINTERVAL iterations are used for the linear regression test. Default is 10. May also be coded CNSAMP. If CINTERVAL is not specified, then the PRINT option is used.

`CONSTRAIN=`*n* (NM72)

Requests simulated annealing for parameters to slow the rate of reduction of the elements of OMEGA during the burn-in phase of the SAEM method, allowing for a more global search of parameters that minimize the objective function. Values for *n* are:

0 or 4 No simulated annealing.

1 or 5 Requests simulated annealing for OMEGA.

2 or 6 Requests simulated annealing for SIGMA.

3 or 7 Requests simulated annealing for both OMEGA and SIGMA.

Default is 1.

When CONSTRAIN>=4, simulated annealing is also performed on diagonal elements of OMEGA that are fixed to 0 to facilitate any associated thetas.

Simulated annealing is performed by subroutine CONSTRAINT.

The $ANNEAL record facilitates EM search methods for this additional annealing technique. The subroutine CONSTRAINT may also be used to provide any kind of constraint pattern on any parameters.
(See **$ANNEAL**).
The user may modify the subroutine CONSTRAINT that performs the simulated annealing algorithm.
(See **additional_output_files**, **raw_output_file**).

`CTYPE=[0|1|2|3|4]`

CTYPE is used to define the termination test to be applied to the burn-in phases for `SAEM` and `BAYES` methods and to the estimation phases for the `ITS`, `IMP`, and `IMPMAP` methods. (CTYPE=4 Applies to classical methods FO/FOCE/Laplacean.)
`CTYPE=0` indicates no termination test, the default.
`CTYPE=1` indicates that the test should be applied to the objective function value, THETA's and SIGMA's but not to the OMEGA's.
`CTYPE=2` indicates that the test should be applied to the objective function value, THETA's, SIGMA's and diagonal elements of OMEGA.
`CTYPE=3` indicates that the test should be applied to the objective function value, THETA's, SIGMA's and all OMEGA elements.
`CTYPE=4` indicates that NONMEM should test if the objective function has not changed by more then NSIG digits beyond the decimal point over 10 iterations, even though a parameter may oscillate at some digit. If this condition is satisfied, the estimation will terminate successfully. Applies to FO/FOCE/Laplacean methods.

`DERCONT=[0|1]` (NM73)

The derivative continuity test (`DERCONT`) by default is off (0). When `DERCONT=1`, the partial derivative of the objective function with respect to thetas will perform an additional test to determine if a backward difference assessment is more accurate than a forward difference assessment. The forward difference assessment can differ greatly from the backward difference assessment in cases of extreme discontinuity when varying certain thetas by even just a small amount in the model results in a large change in objective function, (such as a viral model in which a very small change in the potency of an anti-viral agent results in widely varying time of return of viral load). This results in standard errors being poorly assessed for thetas that do not have inter-subject variances associated with them. Setting `DERCONT=1` slows the analysis, but can provide more accurate assessments of SE in such models. The `DERCONT` works only for the Monte Carlo EM algorithms such as `IMP`, `DIRECT`, `IMPMAP`, and `SAEM`.

`DF=`*n*

The proposal density is to be a t distribution with *n* degrees of freedom. Default is 0, a normal density. Used with the `IMP` and `IMPMAP` methods.

`DFS=`*n* (NM73)

Degrees of freedom for the Sigma matrix for simulation purposes by CHAIN.

`DFS=-1`

This is the default. The cholesky elements are uniformly varied over the interval (1-accept)*initial value and (1+accept)*initial value.

`DFS=`*n*

The SIGMA matrix is randomly created with an inverse Wishart distribution centered about the initial SIGMA values, with degrees of freedom DFS for dispersion.

DFS=0
As above, but the size of the SIGMA matrix is used as degrees of freedom.

DFS=>1000000
SIGMA is fixed at its initial value.

EONLY=[0|1]
A value of 1 indicates the IMP objective function should be evaluated by performing only the expectation step without advancing the population parameters. Default is 0.

ETABARCHECK
There is an etabar statistic (See **etabar**) from a previous problem, and the P-value associated for the etabar statistic with the problem at hand relates to a hypothesis test that the true etabar is the same as that with the previous problem.

NOETABARCHECK
The P-value associated for the etabar statistic (See **etabar**) relates to a hypothesis test that the true etabar is 0. This is the default.

ETADER=*n* (NM73)
For evaluating individual variances by numerical derivative methods. In evaluating the MAP objective function, the term log(Det(V)) must be evaluated to obtained the marginal or integrated posterior density, where V is the eta Variance matrix based on the subject's posterior density. With ETADER>0, SLOW option may be needed.

ETADER=0
Expected value V, using analytical first derivatives

ETADER=1
Expected value V, using forward finite difference numerical first derivatives. Needed if not all code evaluating F and Y derivatives with respect to eta are available for processing by NM-TRAN or in user supplied code.

ETADER=2
Expected value V, using central finite difference numerical first derivatives. Needed if not all code evaluating F and Y derivatives with respect to eta are available for processing by NM-TRAN or in user supplied code.

ETADER=3
2nd derivative method of evaluating V, using numerical second derivatives of -log(L) with respect to etas. This is equivalent to using the "Laplace NUMERICAL" method, even though FOCE may be selected.

ETASAMPLES=[0|1] (NM74)
Used with $EST METHOD=SAEM or $EST METHOD=BAYES. ETASAMPLES=0 is the default. ETASAMPLES=1 causes individual ISAMPLE random eta samples per subject, to be written to root.ets, where root is the root name of the control stream file.
See "Stochastic Approximation Expectation Maximization (SAEM) Method" in Guide INTRODUCTION TO NONMEM 7.

ETASTYPE=0 (NM73)
Eta shrinkage is averaged for all subjects. This is the default.

ETASTYPE=1
Eta shrinkage is averaged only among subjects that provided a non-zero derivative of their data likelihood with respect to that eta. (See **etasxi**).

FAST (NM74)
The FAST option is available for FOCE/ITS methods. The FAST method allows use

of analytical theta derivatives to facilitate FOCE analysis. All thetas should be MU-referenced in the manner described in Guide INTRODUCTION TO NONMEM 7, "MU Referencing". For thetas that should not have inter-subject variability, or should not be MU referenced, MU reference it anyway by adding addional etas and assigning them to these thetas through MU referencing, but set their associated omega values to 0.0 FIXED.

`FILE`=filename

Name for the raw output file. Parameter estimates and objective function value will be printed to this file every printed iteration as indicated by the `PRINT` option. Default: root.ext, where root is the name of the control file (not including any extension; "nmbayes" if the name is not specified on the nmfe command line). Note that the names of additional output files are not affected by this option. Additional output files have extensions .ext, .phi, .phm, .shk, .shm .grd, .xml, .cov, .cor, .coi, .cnv, .smt, .rmt, .imp, .npd, .npe, .npi, .fgh, .clt, .vpd, .vpt, .ets, .vpt, .bfm, .iph They always have the name of the control file or "nmbayes" for root.
(See **additional_output_files**, **raw_output_file**).

`FNLETA`=*n* (NM72)

For a thorough discussion:
See **INTRODUCTION TO NONMEM 7, General New Options for $ESTIMATION Record**
`FNLETA=0` requests that the FNLMOD and FNLETA routines not be called after the Estimation and Covariance steps are completed. May improve run time. Post-hoc etas for METHOD=0 will not be computed.
All table outputs (diagnostics and user selected items) will use EBE's from final estimation method (conditional modes for FO/FOCE/Laplace/ITS, conditional means for IMP/SAEM, MCMC posterior means for BAYES).
`FNLETA=1` is the default. FNLMOD and FNLETA routines are called as usual.
Diagnostics depending on EBE's such as CWRES, CIWRES, CIPRED, etc., will use EBE's based on the final estimation method (conditional mode for FO/FOCE/Laplace/ITS, conditional mean for IMP/SAEM, MCMC posterior means for BAYES), while user selected items will use EBE's from the FNLETA step (eta modes).
`FNLETA=2` Requests that the estimation step not be done. All table outputs will use a common set of EBE's from an imported source. This has value if you loaded the individual etas from an MSF file, or from a $PHIS/$ETAS record, and you want to calculate $TABLE items based on those etas, rather than from a new estimation.
`FNLETA=3` (as of nm74) Like FNLETA=1, will call FNLETA, and all table outputs (diagnostics and user selected items) will use EBE's from the FNLETA step (eta modes).

`FORMAT`=*s*

Format for the raw output file and all additional output files. *s* defines the delimiter [,|s(pace)|t(ab)] followed by a Fortran format specification. The default is s1PE12.5. For more details, see the format help item:
(See **format**).
May also be coded `DELIM`.

`DELIM`=*s*

Same as FORMAT option.
See **INTRODUCTION TO NONMEM 7, $EST: Format of Raw Output File**

FO Requests that the First-Order Model be used with METHOD=1 and CENTERING. Cannot be used with LAPLACIAN.

NOFO
Requests that the First-Order Model not be used with METHOD=1 and CENTERING. This is the default.

FPARAFILE=filename (NM74)
Final etas (empirical Bayes estimates; EBE's) are evaluated after the last Estimation Step (when FNLETA=1). This computation is parallelized if parallelization is on for the final Estimation Step.
FPARAFILE=filename specifies a different parafile than was used for the Estimation Step.
FPARAFILE=ON turns on parallelization for the EBE's.
FPARAFILE=OFF turns off parallelization for the EBE's.
The FPARAFILE option may be specified on any $ESTIMATION record, but applies only after the last Estimation Step.

GRD=*s*
s is a string of [G|N|D|S]'s with each symbol representing a THETA or SIGMA parameter in numerical order. The first m letters of GRD refer to the m THETA's. Then the m+1th letter refers to SIGMA(1,1), m+2 refers to SIGMA(2,2), etc (going along the diagonal of SIGMA). Omitted symbols are assumed to be D.
G indicates that the THETA should be Gibbs sampled. N indicates the THETA should be sampled using the Metropolis-Hasting algorithm. S indicates that the THETA is being used to model a SIGMA parameter. S is used with Monte Carlo EM methods. D (default) indicates the program will decide. G and N are used only with the BAYES method.
Default is DDDD...

GRD=t1v1(n1):t2v2(n2):t3v3(n3)...
An alternative syntax may be used. T is parameter type (T for theta, S for sigma-like theta). V is a letter (S,D or N), and n is a number list. For example, to specify thetas 3, and 5 through 8 to be Gibbs samples, theta 4 is sigma-like, and sigmas 1-3 are to be Metropolis-Hastings processed,
GRD=TG(3,5-8):TS(4):SN(1-3)
Thetas and sigmas not specified are given a default D designation.

GRDQ=0 (NM74)
Optional. The gradient quick option, called GRDQ, allows thetas that must be gradient assessed (such as those that are not mu-referenced) and SIGMAS to be more quickly evaluated by not evaluating the gradients for every one of the ISAMPLE random samples, but chooses a subset of the most important samples.

GRID=(nr,ns,r0,r1)
Optional. May be used with STIELTJES. For nr, ns, r0, and r1, see the Introduction to Version VI 2.0. Briefly, a grid is obtained by first taking the interval [r0,r1] of the length axis and dividing this interval into nr equal subintervals. ns may be thought of as the number of points in a single quadrant of a 2-dimensional ellipse in n-space. Constraints are nr<=100, ns<=9999, 0<r0<r1<1. If r1>.9999, there is no tail region. nr and ns should be integers. The default values are: GRID=(3,1,.6,.9).

IACCEPT=*x*
Has different meanings, depending on the method. With SAEM and BAYES, the scaling of OMEGA is adjusted so that samples are accepted *x* fraction of the time.

See ISAMPLE_M2. Default is 0.4.

With Importance sampling (IMP INTERACTION), expand proposal (sampling) density variance relative to conditional density so that on average conditional density/proposal density=IACCEPT (default 0.4). For very sparse data or highly nonlinear posterior densities (such as with categorical data), you may want to decrease to 0.1 to 0.3.

IACCEPT=0

For importance sampling only, you may set IACCEPT=0.0, and NONMEM will determine the most appropriate IACCEPT level for each subject, and if necessary, will use a t- distribution (by altering the DF for each subject) as well. If IACCEPT=0, the individual IACCEPT values and DF values will be listed in root.imp.

IACCEPTL=*x* (NM74)

A scale to a second multi-variate normal density, to cover long tails in the posterior density (hence L for long tails), in combination with the normal IACCEPT value to cover the posterior density near the mode.

IKAPPA=*x*] (NM74)

Used in computing weight for individual parameters in ISAMPLE_M1B mode. IKAPPA is 1.

INTERACTION

The dependence on etas of the model for intra-individual random error is preserved in the computation of the objective function.

Cannot be used with CENTERING. With NONMEM 7.3, This is the default with EM/Bayes methods and is supplied if NOINTERACTION is not specified by the user. With NONMEM 7.4, INTERACTION is not supplied if LIKELIHOOD is present.

NOINTERACTION

Always set etas to 0 during the computation of the model for intraindividual random error. This is the default with non-EM/Bayes methods.

ISAMPEND=*n* (NM73)

For SAEM, if ISAMPEND is specified as an upper integer value (usually 10), then NONMEM will perform a ISAMPLE preprocess to determine the best ISAMPLE value. See also STDOBJ.

ISAMPLE=*n*

When used with the IMP or IMPMAP methods *n* is the number of random samples per subject used for the expectation step. Default is 300. When used with the SAEM or BAYES method *n* is the number of chains used by the Metropolis-Hastings algorithm for individual parameter estimation. The default is 2 for SAEM and 1 for BAYES.

A kernel is the Metropolis-Hastings sampling and testing mode used. The ISAMPLE_Mx options define how many times to generate and test a sample for goodness-of-fit using a given kernel. ISAMPLE does not refer to a kernel, but defines the number of chains that are maintained, each chain having their own sample generation and testing sequence using the various kernels. Each chain retains a final sample for each subject, at the end of each iteration.

ISAMPLE_M1=*n*

n is the number of mode 1 iterations for the Metropolis-Hasting algorithm for estimating individual parameters using the population means and variances as proposal density. Used with the SAEM and BAYES methods. Default is 2.

ISAMPLE_M1A=*n*(NM72)

n is the number of mode 1A iterations for the Metropolis-Hasting algorithm, testing model parameters from other subjects as possible values. Used with the SAEM and BAYES methods. Default is 0.

ISAMPLE_M1B=*n* (NM74)

n is the number of mode 1B iterations for the Metropolis-Hasting algorithm Default is 2.

ISAMPLE_M2=*n*

n is the number of mode 2 iterations for the Metropolis-Hasting algorithm for estimating individual parameters using the current parameter vector position as mean and a scaled variance of OMEGA as variance. Used with the SAEM and BAYES methods. Default is 2.

ISAMPLE_M3=*n*

n is the number of mode 3 iterations for the Metropolis-Hasting algorithm for estimating individual parameters in which samples are generated for each parameter separately. Used with the SAEM and BAYES methods. Default is 2.

ISCALE_MIN=*x* ISCALE_MAX=*x* (NM72)

In importance sampling, the scale factor used to vary the size of the variance of the proposal density in order to meet the IACCEPT condition is by default bounded by ISCALE_MIN of 0.1, and ISCALE_MAX=10.0. Defaults are (1.0E-06,1.0E+06) for MCMC sampling. These options allow the scale factor boundary to be modified.

KAPPA=*x* (NM74)

Used with NUTS method. Default is 1.

KNUTHSUMOFF=*n*] (NM74)

The Knuth summing method is used to allow the most accurate summation of individual objective function values, even with large variations in values of the individual objective function. To turn this off, and allow a standard summation (not recommended except for comparison purposes from earlier versions), set KNUTHSUMOFF=1. With KNUTHSUM algorithm on by default, the SORT option is not necessary. Default is 0. May also be set with $COVARIANCE record.

LAPLACIAN

Use the Laplacian method, in which second derivatives with respect to eta are used. Laplacian may not be used with METHOD=0. It may be used with the EM/Monte Carlo methods, in which case the Laplacian option will be properly utilized, such as during MAP estimation used during IMP, IMPMAP, and ITS, or ignored, such as during SAEM or BAYES.

Cannot be used with $ABBREVIATED DERIV2=NO unless NUMERICAL option is also specified.

NOLAPLACIAN

Do not use the Laplacian method. This is the default.

LEVCENTER=[0|1] (NM75)

There is no default. Required with $LEVEL and $ESTIMATION. If LEVCENTER=1, this ensures the etas of super ID random levels sum to 0. In earlier versions of NONMEM, this was the default (and only) action. To obtain similar results as earlier versions of NONMEM, set LEVCENTER=1.

If LEVCENTER=0, level etas are not forced to sum to 0.

See **INTRODUCTION TO NONMEM 7** for more information.

LEVWT=*n* (NM74)

This option applies when $LEVEL record is present. By default, LEVWT=0, and weights each level value equally, regardless of number of subjects per level value. To weight according to number of subjects for that value, set LEVWT=1.

LIKELIHOOD

This is designed mainly, but not exclusively, for use with noncontinuous observed responses ("odd-type data"). Indicates that Y (with NM-TRAN abbreviated code) or F (with a user-supplied PRED or ERROR code) will be set to a (conditional) likelihood. Upon simulation it will be ignored, and the DV data item will be set directly to the simulated value in abbreviated or user code. Also etas, if any, are understood to be population etas. Epsilon variables and the $SIGMA record may not be used. The L2 data item may not be used. The CONTR and CCONTR options of the $SUBROUTINES record may not be used. NONMEM cannot obtain the initial estimate for omega. If the data are population, and MAXEVALS=0 is not coded, then NOINTERACTION is required. Compare with PREDICTION option.

-2LOGLIKELIHOOD

Indicates that Y (with NM-TRAN abbreviated code) or F (with a user-supplied PRED or ERROR code) is a -2 log (conditional) likelihood. All remarks for LIKELIHOOD apply. May also be coded -2LLIKELIHOOD. Compare with PREDICTION option.

MADAPT=*n* (NM74)

Used with NUTS method. Default is -1.

MAPCOV=1 (NM74)

Unused. The default is 1.

MAPINTER=*n* (NM72)

Every *n*th iteration, the MAP estimation should be used to provide parameters to the sampling density. Thus, if MAPITER=20 and MAPINTER=5, then for the first 20 iterations, MAP estimation is used, and thereafter, every 5th iteration the MAP estimation is used. If MAPINTER=-1, then mapinter will be turned on only if the objective function increases consistently over several iterations. Setting any of the above parameters to -100 will force NONMEM to select the default value for that parameter.

MAPITER=*n* (NM72)

The first *n* iterations are to use MAP estimation to assess parameters for the sampling density. After these *n* iterations, the conditional means and variances of the pervious iteration are used for the sampling density parameters of the present iteration. If MAPITER=0, then the first iteration will rely on conditional means and variances that are in memory. These may have come from an MSF file, or from a previous estimation step.

MAPITERS=[0|1] (NM75)

By default, no MAP estimation is performed with SAEM or BAYES. To get good individual parameter values near the mode of the posterior density for the first iteration of SAEM, you can set MAPITERS=1. Alternatively, you can insert the record:
$EST METHOD=ITS NITER=0
Followed by
$EST METHOD=SAEM
or
$EST METHOD=BAYES

MASSRESET=*n* (NM74)
Mass matrix information accumulation for NUTS method. Default is -1.

MAXEVALS=*n*
Maximum allowable number of evaluations of the objective function during the Estimation Step. Default: a generous number. (Each evaluation of the objective function requires one pass through the data set. This is also referred to as a "function evaluation.") MAXEVALS=-1 may be specified when a $MSFI record is present. It requests that NONMEM re-use the value from the previous run, and is the default with $MSFI.

MAXEVALS=0 requests that the Estimation Step be omitted. This is useful, for example, with POSTHOC (see above).

MCETA=*n* (NM73)

MCETA=0
Eta=0 is initial setting for MAP estimation (eta optimization). This is the default.

MCETA=1
ETA=values of previous iteration is initial setting for MAP estimation, or ETA=0, whichever gives lower objective function.

MCETA>1
MCETA-1 Random samples of ETA, using normal random distribution with variance OMEGA, are tested. Plus previous ETA is tested, and ETA=0 is tested. Whichever gives the lowest objective function is used as initial setting for the MAP optimization.

METHOD=kind
Values for kind are:

0 or ZERO
Always set etas to 0 during the computation of the objective function. Also called the "first order (FO) method." This is the default.

1 or CONDITIONAL
Use conditional estimates for the etas during the computation of the objective function. METHOD=1 (without the LAPLACIAN option) is also called the "first order conditional estimation (FOCE) method." The conditional estimates of the etas are referred to as Conditional Parametric Etas (CPE).

METH=COND NOLAPLACIAN is referred to as the FOCE method.
METH=COND LAPLACE is referred to as the Laplace method.
METH=COND NOLAPLACE CENTERING is referred to as the Centering FOCE method.
METH=COND LAPLACE CENTERING is referred to as the Centering Laplace method.

HYBRID
Use conditional estimates for the etas during the computation of the objective function, with the exception of those etas listed in the ZERO option. Cannot be used with LAPLACIAN or CENTERING.

The following methods are new to NONMEM 7. When any of these methods are used, the data are inferred to be population, and METHOD=1 is supplied if it is not already present. The first four methods are referred to as EM (Expectation-Maximization) Methods.

ITS
Use the iterative two-stage method. This method evaluates the conditional

mode and first order approximation of the conditional variance of parameters of individuals by maximizing the posterior density. This integration step is the same as used in the FOCE method. Population parameters are updated from individuals' conditional mode parameters and their approximate variances by single iteration maximization steps.

`IMP`
: Use the Monte-Carlo Importance Sampling Expectation Maximization method. This method evaluates the conditional mean and variance of parameters of individuals by Monte Carlo sampling. It uses the posterior density, which incorporates the likelihood of parameters relative to population means and variances with the individual's observed data. The normal density near the mean or mode of the posterior is used as a proposal density, then weighted according to the posterior density as a correction.

`IMPMAP`
: Use the Importance Sampling method assisted by Mode a Posteriori (MAP) estimation. At each iteration, conditional modes and conditional first order variances are evaluated as in the ITS or FOCE method. These are then used as parameters to the multivariate normal proposal density for the Monte-Carlo importance sampling step.

`SAEM`
: Use the Stochastic Approximation Expectation Maximization method. As in importance sampling, random samples are generated from normal proposal densities. However, instead of always being centered at the mean or mode of the posterior density, the proposal density is centered at the previous sample position.

`BAYES`
: Use the Markov Chain Monte Carlo (MCMC) Bayesian Analysis method. The goal of the MCMC Bayesian analysis is to obtain a large sample set of probable population parameters. Various summary statistics of the population parameters may then be obtained such as means and confidence ranges.

`DIRECT`
: Requests Monte Carlo Direct Sampling. Creates completely independent samples (unlike MCMC), and there is no chance of causing bias if the sampling density is not similar enough to the conditional density (unlike IMP). However, it is very inefficient, requiring ISAMPLE settings of 10000 to 300000 to properly estimate the problem.

`NUTS` (NM74)
: Requests No U-Turn Sampling (NUTS) Markov Chain Monte Carlo (MCMC) Bayesian Analysis Method. Options unique to this method are listed alphabetically under `NUTS_...`. Other options of interest with their defaults are as follows:

```
MASSRESET=-1
MADAPT=-1
KAPPA=1
TTDF=0
OLKJDF=0
OVARF=1
SLKJDF=0
SVARF=1
```

CHAIN
: Allows the user to create a series of random initial values of THETAs and OMEGA's, or for reading in initial population parameters from a file of rectangular (rows/column) format. Applies only to the Estimation Step.

LNTWOPI (NM74)
: The objective function is reported including the N*LOG(2pi) constant term, where N is the total number of normally distributed data values in the data set. Compare OLNTWOPI. Either or both may be used.

MSFO=filename
: A Model Specification File is output to a file with the given filename. Filename may not contain embedded spaces. If filename contains commas, semicolons, or parentheses, then it must be surrounded by quotes (' or "). Filename may also contain equal signs if it is enclosed in quotes. Filename may contain at most 71 characters. If filename is the same as any option of the $ESTIMATION record, it must be enclosed in quotes. If the $NONPARAMETRIC record is present and also specifies the MSFO option, the filename is required on the record which appears first in the control stream. If filename is present on both, it must be the same. If the filename is omitted on the second of the two records, the MSF option must be the final option on that record. Default: If the MSFO option is not used, no MSF is output.

  If a MSFO is output, then the iteration estimates may also be seen in the original parameterization for those iterations whose summaries appear in intermediate printout. These estimates may be found in file INTER.

  When MAXEVAL=0 and the Covariance Step is implemented, the MSFO option may also be used, and then a model specification file will be output which will include information from the Covariance Step and from the input model specification file concerning the earlier Estimation Step (in this case there must be an input model specification file).
  (See **model_specification_file**).

MUM=*s*
: *s* is a string of [M|N|D|X]'s with each letter representing a THETA parameter in numerical order. M indicates that the THETA should be Mu modeled. N indicates the THETA should not be Mu modeled. D indicates the program will decide if the parameter should be Mu modeled or not. X indicates that THETA is involved in a covariate-dependent mixture model and is required if this is the case.
  Default is DDDD...

MUM=v1(n1):v2(n2):v3(n3)...
: An alternative syntax may be used. V is a letter (N,M,D, or X), and n is a number list. For example, to specify that thetas 3 and 5 through 8 should not be MU modeled, theta 2 is a population mixture parameter, and thetas 6 and 12 are to be MU modeled:
  MUM=N(3,5-8):X(2):M(6,12)

NBURN=*n*
: When used with the SAEM method *n* is the maximum number of iterations used to perform the stochastic phase. Default is 2000. When used with the BAYES method *n* is the maximum number of iterations used to perform the burn-in phase. Default is 4000.

NITER=*n*
: When used with the ITS, IMP, and IMPMAP methods *n* is the maximum number of

iterations. Default is 50. When used with the SAEM method n5 is the number of iterations for the non-stochastic accumulation phase. Default is 1000. When used with the BAYES method n5 is the number of iterations used to obtain the stationary distribution. Default is 10000. NITER may also be coded NSAMPLE.

NOCOV=[0|1] (NM73)

If covariance estimation is not desired for a particular estimation step, set NOCOV=1. It may be turned on again for the next estimation step with NOCOV=0. If NOCOV=1 is set for an FOCE/Laplace/FO step, this is equivalent to $COV NOFCOV setting.

For ITS and IMP, covariance estimation can take some time for large problems, and you may wish to obtain only the objective function, such as in the case of $EST METHOD=IMP EONLY=1 after an SAEM estimation. NOCOV has no effect on BAYES analysis, as no extra time is required in assessing covariance for BAYES.

By default, standard error information for the classical methods (FO/FOCE/Laplace) will be given only if they are the last estimation method, even if NOCOV=0 for an intermediate estimation step. If NOCOV=1 for the FOCE/LAPLACE/FO method, and it is the last estimation step, then standard error assessment for it will be turned off.

NOLABEL=[0|1]

1 indicates that the row of item names for FILE will not be written, otherwise 0, the default. Affects the raw output file and all additional output files.

NONINFETA=[0|1] (NM73)

Sometimes, gradients are not properly evaluated for classical NONMEM methods, when not all etas are used for all subjects. For example, an eta to a ka absorption rate constant during a fit of a subject with only IV dosing would be considered a non-influential data. If $EST METHOD=1 or 0 is used without the SLOW option, this can result in evaluating very large and incorrect gradients, which in turn affects the search path, and sometimes the final objective function value. Should this occur, add NONINFETA=1 to the $EST record. NONINFETA=0 is the default.

NOPRIOR=[0|1]

If prior information was specified using the $PRIOR statement, then normally the analysis is set up for three stage hierarchical analysis. By default NOPRIOR=0, and this prior information will be used. If NOPRIOR=1, then for the particular estimation, the prior information is not included in the analysis. This is useful if you do not want to use prior information during a maximization (METHOD=IMP, CONDITIONAL, IMPMAP, SAEM, or ITS), but then use it for the Bayesian analysis (METHOD=BAYES). With NOPRIOR=1, FOCE is still allowed to evaluate an S MATRIX, since prior information is not used. I.e., $EST NOPRIOR=1 and $COV MATRIX=S are permitted. With NONMEM 7.3, when NOPRIOR=1 is set, the estimation will not use TNPRI prior information (TNPRI should only be used with FO/FOCE/Laplace estimations). In previous versions of NONMEM, NOPRIOR=1 did not act on TNPRI priors.

NOSUB=[0|1] (NM74)

With NOSUB=0, label substitution will be performed for final estimates in the NONMEM report file. (See **$ABBREVIATED**). This is the default. With NOSUB=1, label substitution will not be performed.

NOTITLE=[0|1]

1 indicates the header for FILE will not be written, otherwise 0, the default. Affects the raw output file and all additional output files.

NUMDER=[0|1|2|3] (NM73)
With NUMDER=1, NONMEM computes and displays numerically evaluated derivatives of Y or F with respect to eta and eps (G and H). These numeric values are displayed in root.fgh, but are not used in estimation.
With NUMDER=0, file root.fgh is not produced. This is the default.
With NUMDER=2, analytical derivatives values are stored in root.agh
With NUMDER=3, both root.agh and root.fgh are produced.

NUMERICAL
Requests that second eta-derivatives for the Laplacian method be obtained numerically.

NONUMERICAL
Requests that second eta-derivatives for the Laplacian method be computed by PRED. Not permitted with the combination LAPLACIAN and INTERACTION. Otherwise, this is the default.

NUTS_BASE (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 0.025.

NUTS_DELTA (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 0.8

NUTS_EPARAM (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 0

NUTS_GAMMA (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 0.05

NUTS_INIT (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 0.075.

NUTS_MASS (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is B

NUTS_MAXDEPTH (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 10

NUTS_OPARAM (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 1

NUTS_REG (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 0.0

NUTS_SPARAM (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 1

NUTS_STEPINTER (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 0

NUTS_STEPITER (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 1

NUTS_TERM (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 0.05

NUTS_TEST (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 0

NUTS_TRANSFORM (NM74)
See Guide INTRODUCTION TO NONMEM 7. Default is 0

OACCEPT=*n*

Used only with the BAYES method. *n* has meaning only for OMEGA sampled by the Metropolis-Hastings algorithm. The scaling of degrees of freedom is adjusted so that samples are accepted *n* fraction of the time. See OSAMPLE_M2=. Default is 0.5.

OLKJDF=*n* (NM74)

Used with NUTS method, OLKJDF stands for Omega LKJ density degrees of freedom. When 0, the usual inverse Wishart prior is used for Omegas. When OLKJDF>0, then the LKJ density is used as the prior, with OLKJDF degrees of freedom for all omega blocks. In addition, only diagonal elements of the OMEGA prior are used, assuming a density dependent on the OVARF value. OLKJDF may be set >0 when using METHOD=BAYES as well, but thetas will then be M-H sampled using the OSAMPLE_M1, OSAMPLE_M2, and OSAMPLE_M3 settings.
Default is 0.
See record $OLKJDF (NM75) to specify LKJ correlation degrees of freedom for each omega block.

OLNTWOPI (NM74)

The objective function is reported including the NETA*NIND*LOG(2pi) constant term for SAEM and BAYES, where NETA is the number of etas, and NIND is number of individuals. Compare LNTWOPI. Either or both may be used.

THETABOUNDTEST, OMEGABOUNDTEST, SIGMABOUNDTEST

With NONMEM VI, the estimation step sometimes terminates with the message
PARAMETER ESTIMATE IS NEAR ITS DEFAULT BOUNDARY.
These options request that the "default boundary test" be performed for THETA, OMEGA, and SIGMA, respectively. THETABOUNDTEST may also be coded TBT or TBOUNDTEST; OMEGABOUNDTEST may also be coded OBT or OBOUNDTEST; SIGMABOUNDTEST may also be coded SBT or SBOUNDTEST. These options are the defaults.

NOTHETABOUNDTEST, NOOMEGABOUNDTEST, NOSIGMABOUNDTEST

Instructs NONMEM to omit the "default boundary test" for this type of variable, i.e., to behave like NONMEM V in this regard. Any option listed above may be preceded by "NO". The THETA, OMEGA, and SIGMA choices are independent of each other. E.g., it is possible to specify NOOBT (to prevent the "default OMEGA boundary test") and permit both the "default THETA boundary test" and "default SIGMA boundary test".

OMITTED

The Estimation Step is not implemented.

OPTMAP=*n* (NM73)

For alternative MAP (eta optimization) methods. With OPTMAP>0, SLOW option may be needed.

OPTMAP=0

Standard variable metric (Broyden, Fletcher, Goldfarb, and Shanno (BFGS)) optimization method used by NONMEM to find optimal eta values (referred to as eta hat) for each subject at the mode of their posterior densities, using analytical derivatives of F with respect to etas (G), and analytical derivatives of F with respect to etas (H), that were supplied by NMTRAN or by the user. This is the default.

OPTMAP=1

Variable metric method using numerical finite difference methods for first

derivatives of F with respect to etas. Necessary when not all code used in evaluating F, G and H for observation event records is abbreviated code (some may be in verbatim code), and/or some portions of the computation of F, G and H are evaluated in a hidden subroutine specified by "$SUBROUTINES OTHER=" and the user-written code does not compute the eta derivatives. When OPTMAP=1 is present, values of G and H are ignored during eta optimization. This may be used to test user-coded deriatives, because two runs, one with OPTMAP=1 and one without it, should give very similar values for the OBJV, WRES, etc. if the user-coded derivatives are correct.

`OPTMAP=2`

Nelder Mead method, which uses a secant method, rather than relying on derivatives.

`ORDER=xxxf`

The values of x may be T (Theta), S (Sigma), and O (Order). The value of f may be U (Upper) or L (Lower). Affects the way theta, omega, and sigma are displayed in the raw and additional output files. xxx gives the overall order, and f gives the order within OMEGA and SIGMA. Affects the raw output file and all additional output files. The default is TSOL: THETA, SIGMA, OMEGA in Lower triangular form. Does not affect the NONMEM report file.
(See **order_option**).

`OSAMPLE_M1=`*n*

Used only with the `BAYES` method. *n* is the number of times each iteration that OMEGA is generated using the Metropolis-Hastings algorithm and a Wishart proposal density that has variance based on the previous samples. If $n < 0$ this indicates that the OMEGA's are Gibbs sampled using the appropriate Wishart proposal density and other options are not relevant. Default is -1.

`OSAMPLE_M2=`*n*

Used only with the `BAYES` method. *n* has meaning only for OMEGA sampled by the Metropolis-Hastings algorithm. *n* is the number of times that OMEGA is generated using a Wishart proposal density at the present OMEGA position and degrees of freedom scaled to have samples accepted a particular fraction of the time. If $n < 0$ this is done as many times as there are non-fixed OMEGA elements. Default is -1.

`OVARF=`*x* (NM74)

Used with NUTS method and OLKJDF option. OVARF is the weight factor to STD prior to the log sqrt OMEGA diagonal elements, the normal density of the log square root of OMEGA centered about log square root of Omega prior, and scaled with OVARF (see below). That is, log(sqrt(Omega(i))) ~ Normal(log(sqrt(OmegaPrior(i))),1/OVARF). If OVARF<0, then a half-t-distribution of degrees of ABS(OVARF) is used as the prior to the sqrt of OMEGA diagonal elements. Use OVARF=-1 for the half-Cauchy distribution.
Default is 1. See also $OVARF control record.

`PACCEPT=`*n*

Used only with the `BAYES` method. *n* has meaning only for population parameters sampled by the Metropolis-Hastings algorithm. The scaling of variance is adjusted so that samples are accepted *n* fraction of the time. See `PSAMPLE_M2=`. Default is 0.5.

`PARAFILE=`filename

Name of the "parallel file" (the parallelization profile) that controls parallelization

(distributed computing). Default file name if not specified: parallel.pnm or parafile name specified on nmfe command.
PARAFILE=ON turns on parallelization for this $ESTIMATION record. PARAFILE=OFF turns off parallelization for this $ESTIMATION record.

PARAFPRINT=*n* (NM74)
The print iteration intervals to the parallelization log file can be controlled by this option during parallelization of the $EST step. See also $COVARIANCE record and nmfe74 command. Default is PARAFPRINT=1.

PHYTYPE=*n* (NM74)
Default is 0. By default, after an estimation is performed, the phi(), conditional means of the individual parameters, and their variances, are reported in the root.phi file, where root is the root name of the control stream file. If you wish to have conditional mean etas reported, set PHITYPE=1.
See "Stochastic Approximation Expectation Maximization (SAEM) Method" in Guide INTRODUCTION TO NONMEM 7.

POSTHOC
This option may be used when the FO method is used. After the Estimation Step terminates, the eta values are estimated for each individual. To estimate the etas based on the initial estimates of THETA, OMEGA, and SIGMA (found either in the control stream or in a model specification file), also specify MAXEVAL=0 (which omits the Estimation Step). The conditional estimates of the etas are referred to as Conditional Parametric Etas (CPE).

NOPOSTHOC
Etas are not estimated. This is the default with METHOD=0. May not be used with METHOD=1.

PREDICTION
Indicates that Y (with NM-TRAN abbreviated code) or F (with a user-supplied PRED or ERROR code) will serve as a prediction variable, i.e., it will be set to a prediction. Upon simulation, the simulated observation is possibly also being set in Y or F. (However, the DV data item may instead be set directly to the simulated observation.) Also, etas (if any) are population etas only if epsilons also appear. This is the default. Compare with LIKELIHOOD, -2LOGLIKELIHOOD options.

PRINT=*n*
Iteration summaries are printed for the 0th, every *n*th iteration, and last iteration. When *n*=0, no summaries are printed. Default: 9999 (so that summaries are printed for 0th and last iterations).

PRIORC (NM74)
The objective function is reported including the prior constant term (constant term to the prior).

PSAMPLE_M1=*n*
Used only with the BAYES method. *n* has meaning only for population parameters sampled by the Metropolis-Hastings algorithm. *n* is the number of times that a vector of THETA's and SIGMA's are generated using a multivariate normal proposal density that has mean/variances based on the previous samples. Default is 1.

PSAMPLE_M2=*n*
Used only with the BAYES method. *n* has meaning only for population parameters sampled by the Metropolis-Hastings algorithm. *n* is the number of times that a vector of THETA's and SIGMA's are generated using a multivariate normal proposal

density that has mean at the present parameter position and variance scaled to have samples accepted a particular fraction of the time. If $n < 0$ this is done as many times as there are Metropolis-Hastings parameters. Default is -1.

PSAMPLE_M3=*n*

Used only with the BAYES method. *n* has meaning only for population parameters sampled by the Metropolis-Hastings algorithm. *n* is the number of times in mode 3 that each parameter is individually sampled. Default is 1.

PSCALE_MIN=*n*, PSCALE_MAX=*n* (NM74)

In MCMC sampling, the scale factor used to vary the size of the variance of the proposal density population parameters (theta/sigma) that are not Gibbs sampled, in order to meet the PACCEPT condition, is by default bounded by PSCALE_MIN of 0.01, and PSCALE_MAX=1000. This should left alone for MCMC sampling, but on occasion there may be a reason to expand the boundaries (perhaps to PSCALE_MIN=1.0E-06, PSCALE_MAX=1.0E+06).

RANMETHOD=[n|S|m|P] (NM72)

See **INTRODUCTION TO NONMEM 7, Reference [5] and [7].**

n: the random number generator used for all Monte Carlo EM and Bayesian methods.

0: ran0 of reference [5], minimal standard generator
1: ran1 of reference [5], Bays and Durham.
2: ran2 of reference [5].
3: ran3 of reference [5], Knuth. (Default)
4: NONMEM's traditional random number generator used in $SIMULATION

S: sobol method without scrambling, used during importance or direct sampling (methods IMP, IMPMAP, and DIRECT) and only for the purpose of creating quasi-random samples of eta vectors. As of NONMEM 7.3, Sobol may be used for BAYES and SAEM methods as well.

m: the type of scrambing desired

0: no scrambing (S0 is the same as S)
1: Owen type scrambling
2: Faure-Tezuka type scrambling
3: Owen plus Faure-Tezuka type scrambling.

P: each subject will receive its own seed path, that will stay with that subject regardless of whether the job is run as a single process or parallel process. (NM74)

The RANMETHOD specification propagates to subsequent $EST records in a given problem, but does not propagate to $CHAIN or $TABLE records.

REPEAT

The search is repeated with the initial estimates being the final estimates from the first search and with new UCP, so that a UCP value of 0.1 now corresponds to a final estimate from the first search. Cannot be used with STIELTJES.

NOREPEAT

The estimate obtained at the end of the minimization search is taken to be the final parameter estimate. This is the default. Cannot be used with STIELTJES.

REPEAT1

The search of the first stage of the Stieltjes method is repeated with the initial estimates being the final estimates from the first search and with new UCP, so that a UCP value of 0.1 now corresponds to a final estimate from the first search. May only be used with STIELTJES.

NOREPEAT1

The estimate obtained at the end of the search of the first stage of the Stieltjes method is taken to be the final parameter estimate at the first stage. This is the default. May only be used with STIELTJES.

REPEAT2

The search of the second stage of the Stieltjes method is repeated with the initial estimates being the final estimates from the first search and with new UCP, so that a UCP value of 0.1 now corresponds to a final estimate from the first search. May only be used with STIELTJES.

NOREPEAT2

The estimate obtained at the end of the search of the second stage of the Stieltjes method is taken to be the final parameter estimate at the second stage. This is the default. May only be used with STIELTJES.

SADDLE_HESS=*n* (NM74)

SADDLE_HESS=0 selects the Hessian matrix last generated by the variable metric method. SADDLE_HESS=1 causes the full second derivative information matrix (identical to R matrix in the $COV step) to be evaluated. Default is 0.

See "Resetting the Search to Circumnavigate Saddle Points and Detect Inestimable Parameters" in Guide INTRODUCTION TO NONMEM 7 for a discussion of SADDLE_HESS and SADDLE_RESET options.

SADDLE_RESET=*n* (NM74)

SADDLE_RESET is the number of times that a reset should occur in the course of the search. Normally, should be set to 1. Default is 0.

SEED=*n*

The initial seed for the random number generator used for the Monte-Carlo methods. Default is 14455.

CLOCKSEED=[0|1] (NM75)

If CLOCKSEED=1 (default is 0), actual starting seed will be 10000*(seconds after midnight)+SEED. This allows a control stream to produce different stochastic results for automated replications, without the need to modify the seed value in the control stream file in each replication.

SELECT=[0|1|2|3] (NM73)

Used with METHOD=CHAIN and $CHAIN to specify how the sample is selected.

SELECT=0

If ISAMPEND>=ISAMPLE, then the default action for selecting between ISAMPLE and ISAMPEND is taken, which for $EST METHOD=CHAIN is to find the one giving the best OBJ at the initial values, and for $CHAIN is to randomly select a sample, with replacement. This is the default.

SELECT=1

The sample is selected sequentially from ISAMPLE to ISAMPEND with each new use of $CHAIN/$SIML with multiple sub-problems for the given problem, and with each new $EST METHOD=CHAIN with multiple sub-problems and across problems. When ISAMPEND is reached, the sample selection begins at ISAMPLE again.

SELECT=2

Uniform random selection of sample, without replacement. Should the sample selection become exhausted, which would occur if CHAIN or $CHAIN records are utilized for more than ISAMPEND-ISAMPLE+1 times, subsequent sample selection

then occurs with replacement.
SELECT=3
Uniform random selection of sample, with replacement (this is equivalent to SELECT=0 for $CHAIN).

SIGDIGITS=*n*
Number of significant digits required in the final parameter estimate. SIGDIGITS is not used by the Monte-Carlo methods. Default: 3. May also be coded NSIGDIGITS.

SLKJDF=*n* (NM74)
Used with NUTS method, SLKJDF stands for Sigma LKJ density degrees of freedom. When 0, the usual inverse Wishart prior is used for Sigmas. When SLKJDF>0, then the LKJ density is used as the prior, with SLKJDF degrees of freedom. In addition, only diagonal elements of the Sigma prior are used. SLKJDF may be set >0 when using METHOD=BAYES as well, but Sigmas (in cholesky format) will then be M-H sampled using the PSAMPLE_M1, PSAMPLE_M2, and PSAMPLE_M3 settings (choleskys of sigma elements are treated as extensions of the THETA parameters in M-H sampling methods).
Default is 0.
See record $SLKJDF (NM75) to specify LKJ correlation degrees of freedom for each sigma block.

SIGL=*n*
n is used to calculate the step-size for finite difference derivatives independent of the SIGDIGITS value. If *n*=0 or *n*=100 then SIGL is ignored and SIGDIGITS is used as in versions prior to NONMEM 7. SIGL should usually be 2 to 3 times the value of NSIG. It is not used by the SAEM or BAYES methods.

SIGLO=*n* (NM72)
The precision to which the individual etas are optimized. The SIGL value set by the user continues to be the precision (or delta) setting for the finite difference algorithms in the higher level estimation process for THETAS, OMEGAS, and SIGMAS. By default, if SIGLO is not specified, then SIGLO is set to the same value as SIGL. Should SIGLO be used, the recommended setting would be:
SIGLO<=TOL
SIGL<=SIGLO
NSIG<=SIGL/3

SIGMABOUNDTEST
See OMEGABOUNDTEST.

SLOW
Requests a slower method of computation. Required when either a mixture model is used along with CENTERING, or NUMERICAL is used. If not present, the option is automatically supplied in these two cases. For problems where NONMEM VI does not behave as well (e.g. yields a higher OFV at termination) compared to NONMEM V, inclusion of the SLOW option may sometimes, but not always, yield NONMEM VI results that are similar to NONMEM V.

SLOW=1
Same as SLOW.

NOSLOW
Requests a faster method of computation. This is the default (but see SLOW)

SLOW=2
This option is permitted with STIELTJES.

SORT
Individual contribution to the objective function value and individual contributions to the gradients are sorted before they are summed, so that smaller numbers are summed before larger numbers.

NOSORT
Individual contribution to the objective function value and individual contributions to the gradients are summed in the order in which the individual records appear in the NONMEM data set, as was done prior to NONMEM VI. This is the default.

STDOBJ=*x* (NM74)
For importance sampling and direct sampling only, if ISAMPEND is specified as an upper integer value, and STDOBJ is set to a real value greater than 0, then NONMEM will vary the number of Monte Carlo samples under each subject between ISAMPLE and ISAMPEND, until the stochastic standard deviation of the objective function falls below STDOBJ.

STIELTJES
A set of tentative population estimates are first obtained using some 1st- or 2nd-order method. A tentative value for the integral (i.e. an area) is obtained. Then numerical integration is used to obtain second-stage estimates. See the Introduction to Version VI 2.0. Not permitted with METHOD=HYBRID.

SVARF=*x* (NM74)
Used with NUTS method and SLKJDF option. SVARF is the weight factor to STD prior to the log sqrt SIGMA diagonal elements, the normal density of the log square root of SIGMA centered about log square root of SIGMA prior, and scaled with SVARF (see below). That is, log(sqrt(Sigma(i))) ~ Normal(log(sqrt(SigmaPrior(i))),1/SVARF). If SVARF<0, then a half-t-distribution of degrees of ABS(SVARF) is used as the prior to the sqrt of SIGMA diagonal elements. Use SVARF=-1 for the half-Cauchy distribution.
Default is 1.
See also $SVARF control record.

TBLN=*n* (NM75)
Used with $EST METHOD=CHAIN and $CHAIN records to allow selecting a table within a raw output file. See "Method for creating several instances for a problem starting at different randomized initial positions" in Guide INTRODUCTION TO NONMEM 7.

THETABOUNDTEST
See OMEGABOUNDTEST.

THIN=*n* (NM74)
The Bayesian records retained in the raw output file may be adjusted by every THINth iteration. So, if THIN=10, then every 10th iteration is recorded in the raw output file. The PRINT option controls only the iterations printed to the console and NONMEM report file.
Default is 1.

TPU=*n* (NM75)
If TPU>0, use THETA_PRIORU routine in ..\source\THETA_PRIORU.f90. You can make a copy of THETA_PRIORU.f90, modify it, call it USERPRIORT.f90, for example, then specify it as an OTHER routine: $SUBR OTHER=USERPRIORT.f90.

See also \$OLKJDF and \$SLKJDF for OMEGA and SIMGA.
Default is 0.

`TTDF`=*n* (NM74)

TTDF stands for Theta t-density degrees of freedom. It is used with NUTS method. Default is 0. When 0, the usual normal density prior is used as a prior density for thetas. When TTDF>0, a t-distributed prior is used. TTDF may be set >0 when using METHOD=BAYES as well, but thetas will then be M-H sampled using the PSAMPLE_M1, PSAMPLE_M2, and PSAMPLE_M3 settings. TTDF may be a real number.
See also $TTDF control record (NM75) to specify degrees of freedom for each theta. The value of TTDF overrides $TTDF.

`ZERO`=list

Required with `METHOD=HYBRID`. A list of indices for etas which are fixed to zero during the Estimation Step. "list" contains one or more integers. If more than one, they must be surrounded by parentheses. The list must be contained entirely on one line. The indices may be separated by commas or spaces.

Reserved Variables that are of Interest During the Estimation Step

MUFIRSTREC (NM74)

The MUFIRSTREC option can speed up the NUTS method, and also ordinary BAYES, FAST FOCE, ITS, and the EM methods. Set MUFIRSTREC=1 in $PRED or $PK. MUFIRSTREC=1 selects the covariate of the first record of the subject, rather than averaging among its records when using that covariate in a MU reference.
The first statement of the $PRED or $PK block should be
`include nonmem_reserved_general`.

OBJQUICK (NM74)

The OBJQUICK option can speed up the NUTS method, and also ordinary BAYES, FAST FOCE, ITS, and the EM methods. Set OBJQUICK in $PRED or $PK.

OBJQUICK=0
: Default. Standard NONMEN processing of the model occurs.

OBJQUICK=1
: Certain tests and initializations are skipped.

OBJQUICK=2
: A simplified modeling process occurs, but which cannot be used when $LEVEL or $MIX is used in the model.

The first statement of the $PRED or $PK block should be
`include nonmem_reserved_general`.

REFERENCES: Guide I, section C.3.5.1
REFERENCES: Guide II, section E , F
REFERENCES: Guide IV, section III.B.14 , IV.G
REFERENCES: Guide V, section 9.4.1, , 13.2
REFERENCES: Guide VII, section I , II , III
REFERENCES: Guide Introduction_7

$ETAS,$PHIS

MEANING: Specifies Initial Values for Etas or Phis
CONTEXT: NM-TRAN Control Record

USAGE:

`$ETAS` [[*value*$_1$ [*value*$_2$] [*value*$_3$] ... [*value*$_n$]]
[`FILE=`filename [`FORMAT`|`DELIM =`*s*] [`TBLN=`*n*]]

`$PHIS` [[*value*$_1$ [*value*$_2$] [*value*$_3$] ... [*value*$_n$]]
[`FILE=`filename [`FORMAT`|`DELIM=`*s*1] [`TBLN=`*n*]]

SAMPLE:

```
$ETAS 0.4 3.0 3.0 5.0
$PHIS 0.4 3.0 3.0 5.0
$ETAS FILE=myprevious.phi FORMAT=s1pE15.8 TBLN=3
$PHIS FILE=myprevious.phi FORMAT=s1pE15.8 TBLN=3
```

NONMEM describes the use of these records with messages in the report file such as the following:

```
LOADED    3 PHI/ETA ITEMS FROM CONTROL STREAM
```

DISCUSSION:

By default, the initial value used for ETA's in the Estimation Step search is 0. The $ETAS and $PHIS records provide different initial estimates. Optional.

OPTIONS:

There are two forms:

`$ETAS` *value*$_1$ [*value*$_2$] [*value*$_3$] ...[[*value*$_n$]

All of the subjects in the data set will be given these initial values of etas. If fewer values are listed than the number of etas in the problem, the value 0 will be used for the remaining etas. Any real value (positive, negative, zero) may be specified.

When the record is $PHIS, values are entered as phi values, convenient for EM methods. The eta values will then be evaluated as $eta(i) = phi(i) - mu(i)$ for each eta, where $mu(i) = mu\_i$ is evaluated according to their definitions in the $PK section.

`FILE=`filename `FORMAT=`*s* `TBLN=`*n*

Uses initial etas and/or phis for an entire set of subjects from a file .phi or .phm (in the case of mixture problems) of a previous analysis.

With FORMAT, *s* defines the delimiter [,|s(pace)|t(ab)] followed by a Fortran format specification. The default is s1PE12.5. FORMAT should at least have the delimiter appropriate to read the file. May also be coded DELIM.
For more details, see the format help item:
(See **format**).
See **INTRODUCTION TO NONMEM 7, FORMAT=s1PE11.4**

TBLN is the table number in the file. If TBLN is not specified, it defaults to 1, i.e., the first set of etas/phis are brought in.

In matching the etas/phis to the data set given in $DATA of the control stream file, the attempt will be to match ID numbers rather than subject numbers, if an ID column in the file exists, which it will, if you are using a .phi or .phm file generated

from a previous nonmem analysis. The phc/etc variances will also be brought in.

One purpose to bringing initial eta/phi and etc/phc values is you can readily resume an analysis, if an MSF file was not set up in the previous analysis (the MSF file system is still the most complete information transfer for resuming an analysis.

FURTHER DISUCSSION

The etas from $ETAS/$PHIS can be used in several ways.

In METHOD=0 (FO), they are ignored, because FO is specifically designed as a first order Taylor series approximation process centered around eta=0.

In METHOD=1 (FOCE), they are ignored unless MCETA>0. When MCETA>0, then various starting etas are tested, including a set from $ETAS, when available.

In Bayes, SAEM, IMP MAPITER=0 they are used as the starting etas.

In MAP estimation methods, such as METHOD=1, or ITS, or IMP MAPITER>0, or IMPMAP, and if MCETA>0, then these etas are one of the initial eta vector positions tested (during the first iteration), and the one giving the lowest OBJ is then selected.

In cases where FNLETA=2, the estimation step is skipped, and etas inputted from $ETAS are treated as if they were the final result of an estimation.

For examples and discussion:

See **INTRODUCTION TO NONMEM 7, $ETAS and $PHIS Record For Inputting Specific Eta or Phi values**

REFERENCES: Guide Introduction_7

$FORMAT

MEANING: Specifies significant digits for the NONMEM report file
CONTEXT: NM-TRAN Control Record

USAGE:
`$FORMAT FMTN=`*n* (NM75)

$FORMAT FMTN=3 (default) (NM75)

You may now display any significant digits for thetas, omegas, and sigmas results, and variance- covariance of estimates, listed in the NONMEM report file. The FMTN is the number of significant digits (which is by legacy and default 3), between 3 and 23. This also applies to $TABLE outputs to the NONMEM report file. The FORTRAN format will be formed as 1PE{FMTN+6}.{FMTN-1}. For example FMTN=6 will be 1PE12.5. If you wish G field format, set the FMTN to a negative value. For example FMTN=-4 will be 1PG10.3.

It is recommended that you place the $FORMAT record immediately after the $PROB record:

```
$PROB My problem
$FORMAT FMTN=5
```

This FMTN precision format will be in effect for outputs of all problems, until another $FORMAT record is given for a new $PROB in the control stream.

REFERENCES: Guide Introduction_7

$INDEX,$INDXS

MEANING: Defines values for the PRED/PREDPP INDXS array
CONTEXT: NM-TRAN Control Record

USAGE:
`$INDEX` [$label_1|value_1$] [$label_2|value_2$] [$label_3|value_3$] ...

SAMPLE:
```
$INDEX   DOSE  1
```

DISCUSSION:
Optional. Used only with user-supplied FORTRAN subroutines (ERROR, PK, INFN, PRED) which make explicit use of the INDXS array. May also be coded `$INDXS` or `$INDEXES`.

OPTIONS:

The labels are those of data items on the $INPUT record. The values are integers no larger than the number of data items in the NONMEM input data set. Either the index of the data items with $label_i$ or $value_i$ is stored in INDXS(i). (The index of a data item is its position in the NONMEM data record.)

REFERENCES: Guide I, section C.4.1
REFERENCES: Guide IV, section III.B.3
REFERENCES: Guide VI, section III.C , V.A

$INFN

MEANING: Marks the beginning of abbreviated code for the INFN routine
CONTEXT: NM-TRAN Control Record

USAGE:
```
$INFN
```
abbreviated code

SAMPLE:

```
$INFN
IF (ICALL.EQ.1) CALL SUPP (1,1)
```

DISCUSSION:
The $INFN record is used to describe initialization processing for a NONMEM run, or NONMEM problem, or finalization processing for a NONMEM problem. It is used with PREDPP.

General rules for abbreviated code are documented elsewhere (See **abbreviated code**). Specific rules for $INFN blocks are described elsewhere.
(See **Initialization-Finalization block**, **Finalization example**).

RECORD ORDER:

Follows $SUBROUTINES.

REFERENCES: None.

$INPUT

MEANING: Defines the data item types in the data set
CONTEXT: NM-TRAN Control Record

USAGE:
`$INPUT` *item*$_1$ *item*$_2$ *item*$_3$ $\cdots$

SAMPLE:
```
$INPUT       ID DOSE TIME CP=DV WT
```

DISCUSSION:
The items define the data item types that appear in the NM-TRAN data records, and define the order of their appearance.

This record is required. It must precede any other NM-TRAN control record that refers to specific data item types.

With NONMEM VI 2.0 and later versions, the length of a single $INPUT record (and all records of the NM-TRAN input file) is at most 160 characters. (Previously, it was 80 characters.) With NONMEM 7.3, the maximum length is given by FSD in resource/SIZES.f90 (FSD=67000 with NONMEM 7.3). With NONMEM 7.2 and higher, both lower and upper case may be used for all user-defined and reserved words in the control stream. With NONMEM 7.3 and higher, & may be used at the end of any line of the control stream to indicate that the line is to be continued, including control records as well as abbreviated code.

Multiple $INPUT records may be used. Each continues the previous record.

OPTIONS:

Each item has form B or A=B, where A and B are data item labels.

With previous versions of NONMEM labels were restricted to 4 characters in length and could not include the character _. With NONMEM 7 a label consists of 1-24 letters (A-Z), numerals (0-9), and the character '_', beginning with a letter. (The length 24 is specified by constant SDF in SIZES)
(See **SIZES**).

The labels may be used in subsequent NM-TRAN control records and will be used as labels for data items in NONMEM output.

Certain data item labels are *reserved* and refer to data items that may be needed by NONMEM or PREDPP, i.e.,

```
ID L1 L2 DV MDV
RAW_ MRG_ RPT_
TIME DATE DAT1 DAT2 DAT3 DROP SKIP EVID AMT RATE SS II ADDL
CMT PCMT CALL CONT
```

Certain data item labels are semi-reserved, in that they have reserved meanings if used in $INPUT, but can also be user-defined in abbreviated code, in which case they have no reserved meaning, i.e.,

```
XVID1 XVID2 XVID3 XVID4 XVID5
```
(See **evid data item**).
```
REPL_
```

Others are labels assigned by the user, e.g., to label user (concomitant) data.

Certain labels should not appear in the $INPUT record:

```
ETA1 ETA2 ETA3 ETA4 ETA5 ETA6 ETA7 ETA8 ETA9
```
Labels for basic PK parameters (e.g., `CL, V, K, KA`)
Labels for additional PK parameters (e.g., `S1, F0, R1, F1`)
Arguments of subroutines internal to NONMEM/PREDPP
Mu variables (e.g., `MU_ MU_i `)

When the form A=B is used, at least one of the labels A or B must be a reserved label. If A or B is a non-reserved label, it is a synonym for the reserved label. The synonym is used as the label in NONMEM output.

If DROP (or SKIP) is used as a data item label or synonym, the data item type will not appear in the NONMEM data set. DROP (or SKIP) may be used with more than one item.

If the label DATE (or DAT1, DAT2, or DAT3) is used, DATE=DROP causes the data item to not appear in the NONMEM data set. However, NMTRAN will adjust the TIME data item to reflect the date.
(See **date data item**).

REFERENCES: Guide IV, section III.B.2 , V.C.1
REFERENCES: Guide V, section 6.5
REFERENCES: Guide VI, section V.A

$LEVEL

MEANING: Specifies Nested Random Levels Above Subject ID
CONTEXT: NM-TRAN Control Record

USAGE:
`$LEVEL item=(`n_1`[`m_1`] , `n_2`[`m_2`] ... ) ...`

SAMPLE:
```
$LEVEL
```

DISCUSSION:

Identifies one or more Super ID data items.

item
: Item is the name of a data item listed on $INPUT. It defines an additional nested random level. and is referred to as a "super ID" data item. The first super ID item defines an additional random nesting level above that of subject ID. More than one super ID item may be listed on $LEVEL. Each subsequent super ID item defines an additional nesting level above that of the previous nesting level, i.e., above the previous super ID.

$n_k[m_k]$
: States that ETA(n_k) is associated with this super ID item, and ETA(m_k) is nested within ETA(n_k).

With NONMEM 7.4, a short-hand notation may be used to describe a series of values of n_k. A sequence of values for n_k can be described as
start TO end BY interval

: TO is required. The character : may be used instead of TO. BY is optional. Default is 1. The value of BY may be negative.

If the second value of BY differs, the same syntax may also be used for m_k.

Nesting levels below the subject ID is modelled as with previous versions of NONMEM. (See **Interoccasion_variability example**).

The order that super ID's are listed on $LEVEL defines their nesting level. The order that standard and super ID's are listed on $INPUT (i.e., the order in which they appear in each record of the data set) is immaterial.

When $LEVEL is used with FOCE ($ESTM METHOD=1), the SLOW option is required, and MATRIX=R is required with $COV.

EXAMPLE:

```
$INPUT ... ID ... SID ... CID ...
 ...
$PK
MU_1=THETA(1)
MU_2=THETA(2)
CL=DEXP(MU_1+ETA(1)+ETA(5)+ETA(9))
V1=DEXP(MU_2+ETA(2)+ETA(6)+ETA(10))
 ...
$LEVEL
SID=(5[1],6[2])
CID=(9[5],10[6])
```

The data item named SID is the site ID. The data item named CID is the country ID. There are several sites belonging to one country, some other sites belonging to another country, etc. For clearance, eta(9) is the country variability that has nested in it the site variability eta(5), which in turn has nested in it the subject variability (the standard ID data) eta(1). For V1, eta(10) is the country variability that has nested in it the site variability eta(6), which in turn has nested in it the subject variability (the standard ID data) eta(2).

An alternate way of coding the $LEVEL records is

```
SID=(5 to 6[1])
CID=(9 to 10[5])
```

NONMEM performs appropriate summary statistics for eta(5), and makes the appropriate constraints on eta(5), so eta(5) changes by site, that is, by every SID value change, and not by every ID value change.

The above method, using $LEVEL, is a linearized approximation at the super ID level, and takes advantage of a dual OBJ function call, freely allowing all etas to vary on the first call of OBJ, then averaging the SID etas, fixing them to these averages, and going through another OBJ call to allow the subject (ID) etas to be assessed. This approximation method works very well for the EM and Monte Carlo methods, and reasonably well for the FOCE/Laplace methods.

To perform an exact analysis, separate thetas must be defined for each value pertaining to a super ID data item, so that theta is shared only by the subjects with the particular SID value. $LEVEL is not used.
(See **superid3_6**).

If there are multiple $PROBLEM records, $LEVEL should be restated with each problem for which it is still relevant. For example, this is the case with $SUPER problems and $LEVEL.

See also `LEVWT` option of the $ESTIMATION record (NM74). By default, LEVWT=0, and weights each level value equally, regardless of number of subjects per level value. If you wish to weight according to number of subjects for that value, set LEVWT=1 on the $EST record.

REFERENCES: Guide Introduction_7

$MIX

MEANING: Marks the beginning of abbreviated code for the MIX routine
CONTEXT: NM-TRAN Control Record

USAGE:
`$MIX`
abbreviated code

SAMPLE:

```
$MIX
NSPOP=2
P(1)=THETA(5)
P(2)=1-THETA(5)
```

DISCUSSION:
The $MIX record is used to describe the mixture probabilities of a mixture model. It is evaluated with individual records.

General rules for abbreviated code are documented elsewhere (See **abbreviated code**). Specific rules follow.

ASSIGNMENT AND CONDITIONAL STATEMENTS

Left-hand quantities in assignment statements:

`NSPOP`

An integer variable. The number of sub-populations. Must be given a value when ICALL=1.

`P(i)`

For each i (i=1, ... , NSPOP), P(i) is the modeled fraction of the population in the ith subpopulation. The sum of the P(i) should equal 1. In principle, the P(i) can change from individual to individual. If for a given individual, the second (for example) subpopulation doesn't apply, then set P(2)=0 for that individual. `MIXP` may be coded instead of `P`; they are synonymous in the context of the $MIX block.

Right-hand quantities in assignment statement and in conditions:

`THETA(n).`

MIX-defined items that appeared earlier as left-hand quantities.

`ICALL`
`ICALL=0`: Run initialization.
`ICALL=1`: Problem initialization.
`ICALL=2`: Normal call.
`ICALL=4`: Simulation.

Data items listed in DATA option of $CONTR record
E.g., assume that the following records appear in the control stream prior to the $MIX block:

```
$INPUT ... STDY ...
```

```
$CONTR DATA=(STDY)
```
Then STDY may be used on the right in $MIX. STDY and STDY(1) both refer to the value of STDY on the first observation record of the individual record. STDY(i) refers to the value of STDY on the i-th. observation record of the individual record. TEMPLT may be used.
(See **MIX CONTR: TEMPLT**).

Global Variables in Modules

Certain variables in FORTRAN modules can be used. (See **Variables_in_modules**)

May not include:

EXIT, CALL , DO WHILE / ENDDO statements

COMRES, CALLFL pseudo-variables

Forbidden Variable Names:

NEWIND,ETA(i),EPS(i),ERR(i) COM(i)

Variables defined in $MIX are not listed in module NMPRD4 and may not be displayed in $TABLE and $SCATTER.

RECORD ORDER:

Follows $SUBROUTINES.

REFERENCES: Guide IV, section III.B.4 , III.B.6

REFERENCES: Guide IV, section IV.E.1 , 4.E.2

REFERENCES: Guide VI, section III.L.2 , Figure 6

$MODEL

MEANING: Specifies the MODEL subroutine of PREDPP
CONTEXT: NM-TRAN Control Record

USAGE:

```
$MODEL  [NCOMPARTMENTS=n1] [NEQUILIBRIUM=n2] [NPARAMETERS=n3]
        [COMPARTMENT=([name] [attribute1] [attribute2] ...)] ...
        [LINK compname_a [TO|AND] compname_b BY k [l]] ...
        [I_SS=n]
```

SAMPLE:

```
$MODEL    NPARAMETERS=3
          COMP=(DEPOT DEFDOSE INITIALOFF) COMP=(CENTRAL DEFOBS NOOFF)
```

DISCUSSION:
Required with a general ADVAN (ADVAN5,6,7,8,9,13,14,15,16,17,18).

OPTIONS:

`NCOMPARTMENTS=`n_1
: Total number of compartments other than the output compartment. Default: the number of COMPARTMENT options. May also be coded `NCM` or `NCOMPS`.

`NEQUILIBRIUM=`n_2
: Number of equilibrium compartments. Default: 0

`NPARAMETERS=`n_3
: Number of basic PK parameters. Default: The number of basic PK parameters defined in the $PK abbreviated code. May also be coded `NPARAMS`. May be 0 with the general non-linear models (ADVAN6, ADVAN8, ADVAN9, ADVAN13, ADVAN14, ADVAN15, ADVAN16, ADVAN17, ADVAN18).

`COMPARTMENT= ([name] [`*attribute*$_1$`] [`*attribute*$_2$`] ...)`
: Each COMPARTMENT option defines a single compartment. Compartments are numbered in the order in which they are defined. The name option gives the name for the compartment. The name option may not be one of the compartment attributes below, unless it is enclosed in quotes (' or ").
If the name option is not used, then the compartment is named "COMPn," where n is the compartment number.

  Attributes are chosen from the following list. When an attribute is not chosen, its opposite is the default.

  `INITIALOFF` - Compartment is initially off.
  `NOOFF` - Compartment may not be turned on or off.
  `NODOSE` - Compartment may not receive a dose.
  `EQUILIBRIUM` - Compartment is an equilibrium compartment (ADVAN9, ADVAN15, and ADVAN17 only; implies `NODOSE`).
  `EXCLUDE` - Compartment amount is excluded from the computation of the amount (Amt) of the output compartment (ADVAN9, ADVAN15, and ADVAN17 only). [Amt=(total amount of drug thus far input into system) minus (total amount remaining in the system)]

  The following two attributes have no opposites.

  `DEFOBSERVATION` - Compartment is the default observation compartment.

DEFDOSE - Compartment is the default dose compartment.
If no compartment has the DEFOBSERVATION attribute, the default is the first compartment with the name CENTRAL; otherwise, the first compartment. If no compartment has the DEFDOSE attribute, the default is the first compartment with the name DEPOT which may receive doses; otherwise, the first compartment which may receive doses.

`LINK compname [TO|AND] compname BY k [l]`

Link clauses are used with ADVAN5 and ADVAN7, but *only* in the absence of $PK abbreviated code. They are rarely used, and are described in Guide IV.

`I_SS=` *n*

I_SS may be used with the general non-linear models (ADVAN6, ADVAN8, ADVAN9, ADVAN13, ADVAN14, ADVAN15) to request the initial state feature of PREDPP. Values of n are

0 No initial state state (the default)
1 Initial steady state
2 Initial steady state, adds to current compartment amounts.
3 Initial steady state, use current compartment amounts as initial estimates.

The attributes `INITIALOFF NODOSE` define an output-type compartment. As with the output compartment, it is possible to use a negative value of CMT to obtain an observation and turn off such a compartment with one observation record. Like the output compartment, it is initially off, and it remains off (so that the amount therein remains zero) until it is explicitly turned on by an other type event record which has the output compartment's number in the CMT data item. Unlike the output compartment, a differential equation (DADT) must be supplied for an output-type compartment.

(See **advan68**, **advan9_15**, **$pk**)
(See **Initial Steady State: I_SS,ISSMOD**).

RECORD ORDER:
Follows $SUBROUTINES
Precedes $PK (if present)

SYMBOLIC LABEL SUBSTITUTIONS OF MODEL COMPARTMENTS (NM75)

Compartment names defined in $MODEL are automatically available for substitution in abbreviated code. This is referred to as "implicit" compartment name substitution. It is an alternative to "explicit" compartmet name substitution using $ABBR REPLACE records

For example:

```
$MODEL
COMP=(GUT,DEFDOS)
COMP=(CENTRAL,DEFOBS)
COMP=(PERI)
```

allows susbsitutions to be made for `A(GUT), DADT(GUT), A(CENTRAL), DADT(CENTRAL),` etc, so you may use these symbols in your abbreviated code, as in the following:

```
$DES
DADT(GUT)=-KA*A(GUT)
DADT(CENTRAL)=KA*A(GUT)-(KCP+KC0)*A(CENTRAL)+KPC*A(PERI)
DADT(PERI)=KCP*A(CENTRAL)-KPC*A(PERI)
 ...
```

```
$ERROR
IPRED=A(CENTRAL)/S2
```

In the generated code, compartment names are replaced by the appropriate values 1, 2, etc.

See **INTRODUCTION TO NONMEM 7, Symbolic Label Substitutions of Model Compartments**

REFERENCES: Guide IV, section V.C.4
REFERENCES: Guide VI, section VI.B

$MSFI

MEANING: Gives the name of an input Model Specification File
CONTEXT: NM-TRAN Control Record

USAGE:

```
$MSFI  filename [NORESCALE|RESCALE] [NPOPETAS[=n]]
       [ONLYREAD] [NOMSFTEST|MSFTEST]
       [VERSION=n]
       [NEW]
```

SAMPLE:

```
$MSFI   MSF13
```

DISCUSSION:
This record gives the name of an input Model Specification File.

OPTIONS:

`filename`
: Name of the Model Specification File. Must be the first option. Filename may not contain embedded spaces. If it contains commas, semicolons, or parentheses, then it must be surrounded by single quotes ' or double quotes ". Filename may also contain equal signs if it is enclosed in quotes. If filename is the same as any option of the $MSFI record, it must be enclosed in quotes. Filename may contain at most 71 characters.

`NORESCALE`
: If a search is continued, it is continued just as it would have been in the previous run. This option has no effect if the Estimation Step is not implemented or if the step is implemented, but using different options on the $ESTIMATION record from those used with the previous run. This is the default option.

`RESCALE`
: The search is to be restarted with initial estimates taken to be the final estimates from the previous run, so that a UCP value of 0.1 now corresponds to a final estimate from the previous run. This option may be used only if the search with the previous run terminated successfully and all the options used on the $ESTIMATION record coincide with those used with the previous run (except the MAXEVAL option).

`NPOPETAS[=n]`
: If the data are to be understood to be population data, the option `NPOPETAS` may be needed, and then the part "=n", n>0, may also need to be coded. The integer n should be the number of population eta variables. The part "=n" is required when the $ERROR block describes a model that changes the value of F, and there is no $PK block. It is also required when variables `A(n)` are used in the $PK abbreviated code. If the data are to be understood to be single-subject data, the option `NPOPETAS`=0 may be needed.
  May also be coded `POPETAS`.

`ONLYREAD`
: This option should be used if and only if the model specification file is being used to convey prior information to a succeeding problem in the control stream (See **TNPRI**). No task records may appear in a problem specification containing a $MSFI record with this option.

When the $MSFI record is used in a problem specification, $THETA, $OMEGA and $SIGMA records should not appear in that specification. It is permissible for the data file to differ from the original data file, but usually this is not done.

MSFTEST
: Prevents an MSF file from being utilized in a subsequent control stream file or problem if there were errors. This is the default.

NOMSFTEST
: Sometimes the MSFI error check is too strict. This occurs particularly when using classical NONMEM methods. To turn of MSFI error checking, use NOMSFTEST.

VERSION[=n]
: Read in MSF files from previous versions. With NONMEM 7.5.0, choices are:
$MSFI myfile.msf VERSION=7.4.0
$MSFI myfile.msf VERSION=7.3.0
$MSFI myfile.msf VERSION=7.2.0
$MSFI myfile.msf VERSION=7.1.2
$MSFI myfile.msf VERSION=7.1.0
$MSFI myfile.msf VERSION=6.2
$MSFI myfile.msf VERSION=6.1

NEW
: When the problem that created the MSF file has successfully completed, calling for a resumed or new estimation is prevented when the method is FO/FOCE/Laplace. To allow analysis to continue, or to allow an analysis on a new data set, resuming from the final parameters of the MSF file, use the option NEW: `$MSFI myfilename NEW`.

With NONMEM 7.3 and later, when MSF or MSFO option is used to specify an MSFO file in the $EST record e.g.,
$EST ... MSFO=msfroot.msf
then in addition to the main MSF file msfroot.msf, file msfroot_ETAS.msf containing individual etas will also be produced, and provide additional information when a $MSFI record is used in a subsequent problem or control stream. This is referred to as an "extra" msf file. If the Covariance Step is also implemented, files msfroot_RMAT.msf and msfroot_SMAT.msf containing intermediate information on the R matrix and S matrix will also be produced. These files provide information when a $MSFI record along with a $COV ... RESUME record is used in a subsequent problem or control stream.

The use of an extension, e.g., .msf, is optional. If the _ETA file is not present, NONMEM issues a warning:
`WARNING: EXTRA MSF FILE COULD NOT BE OPENED: c5msf2x_ETAS`

There is no warning if _SMAT and/or _RMAT are not present.

REFERENCES: Guide I, section B.3
REFERENCES: Guide II, section E.3
REFERENCES: Guide III, section 3.2
REFERENCES: Guide IV, section III.B.12
REFERENCES: Guide Introduction_7

$NONPARAMETRIC

MEANING: Instructions for the NONMEM Nonparametric Step
CONTEXT: NM-TRAN Control Record

USAGE:

```
$NONPARAMETRIC [MARGINALS|ETAS] [MSFO=filename] [RECOMPUTE]
               [EXPAND] [NPSUPP=n | NPSUPPE=n]
               [BOOTSTRAP [STRAT=label] [STRATF=label]]
               [PARAFILE=[filename|ON|OFF]
               [NPESTIM=[0 | 1]] [NPMAXITER=n]
               [UNCONDITIONAL|CONDITIONAL] [OMITTED]
```

SAMPLE:

```
$NONPARAMETRIC     ETAS
```

DISCUSSION:

Optional. Requests that the NONMEM Nonparametric Step be implemented. When present, the $ESTIMATION record must also be present and must specify METHOD=1 or POSTHOC.

For a given eta, the points of support are the vector of posthoc estimates of that eta for all individuals (i.e., the CPE values for that eta), which is essentially equal to the number of individuals in the data set.

OPTIONS:

`MARGINALS`
: Requests that marginal cumulatives be obtained (the default). These values are found in NONMEM global variables
(See **Nonparametric Density: DEN_,CDEN_)**

`ETAS`
: Requests that conditional (nonparametric) estimates of eta values be obtained. (Also called the CNPE).

`MSFO=filename`
: A Model Specification File is output to the given filename. Filename may not contain embedded spaces. If filename contains commas, semicolons, equal signs, or parentheses, it must be enclosed in quotes (' or "). Filename may contain at most 71 characters. If filename is the same as any option of the $NONPARAMETRIC record, it must be enclosed in quotes. The MSFO option may appear without a file name. More precisely, if the $ESTIMATION record is also present, and it also specifies the MSFO option, then the filename is required only on one of the two records, $ESTIMATION or $NONPARAMETRIC, whichever one appears first in the control stream. If the filename appears present on both records, it must be the same on both records. If the filename is omitted on the second of the two records, the MSF option must be the final option on that record. Default: No MSF is output.

`RECOMPUTE`
: Requests that the nonparametric density estimate occurring in an input MSF should be ignored; the nonparametric estimate should be recomputed.

`EXPAND`
: After the parametric estimation is performed, the final eta MAP (or empirical Bayes estimates, EBE) estimates, based on the final SIGMAS, OMEGAS, and THETAS,

are normally used as support points. When EXPAND is selected, an alternative set of EBEs are evaluated using the initial OMEGA values, but using the final THETAS and SIGMAS.

NPSUPP=n
Number of total support points to be used. If NPSUPP>number of subjects, then extra support points are randomly created from the final OMEGAS (even when EXPAND is selected). Only one of NPSUPP or NPSUPPE may be specified.

NPSUPPE=n
Number of total support points to be used. If NPSUPPE>number of subjects, then extra support points are randomly created from the initial, presumably inflated, OMEGAS (even when EXPAND is not selected). Only one of NPSUPP or NPSUPPE may be specified.

BOOTSTRAP
The original data set is fitted during the parametric estimation ($EST), and the eta support points from the original data set are used for the nonparametric version. However, a bootstrap sample, with subjects uniformly randomly selected with replacement from the original data set, is used for the nonparametric distribution analysis.

STRAT
The label of a data item that serves as the stratification. This splits the data set into distinct sub-sets, guaranteeing a specific number of subjects will be selected from each category.

STRATF
The label of a data item that contains the fraction that should represent a category in the bootstrapped data set. Without STRATF, the number of subjects to be taken from a given category is proportional to the number of subjects.

CONDITIONAL
The Nonparametric Step is implemented only when the Estimation Step terminates successfully or is not implemented (i.e., the $ESTIMATION record specifies MAXEVAL=0). This is the default.

NPESTIM=[0|1]
The default non-parametric estimation method for assessing support point probabilities is an (non-Monte Carlo) expectation-maximization (EM) method. You may also choose the non-negative least squares method (NNL, [29]), with NPESTIM=1. While it is touted to be faster than EM (NNL is quadratically convergent whereas EM is linearly convergent), several tests have not indicated that NNL is any strong speed advantage. The reason is, that the computation time increases by at least the square of the number of support points (MAX(NSPSUPP,NIND)) with the NNL method, (least squares methods require matrix inversion, which is at least an N2 order process), whereas with EM the computation time increases in proporation to MAX(NSPSUPP,NIND). Thus the larger the number of support points, the greater the speed advantage of the EM method.

NPMAXITER=n
The default maximum iterations for non-parametric estimation of assessing support point probabilities is 1000, which is usually more than enough.

PARAFILE=filename
As of NONMEM 7.4, nonparametric analysis can be parallelized.
PARAFILE=filename specifies a different parafile than was used for the previous

step.
PARAFILE=ON turns on parallelization for the Nonparametric Step.
PARAFILE=OFF turns off parallelization for the Nonparametric Step.

UNCONDITIONAL
The Nonparametric Step is always implemented.

OMITTED
The Nonparametric Step is not implemented.

REFERENCES: None.

$OLKJDF

MEANING: Specifies LKJ decorrelation degrees of freedom for each OMEGA block
CONTEXT: NM-TRAN Control Record

USAGE:
`$OLKJDF 0 (default)`
`$OLKJDF [`*value* `... ]`
`$OLKJDF [(`*value*`)[x`*n*`]] ...`

DISCUSSION:
OLKJDF is an option of the $ESTIMATION record. $OLKJDF is a separate record that allows the user to specify LKJ decorrelation degrees of freedom for each OMEGA block.

SAMPLE:
```
$OLKJDF 4.5 3.5 -2.0
```
Where 4.5 degrees of freedom are specified for the first omega block, 3.5 for the second, and - 2.0 specifies 2 degrees of freedom for the third omega block, but a user-defined definition of the standard deviations of the diagonals for the third omega block. Use the OMEGA_STD_PRIORU.f90 file in the ..\source directory as a template to modify the OMEGA_STD_PRIORU to provide the desired probability density function. Set $SUBR OTHER=OMEGA_STD_PRIORU.f90 to use the user modified template (you may rename the file for organizational purposes).

RECORD ORDER:

Follows $OMEGA

REFERENCES: Guide Introduction_7

$OMEGA

MEANING: Supplies initial estimates for the NONMEM OMEGA Matrix
CONTEXT: NM-TRAN Control Record

USAGE:

```
$OMEGA [DIAGONAL(n)|BLOCK(n)|BLOCK(n) SAME(m) | BLOCK SAME(m)]
       [[value1]  [value2]  [value3] ...
       [(value,value...) xn]
       [BLOCK(n) VALUES(diag,odiag)]
       [label=value] ...
       [BLOCK(n) [NAMES (label1,...,labeln)] [VALUES (diag,odiag)]
       [FIXED] [UNINT]
       [VARIANCE|STANDARD] [COVARIANCE|CORRELATON] [CHOLESKY]
```

SAMPLE:

```
$OMEGA BLOCK(3)  6. .005 .3 .0002 .006 .4
```

DISCUSSION:

Gives initial estimates and constraints for elements of one or several blocks of the OMEGA matrix, i.e., the matrix of variances and covariances of the eta variables in the statistical model. This record should appear only if the statistical model contains eta variables. Multiple $OMEGA records may be used to define multiple blocks of OMEGA. The order of the appearance of all blocks over all records corresponds to the order of the blocks in OMEGA.

If the initial estimates are omitted for any element(s) of OMEGA, then NONMEM will try to obtain the initial estimates.

See also the SPECIAL CASE below for the analysis type POPULATION WITH UNCONSTRAINED ETAS.

OPTIONS:

There are six forms:

1. `$OMEGA [DIAGONAL(n)] [ v11 v22 v33 ... vnn ]`

   This gives the initial estimates of the diagonal elements of a diagonal block of OMEGA. E.g.,

   ```
   $OMEGA .04 .12
   ```

   Initial estimate of variance of eta(1) = .04
   Initial estimate of variance of eta(2) = .12

   Each initial estimate may optionally be coded with one of the forms:
   init options (init options) (options init)

   With NONMEM 7.3 (*value*,*value*...)x*n* is permitted, so that repeated inputs of $OMEGA may be entered easily. Any initial value or group of initial values may be enclosed in parentheses and followed by "x*n*", which means to replicate the values within parentheses *n* times ("repeated value").

   The following options apply only to a single initial estimate (i.e., a single 1x1 block) and must follow the initial estimate unless within parentheses.

   Option `FIXED` indicates that the variance is to be constrained to be fixed to the given initial estimate. (When `FIXED` appears anywhere, then the block is described

by NONMEM as consisting of separate blocks, each of dimension one.)

Option UNINT is used with NONMEM 7.5. UNINT is used during the Optimal Design Step to identify an eta as uninteresting. UNINT may be used anywhere that FIXED may be used.

Option VARIANCE indicates that the initial estimate is understood to be a variance of the eta. This is the default.

Option STANDARD indicates that the initial estimate is understood to be a standard deviation of the eta. May also be coded SD.

An initial estimate may be 0 only if the variance or standard deviation is fixed to this estimate.

2. $OMEGA BLOCK(n) [v11 v21 v22 v31 v32 v33 ... vn1 vn2 ... vnn]

This gives the initial estimates of all the elements of a nondiagonal ("full") block of OMEGA. E.g.,

```
$OMEGA BLOCK(2) .04 .002 .12
```
Initial estimate of variance of eta(1) = .04
Initial estimate of covariance of eta(2), eta(1) = .002
Initial estimate of variance of eta(2) = .12

Any initial value or group of initial values may be enclosed in parentheses and followed by "x*n*", which means to replicate the values within parentheses *n* times ("repeated value").

The following options apply to the entire block and may appear anywhere among the list of initial estimates:

FIXED indicates that the entire block is constrained to be fixed to its initial estimate.

Option UNINT is used with NONMEM 7.5. UNINT is used during the Optimal Design Step to identify an eta as uninteresting. UNINT may be used anywhere that FIXED may be used.

VARIANCE indicates that all initial estimates given for diagonal elements are understood to be initial estimates of variances of etas. This is the default.

STANDARD indicates that all initial estimates given for diagonal elements are understood to be initial estimates of standard deviations of etas. May also be coded SD.

COVARIANCE indicates that all initial estmates given for off-diagonal elements are understood to be initial estimates of covariances of etas. This is the default.

CORRELATON indicates that all initial estmates given for off-diagonal elements are understood to be initial estimates of correlations of etas.

CHOLESKY indicates that the block is specified in its Cholesky form.

Options VARIANCE or STANDARD may be combined with COVARIANCE or CORRELATON.

Note that NONMEM converts all initial estimates to variance and covariances. The values desplayed in the NONMEM report and in the raw and additional output files are always variances and covariances.

Examples:

The following describe the same block (within rounding errors):

```
$OMEGA BLOCK(2) ; or $OMEGA VARIANCE COVARIANCE BLOCK(2)
0.64
-0.24 0.58
$OMEGA STANDARD BLOCK(2)
0.8
-0.24 0.762
$OMEGA STANDARD CORRELATION BLOCK(2)
0.8
-0.394 0.762
$OMEGA VARIANCE CORRELATION BLOCK(2)
0.64
-0.394 0.58
$OMEGA CHOLESKY BLOCK(2)
0.8
-0.3 0.7
```

The (entire) initial estimate of the block must be positive definite. The only exception is when the entire initial estimate of the block is 0, in which case it must be fixed to this estimate. Initial estimates of some of the elements of the block may be 0, while initial estimates of some other elements may be nonzero, but only in the case where the block is constrained to be of band symmetric form. That is, given the diagonal and a group of contiguous subdiagonals symmetrically ocurring across the diagonal, the elements off both the diagonal and the subdiagonals are constrained to be zero. To specify the initial estimates of such a block, the initial estimates of those elements that are to be constrained to 0 should be given as 0, while all other initial estimates should be given as nonzero. E.g., with these structures for $OMEGA BLOCK(3), the 0's are preserved:

x
0x
00x

x
xx
0xx

With NONMEM 7.3 if the initial estimate of a block is not positive definite because of rounding errors, a value will be added to the diagonal elements to make it positive definite. A message in the NONMEM report file will indicate that this was done. E.g.,

```
DIAGONAL SHIFT OF  1.1000E-03 WAS IMPOSED TO ENSURE POS-
ITIVE DEFINITENESS
```

3. `$OMEGA BLOCK(n) SAME(m)`

This describes a block whose initial estimates, as well as final estimates, are constrained to be equal to those of the preceding block. Values may not be given. "(n)" may be omitted.

With NONMEM 7.3 (m) is permitted. If (m) is present, then this record is equivalent to m identical records without (m). E.g.,

```
$OMEGA BLOCK(2) SAME(3)
```

is equivalent to

```
$OMEGA BLOCK(2) SAME
$OMEGA BLOCK(2) SAME
$OMEGA BLOCK(2) SAME
```

4.

`$OMEGA BLOCK(n) VALUES`(diag,odiag)

This supplies initial values for a block such that the initial estimates of the diagonal elements are all the same, specified by "diag", and the initial estimates of the off-diagonal elements are all the same, specified by "odiag". If present, `VALUES` must follow BLOCK. Other options (such as FIXED, CHOLESKY, VARIANCE,STANDARD,COVARIANCE,CORRELATON, UNINT) may follow VALUES or be placed between BLOCK and VALUES.
E.g.,

```
$OMEGA BLOCK(6) VALUES(0.1,0.01)
is the same as
$OMEGA BLOCK(6)
0.1
0.01 0.1
(0.01)x2 0.1
(0.01)x3 0.1
(0.01)x4 0.1
(0.01)x5 0.1
```

For fixed block (such as for omega priors):

```
$OMEGA BLOCK(6) FIX VALUES(0.15,0.0)
```

5.

`$OMEGA label=`*value* `(NM75)`

The symbolic label substitution feature is new with NONMEM 7.5. This is a compact method of defining an ETA (an element of OMEGA) specifying its initial estimate, and specifying a label for the subscript for this element of OMEGA. The label may be used as a subscript for ETA in abbreviated code, and will also identify this element of OMEGA in the NONMEM output. If new $OMEGA records change the ordering, the abbreviated code does not have to be changed. For example, suppose the first element of OMEGA that is defined happens to be

```
$OMEGA ECL=.4
```

The NONMEM report will describe the relationship, e.g.,

```
LABELS FOR ETAS
ETA(1)=ETA(ECL)
```

and ETA(CL) rather than ETA1 will appear in the NONMEM report. The abbreviated code can use this symbolic subscript instead of the numeric subscript. Then, these take effect on both ETA's and MU_'s.

For example, suppose the following code is present for the first elements of THETA and ETA. Note that $OMEGA and $THETA records must be placed ahead of any records that use the symbolic label.

```
$THETA CL=(0.0,7.0)
$OMEGA ECL= 0.3
$PK
MU_ECL=THETA(CL)
CL=EXP(MU_ECL+ETA(ECL))
```

This is equivalent to

```
$THETA (0.0,7.0)
$OMEGA .3
$PK
MU_1+THETA(1)
CL=EXP(MU_1+ETA(1))
```

Another example defines symbolic labels for a block of OMEGA:

```
$OMEGA BLOCK(4)
ECL=  0.3
EV1=  0.01 0.35
EQ=   0.01 0.01 0.54
EV2=  0.01 0.01 0.01 0.67
```

Or, for diagonals,

```
$OMEGA
ECL= 0.3
EV1= 0.35
EQ=  0.54
EV2= 0.67
```

6.

$OMEGA `BLOCK(n) NAMES` (*label*$_1$,...,*label*$_n$) `VALUES` (odiag,diag) (NM75)
With NONMEM 7.5, Symbolic label substitution may be specified for an entire block using the NAMES option. This is a compact way of defining one or more etas with labels and, when combined with VALUES, with initial values. For example

```
$OMEGA BLOCK(4) NAMES(ECL,EV1,EQ,EV2) VALUES(0.03,0.01)
```

This is equivalent to

```
$OMEGA BLOCK(4)
ECL=  0.03
EV1=  0.01 0.03
EQ=   0.01 0.01 0.03
EV2=  0.01 0.01 0.01 0.03
```

If both are present, VALUES() must come after NAMES().

SPECIAL CASE (NONMEM 7.3)

If all diagonal elements of OMEGA are "1.0E+06 FIXED", then NONMEM describes the data as

```
ANALYSIS TYPE: POPULATION WITH UNCONSTRAINED ETAS
```

Structurally NONMEM sees the analysis as population, but mathematically, the population density portion of the total likelihood is not included. This allows a population data set to be analyzed as if the data from each individual were single-subject data. Furthermore, some theta parameters could be shared across subjects ("pooled fit parameters"), whereas etas are free to fit each individual without any population constraint. Parallelization is possible.

REFERENCES: Guide I, section C.3.4.6 , D.5.2 , D.5.3
REFERENCES: Guide IV, section III.B.10 , V.C.6
REFERENCES: Guide V, section 9.3

$OMEGAP,$OMEGAPD

MEANING: Gives prior information for omegas
CONTEXT: NM-TRAN Control Record

USAGE:
`$OMEGAP` *value*$_1$ [*value*$_2$] [*value*$_3$] ...
`$OMEGAPD` *value*$_1$ [*value*$_2$] [*value*$_3$] ...

SAMPLE:
```
; Prior to OMEGA (NETPxNETP=4x4 of them)
$OMEGAP BLOCK(4)
0.2 FIX
0.0  0.2
0.0  0.0 0.2
0.0  0.0 0.0 0.2
; Set degrees of freedom of OMEGA Prior (one value per OMEGA block)
$OMEGAPD (4 FIX)
```

DISCUSSION:

These are called the informative forms (i.e., informative record names) of the $OMEGA record, for use with the NWPRI utility.

$OMEGAP gives prior information for elements of the OMEGA matrix.
$OMEGAPD gives degrees of freedom (also called the dispersion factor) for OMEGA priors.

The name of the record describes the kind of information it gives, rather than the structure of the information. E.g., in the example above, $OMEGAPD is implemented in FCON with a $THETA record because it is always a vector of values. These records may be located anywhere in the control stream. NM-TRAN inserts the corresponding records in the control stream in the correct order. When the informative forms are used, the options of $PRIOR NWPRI need not be specified. However, if options are listed in $PRIOR NWPRI, then these values are chosen over what is surmised from the informatively labeled theta/omega/sigma records.
(See **nwpri**).

OPTIONS:

The option FIXED should be used. Other appropriate options are BLOCK and VALUES.

REFERENCES: Guide Introduction_7

$OMIT

MEANING: Defines data item types to be excluded from template matching when raw data averages are computed
CONTEXT: NM-TRAN Control Record

USAGE:
`$OMIT` item1 item2 item3 ...

SAMPLE:
```
$OMIT    TIME
```

DISCUSSION:
Optional. If a label of a data item type listed in the $INPUT record, or a synonym for such a data item type, appears in the $OMIT record, then data items of this type are excluded from template matching.

REFERENCES: None

$OVARF

MEANING: Specifies the weighting to the standard deviations of OMEGA
CONTEXT: NM-TRAN Control Record

USAGE:

```
$OVARF 0 (default)
$OVARF [value ... ]
$OVARF [(value)[xn]] ...
```

SAMPLE:

```
$OVARF 2.0 5.0
```

Where 2.0 is specified for the first omega block, 5.0 for the second. If the corresponding $OLKJDF value is negative then this is argument STDSSP in user-defined OMEGA_STD_PRIORU.f90.

DISCUSSION:

The $OVARF is a separate record that allows the user to specify the weighting (inverse variance) to the standard deviations LKJ decorrelation degrees of freedom for each OMEGA block. Used with NUTS method.

$EST OVARF over-rides $OVARF.

REFERENCES: Guide Introduction_7

$PK

MEANING: Marks the beginning of abbreviated code for the PK Routine
CONTEXT: NM-TRAN Control Record

USAGE:
`$PK`
abbreviated code

DISCUSSION:
The $PK record is used to model the values of basic and additional pharmacokinetic parameters. It is used with PREDPP.

Basic PK parameters are typically the rate constants ("micro-constants") for use in kinetic formulas. $PK can compute instead parameters such as clearance and volume, and a translator ("TRANS") subroutine can be used to convert these to rate constants. Additional PK parameters include compartment scale parameters, which PREDPP uses to convert compartment amounts to concentrations, and dose-related parameters such as modeled infusion rates. General rules for abbreviated code are documented elsewhere
(See **abbreviated code**).
Specific rules for $PK follow.

ASSIGNMENT AND CONDITIONAL STATEMENTS

Left-hand quantities in assignment statements:

Basic PK parameters for the ADVAN and TRANS routines (Required, except with the General Nonlinear Kinetics Models ADVAN6, ADVAN8, ADVAN9, ADVAN13, ADVAN14, ADVAN15, ADVAN16, ADVAN17, ADVAN18). One or more of the following, depending on which ADVAN and TRANS are being used.

`K CL V` (ADVAN1)
`KA K CL V` (ADVAN2)
`K K12 K21 CL V Q VSS V1 V2 ALPHA BETA AOB` (ADVAN3)
`KA K K23 K32 CL V Q VSS V1 V2 V3 ALPHA BETA AOB` (ADVAN4)
`Km0 Kmn` (ADVAN5, ADVAN7)
`P(n)` (ADVAN6, ADVAN8, ADVAN9, ADVAN13, ADVAN14, ADVAN15, ADVAN16, ADVAN17, A
`VM, KM` (ADVAN10)
`K K12 K21 K13 K31 CL Q2 Q3 V1 V2 V3 ALPHA BETA GAMMA`
(ADVAN11)
`KA K K23 K32 K24 K42 CL Q3 Q4 V2 V3 V4 ALPHA BETA GAMMA`
(ADVAN12)

P(n) are referred to as "explicit" basic PK parameters.
Any variable defined in $PK may be used on the right-hand side in a $DES or $AES block; these are "implicit" basic PK parameters.

Additional PK parameters (Optional)
One or more of the following,
depending on the compartments defined for the model.
The digit following the letter is the compartment number.

Scale parameters Sn, e.g.: `S1 S2 S3 S4 SC S0`.
Bio-availability fractions Fn, e.g.: `F1 F2 F3`.
Output fractions Fn, e.g.: `F2 F3 F4 F0 FO`.
Infusion rates Rn, e.g.: `R1 R2 R3`.

Infusion durations Dn, e.g.: `D1 D2 D3`.
Absorption lags ALAGn, e.g.: `ALAG1 ALAG2 ALAG3`.
Time scale: `TSCALE` (may be written `XSCALE`).
Model event times MTIME(i), e.g.: `MTIME(1) MTIME(2)`.
(The subscript i is not a compartment number.)
(See **MTIME**)

Initial compartment amounts (Optional), e.g..: `A_0(1) A_0(2)`.
(See **Compartment Initialization Block**)

Initial steady state flag `I_SS` (Optional).
(See **advan68**, **advan9**, **$model**)
(See **Initial Steady State: I_SS,ISSMOD**).

PK-defined (i.e., PRED-defined) items

Right-hand quantities in assignment statement and in conditions:

Data item labels specified on the $INPUT statement.

`THETA(n)`.

`ETA(n)` (Used if the data are population.)

PK-defined items that appeared earlier as left-hand quantities.

`NEWIND`
Same as the NEWIND argument passed by NONMEM to PREDPP.
`NEWIND=0`:
First record of the data set.
THETA value may differ from value at last call with this record.
`NEWIND=1`:
First record of the data set, THETA value does not differ from value at last call with this record, and PRED is nonrecursive (see **I_REC**), or,
First record of a subsequent individual record.
`NEWIND=2`:
Subsequent data record of an individual record.

`ICALL`
Same as the ICALL argument passed by NONMEM to PREDPP.
`ICALL=1`: Initialization.
`ICALL=2`: Normal call.
`ICALL=3`: Finalization.
`ICALL=4`: Simulation.
`ICALL=5`: Expectation.
`ICALL=6`: Raw data averages.
Special rules apply to blocks of abbreviated code that are executed when ICALL is not 2.

(See **Initialization-Finalization block**, **Simulation block**).
(See **Expectation block**, **Data_Average block**).

Global Variables in Modules

Certain variables in FORTRAN modules can be used.
(See **Variables_in_modules**)
The following are of particular interest.

`DOSTIM DOSREC(n)`

If PK is not being called at an additional or lagged dose time, then DOSTIM = 0 and all elements of DOSREC are 0.

If PK is called at an additional or lagged dose time t, then DOSTIM=t
(See **Guide VI, Chapter III**)
DOSREC contains a copy of the dose event record which initiated the additional or lagged dose (actually, only of the final row: EVTREC(NVNT,*)). Data items may be referred to by position or by label, e.g., `DOSREC(1)` or `DOSREC(ID)`. In DOSREC, TIME and all user (concomitant) data items have values from the next event record. All other NONMEM/PREDPP reserved data items have values from the initiating dose event record. (The $BIND record may be used to override this.)

`A(n) TSTATE`

`A(n)` are the latest computed compartment amounts, and `TSTATE` is the time at which they were computed. That is, `A(n)` are the amounts at the previous event time, or if at a later time, but before the time for which PK is being called, a lagged or additional dose was given, or a regular infusion was terminated, or a modeled event occurred, then `A(n)` are the amounts at the latest such time. If there are population etas, and `A(n)` are used in the $PK abbreviated code, then any $OMEGA records referring to etas explicitly used in this code should precede the $PK record, or if an $MSFI record is used, it should precede the $PK record and include the option `NPOP=m`.
Note: If there is no verbatim code and no explicit use of a subscripted variable `A` in the $PK record, then the symbol `A` can be used as a data item label or as a name of a PK-defined item.

`A_0FLG`

`A_0FLG` signals a record with which it is possible to initialize a compartment amount.
(See **Compartment Initialization Block**)

Forbidden Variable Names:

`IDEF IREV EVTREC NVNT INDXS IRGG GG NETAS DADT(n) E(n) EPS(n)`

PSEUDO ASSIGNMENT STATEMENTS

`COMRES=-1`

`CALLFL=-2`: Call with every event record and at additional and lagged dose times.
`CALLFL=-1`: Call with every event record.

CALLFL=0: Call with first event record and new TIME values.
CALLFL=1: Call once per individual record.

Of the last four, CALLFL=-2 is the default when DOSREC, DOSTIM, or MTIME are used explicitly in the abbreviated code. Otherwise, CALLFL=-1 is the default.

The pseudo-assignments statements may be enclosed in parentheses. If two of them are present in the same set of parentheses, separate them with a semicolon. A calling protocol phrase may be used within parentheses instead of a pseudo-assignment statement, and either upper or lower case may be used. E.g.,

```
$PK (ONCE PER IR)       ; same as CALLFL=1
$PK (NEW TIME)          ; same as CALLFL=0
$PK (EVERY EVENT)       ; same as CALLFL=-1
$PK (NONEVENT)          ; same as CALLFL=-2
```

RECORD ORDER:

Follows $SUBROUTINES and $INPUT
Follows $MODEL (with General Nonlinear Kinetics Models ADVAN6, ADVAN8, ADVAN9, ADVAN13, ADVAN14, ADVAN15, ADVAN16, ADVAN17, ADVAN18).
Precedes $ERROR (if present)

REFERENCES: Guide IV, section V.C.5
REFERENCES: Guide V, section 7
REFERENCES: Guide VI, section III

$PRED

MEANING: Marks the beginning of abbreviated code for the PRED routine
CONTEXT: NM-TRAN Control Record

USAGE:
`$PRED`
abbreviated code

DISCUSSION:
The $PRED record is used to model values for the DV data items. It is NOT used with PREDPP (but (See **$ERROR**). General rules for abbreviated code are documented elsewhere (See **abbreviated code**). Specific rules for $PRED follow.

ASSIGNMENT AND CONDITIONAL STATEMENTS

Left-hand quantities in assignment statements:

`Y` (Required. The modeled value for the dependent variable under the statistical model.)

PRED-defined items.

Right-hand quantities in assignment statement and in conditions:

Data item labels specified on the $INPUT statement.

`THETA(n)`.

`ETA(n)` (Used if the data are population or single-subject data, and in the latter case can be coded `ERR(n)`.)

`EPS(n)` (Used if the data are population, and can be coded `ERR(n)`.)

PRED-defined items that appeared earlier as left-hand quantities. This includes `Y`.

`NEWIND`
Same as the NEWIND argument passed by NONMEM to PREDPP.
`NEWIND=0`: First record of the data set. THETA value may differ from value at last call with this record.
`NEWIND=1`: First record of the data set, THETA value does not differ from value at last call with this record, and PRED is nonrecursive (see **I_REC**), or,
First record of a subsequent individual record.
`NEWIND=2`: Subsequent data record of an individual record.

`NEWL2`
`NEWL2=1`: First record of an L2 record.
`NEWL2=2`: Otherwise.

`ICALL`
`ICALL=0`: Run initialization.
`ICALL=1`: Problem initialization.

`ICALL=2`: Normal call.
`ICALL=3`: Problem finalization.
`ICALL=4`: Simulation.
`ICALL=5`: Expectation.
`ICALL=6`: Data Average.
Special rules apply to blocks of abbreviated code that are executed when ICALL is not 2.
(See **initialization**, **finalization**, **simulation**).
(See **expectation**, **data average**).

Variables in Fortran modules
Certain variables in FORTRAN modules can be used. (See **Variables_in_modules**)

Forbidden Variable Names:

```
DATREC INDXS G H DADT(n) A(n) E(n) P(n)
```

PSEUDO ASSIGNMENT STATEMENTS

```
COMRES=-1
```

RECORD ORDER:

Follows $SUBROUTINES and $INPUT

REFERENCES: Guide I, section C.2
REFERENCES: Guide IV, section III.B.8 , IV
REFERENCES: Guide V, section 12.3

$PRIOR

MEANING: Provides instructions for the PRIOR subroutine
CONTEXT: NM-TRAN Control Record

USAGE:
`$PRIOR` subroutine [(conditional clause$_1$), (conditional clause$_2$) ...]
[`DISPLAY[=`**`ALL`**`|CNT]] [ICMAX=n]`
[argument$_1$, argument$_2$ ...]

SAMPLE:
```
$PRIOR TNPRI (PROBLEM 2) PLEV=.9999 ISS=0 IVAR=1
```

DISCUSSION:
Optional. Specifies the use of the PRIOR feature of NONMEM. Note that `$PRIOR` is a control record, not a block of abbreviated code. Therefore, only those options that are listed here may be used. E.g., verbatim code may not be used. Options and arguments may be in any order, and may be on more than one line.

OPTIONS:

subroutine

Required. Either `TNPRI` or `NWPRI`. The following options and arguments apply to calls to this subroutine. Another subroutine option (or another $PRIOR record) may follow, with a new set of options and arguments. Use only NWPRI for the new $ESTIMATION methods of NONMEM 7.

conditional clause

Optional. One or more conditions, within parentheses. The conditions are "AND"ed together, i.e., the subroutine is called when all conditions in the conditional clause are true. If there is more than one conditional clause, the clauses are "OR"ed together, i.e., the subroutine is called if all the conditions in any one conditional clause are true. See the Examples, below. Conditions may be one or more of:

`ESTIMATION` or `SIMULATION`
Specifies the NONMEM task for which the subroutine is to be called. If omitted, PRIOR is called for all tasks (i.e., for all values of ICALL). `ESTIMATION` and `SIMULATION` may not both be specified in the same conditional clause. `ESTIMATION` may be spelled `ESTIMATE` or `ESTM`; `SIMULATION` may be spelled `SIMULATE` or `SIML`.

May also be specified as `ICALL=n`, `ICALL.EQ.n`, or `ICALLn`, where n is 2 (ESTIMATION) or 4 (SIMULATION).

`PROBLEM=n`
Specifies the problem for which the subroutine is to be called. May also be specified as `PROBLEM=n` or `PROBLEM.EQ.n` or `PROBLEMn`. `PROBLEM` may also be coded as `IPROB`. Instead of `=`, `.EQ.` may be used. Other permitted tests are `.NE.`, `.LT.`, `.LE.`, `.GT.`, and `.GE`.

DISPLAY[=**ALL**|CNT]

Optional. The PRIOR subroutine will contain code to print items of interest in the NONMEM report. This is to assist the user in checking that the $PRIOR record is working correctly.

DISPLAY=ALL is the default when only DISPLAY is present. Lines such as the following are printed with every call to PRIOR:

```
PRIOR ICALL,IPROB,IREP,CNT:  2  2  0   -41.898951681785.
```

With DISPLAY=CNT, lines such as the following are printed only when PRIOR is called for simulation or estimation.

```
PRIOR CNT:    -41.898951681785
```

ICMAX=n

Optional. The PRIOR subroutine will set the given value in ICMAX prior to calling the subroutine. (See **PRIOR Simulation: ICMAX**).

Subroutine arguments

Optional. The arguments are described in the help entries for NWPRI and TNPRI. They must be coded excatly as shown, i.e., no abbreviations. Any argument that is omitted defaults to 0.

ITYP, NSAM, ISS, PLEV, CNT

Arguments for both NWPRI and TNPRI

NTHETA, NETA, NEPS, NTHP, NETP, NEPP, NPEXP

Arguments unique to NWPRI

IFND, MODE, IVAR

Arguments unique to for TNPRI

EXAMPLES:

```
$PRIOR TNPRI (ESTIMATION, PROB .GE.3 ) IFND=1
```

TNPRI is called with the Estimation step of problems 3 and higher. IFND is set to 1 with these calls; all other arguments are 0.

```
$PRIOR TNPRI IFND=1
(EST, PROB 3) (EST,PROB 4) (EST,PROB 5)
```

Same as the preceding example, if the run consists of exactly five problems. TNPRI is called with the Estimation step of problems 3, 4 and 5. IFND is set to 1 with these calls; all other arguments are 0.

```
$PRIOR TNPRI (EST, PROB 3) IFND=1
       TNPRI (EST, PROB 4) IFND=1
       TNPRI (EST, PROB 5) IFND=1
```

Same as the preceding example. TNPRI is called with the Estimation step of problem 3. IFND=1 with this call. Similarly with problems 4 and 5. Note that IFND must be specified independently each time to be 1. With re-specification of the subroutine, all arguments are reset to 0. This permits identical or different arguments with each usage of the subroutine. NM-TRAN will warn if no argument is specified in a subsequent specification of TNPRI or NWPRI, in case the arguments have been omitted by error. One or more arguments may be set explicitly to 0 to prevent the warning. When the subroutine is specified more than once, then all specifications must be conditional, i.e., conditional clauses are requrired.

Within the same run, one may use TNPRI with some tasks or problems, and NWPRI with other tasks or problems. There may be at most 10 $PRIOR records per problem.

(See **TNPRI**, **TNPRI**, **prior**).
(See **tnpri example**, **nwpri example**).

REFERENCES: Guide Introduction_VI

$PROBLEM

MEANING: Identifies the start of a NONMEM Problem Specification
CONTEXT: NM-TRAN Control Record

USAGE:
```
$PROBLEM [text]
```

SAMPLE:
```
$PROB  THEOPHYLLINE   POPULATION DATA
```

DISCUSSION:
The $PROBLEM record identifies the start of a NONMEM problem specification. The text becomes a heading for the NONMEM printout. This record is required. If the problem is part of a superproblem, $SUPER must precede the $PROBLEM record. If $SIZES is present, it must precede the first $SUPER or $PROBLEM record. Otherwise, the first NM-TRAN control record must be a $PROBLEM record. A $PROBLEM record other than the first one marks the beginning of another problem specification.

The text must be contained on a single record, and only the first 72 characters of text are used in the heading. Spaces *and* semicolons in text are regarded as part of the text. The text is optional.

REFERENCES: Guide IV, section III.B.1

$RCOV,$RCOVI

MEANING: Inputting Variance-Covariance information from another problem
CONTEXT: NM-TRAN Control Record

USAGE:

```
$RCOV
       FILE=filename [FORMAT | DELIM=s1] [TBLN=n]
$RCOVI
       FILE=filename [FORMAT | DELIM=s1] [TBLN=n]
```

SAMPLE:

```
$RCOV FILE=sirsample.cov TBLN=1 DELIM=,
```

The $RCOV record can be used to load the variance-covariance matrix of estimates results from a previous problem, and use it for subsequent use in assessing total standard errors of table items without having to re-calculate the variance with a $COV step.

The record $RCOVI may be used to load the the variance-covariance information from the inverse-covariance file:

```
$RCOVI FILE=sirsample.coi TBLN=1 DELIM=,
```

The FILE=filename option is required.

If FORMAT or DELIM is used, it should be the same as was specified on the $ESTIMATION record that created the file to be used. The default is s1PE12.5.

TBLN is the table number in the file. If TBLN is not specified, it defaults to 1.

NONMEM describes the use of these records with a message in the report file and to the terminal such as the following:

```
LOADED VARIANCE/COVARIANCE DATA FROM FILE c5.cov
```

If NONMEM is unable to read the file, the message is

```
COULD NOT FIND APPROPRIATE VARIANCE/COVARIANCE DATA IN FILE c5.cov
```

$RCOV and $RCOVI can be used with the $CHAIN record. For examples and discussion:

See **INTRODUCTION TO NONMEM 7, $RCOV and $RCOVI Record For Inputting Variance-Covariance information from another problem**.

REFERENCES: Guide Introduction_7

$SCATTERPLOT

MEANING: Requests that NONMEM generate one or more scatterplots
CONTEXT: NM-TRAN Control Record

USAGE:

```
$SCATTERPLOT list1 VS list2 [BY list3]
             [FROM n1] [TO n2] [UNIT]
             [ORD0|NOORD0] [ABS0|NOABS0] [FIRSTONLY] [OBSONLY]
             [NOSUB=[0|1]]
             [UNCONDITIONAL|CONDITIONAL] [OMITTED]
```

SAMPLE:

```
$SCATTERPLOT       (RES WRES) VS TIME BY ID
```

DISCUSSION:
Requests that families of NONMEM scatterplots be produced. Up to 20 families of scatterplots may be included in the problem. May also be coded $SCATTERS or $SCATTERGRAMS.

OPTIONS:

list1
: A list of item labels to be plotted on the ordinate axis (the long axis on the printed output). The list may be enclosed in parentheses. It may include:

  Data item labels.

  Special items PRED, RES, and WRES.

The user may request the following additional diagnostic items by including their name in the list.

NPRED, NRES, NWRES
: Calculated assuming non-conditional estimation and no eta-epsilon interaction. NPRED and NRES are same as PRED and RES. NWRES is same as WRES when INTERACTION is not set in $EST.

PREDI, RESI, WRESI
: Calculated assuming non-conditional estimation with eta-epsilon interaction. Always same as PRED, RES, and WRES.

CPRED, CRES, CWRES
: Calculated assuming conditional estimation and no eta-epsilon interaction.

CPREDI, CRESI, CWRESI
: Calculated assuming conditional estimation with eta-epsilon interaction.

CIPRED, CIRES,CIWRES
: Conditional individual values

CIPREDI, CIRESI,CIWRESI
: Conditional individual values with eta-epsilon interaction.

EPRED, ERES, EWRES
: Monte-Carlo generated diagnostics and are not linearized approximations like the other diagnostic types. EWRES is the Monte-Carlo version of CWRESI.

`ECWRES`
Monte-Carlo version of CWRES.

`NPDE`
Monte-Carlo generated normalized probability distribution error.

`NPD`
The correlated (or non-decorrelated) NPDE value.

`OBJI`
Objective function values for each individual (same as given in the root.phi file). The sum of the individual objective function values is equal to the total objective function.

Labels `ETA(1)`, `ETA(2)`, ... ,`ETA(10)`, ... , `ETA(70)`, etc.,
or alternatively, labels `ETA1`, `ETA2`, ... ,`ET10`, ... , `ET70`, etc., corresponding to eta(1), eta(2), etc.
The labels in the output will be ETA1, ETA2, ... , ET10, ... ET70, etc.

With NONMEM 7.3, a range of etas may be requested:
```
$SCAT ETAS(1:2) VS ETA3
```
is equivalent to
```
$SCAT ETA1 ETA2 VS ETA3
```
However, unlike $TABLE, $SCAT will ignore implied endings, such as
```
$SCAT ETAS(1:LAST) VS ETA3
```
And just interpret it as
```
$SCAT ETA1 VS ETA3
```
With NONMEM 7.4, more flexible syntax is available, using `TO` and `BY`.
(See **$table**).

With NONMEM 7.4, a symbolic label specified in $ABBR REPLACE may be listed in $SCAT. For example:
```
$ABBR REPLACE ETA(CL)=ETA(1)
 ...
$SCAT  ETA(CL) VS ETA3
```

Reserved positions of MODULE NMPRD4 (see **$ABBREV**). `COM(k)` or :k denotes the kth reserved position. (There must be exactly 4 digits after ":". Use leading 0's as necessary.) E.g., `COM(3)` or `:003`.
Labels of the form :k will be used in the output.

Labels of PRED-defined items in MODULE NMPRD4 if abbreviated code is present (up to PDT distinct such labels in any one problem, for all tables and scatterplots. PDT is a constant in resource/SIZES.f90; default value is 500.) These may include labels of the NM-TRAN-defined items:

0nnn	e.g., 0010	stands for A00nnn
1nnn	e.g., 1010	stands for A01nnn
2nnn	e.g., 2010	stands for C00nnn
3nnn	e.g., 3010	stands for D00nnn
4nnn	e.g., 4010	stands for E00nnn
5nnn	e.g., 5010	stands for F00nnn
6nnn	e.g., 6010	stands for P00nnn

These may also include:
labels `VECTRA(1)`, `VECTRA(2)`, ... ,`VECTRA(9)`, or alternatively, labels `VA_1`,

VA_2, ... ,VA_9, corresponding to VECTRA(1), VECTRA(2), ..., VECTRA(9).
The labels in the output will be VA_1, VA_2, ... , VA_9.
Similarly, for VECTRB and VECTRC.

Synonyms may be defined on either the $TABLE or $SCATTER record for special items PRED, RES, WRES; for PRED-defined items; for NM-TRAN-defined items; and for reserved positions of MODULE NMPRD4.

E.g., assume that IWRES is a PRED-defined label, that at least 3 positions of NMPRD4 are reserved, and that NM-TRAN has generated A00032 as the label for a derivative in the generated FSUBS routine. Either $TABLE or $SCATTER records may include:

```
WRES=RES1,IWRES=RES2,COM(3)=ABC,0032=DK.
```

For a discussion of the values of ETAs and PRED-defined items (e.g., are they based on initial or final values of theta? Simulated or zero or conditional values of eta?), see **values**.

Elements of G and H
E.g., `$SCATTER G11 BY G21`
The format is Gk1 or Hk1, where k is an integer value, e.g. 1-9 or 01-99 or 001-999. Gk1 requests the value of G(k,1), and Hk1 requests the value of H(k,1), where G and H are arguments of subroutine PRED. G(k,1) is the partial of F (the prediction) with respect to ETA(1), and H(k,1) is the partial of F with respect to EPS(1). (HH may be coded instead of H, but it is treated as if it were H.) A variable of the form Gk1 or Hk1 is not a reserved variable. If it is previously defined (i.e., if it is listed in $INPUT, or used on the left in abbreviated code, or used as a synonym e.g., $TABLE G11=COM(1)), then that definition of the variable is used, and there is no change from previous versions of NM-TRAN. Only if there is no other previous definition of the variable will it be understood to be an element of G or H.
What NM-TRAN actually displays is the variable in MODULE NMPRD4 that was used to compute the derivative of interest (a variable in the series A00nnn, C00nn, or D00nnn) with the appropriate synonym such as G11. If there is no such variable, this is an error. NMTRAN will not display variables that are not computed , e.g., G41 when there are only 3 etas in the problem, or when there are 4 etas but ETA(4) does not contribute to the value of Y.
[There is a workaround if the zero is wanted as a place holder in the table. In abbreviated code ($ERROR or $PRED or $PK)
`G41=0 .`

(See **Displayed PRED-Defined Items**).

`list2`
Like list1, but includes the labels of items to be plotted on the abscissa axis (the short axis on the printed output). The list may be enclosed in parentheses. The word "VS" is optional; it may be omitted if the lists are enclosed in parenthesis, or if each list consists of exactly one item label. VS may also be coded *. Each pair of labels, one from list1 and one from list2, defines a family of scatterplots.

`list3`
A list of one or two item labels or synonyms. Each pair of labels from list1 and list2 produces a family of scatterplots, one scatterplot for each unique value (or combination of values) of the data item(s) in list3. If the BY option is omitted, each pair of labels from list1 and list2 produces a family comprised of a single scatterplot.

UNIT
A line of unit slope is superimposed on the scatterplots.

ORD0
A line through the zero value on the ordinate axis is superimposed on the scatterplots. May also be coded ORDZERO.

NOORD0
Prevents a zero line from being superimposed on the ordinate axis of the scatterplots. May also be coded NOORDZERO.
If neither ORD0 nor NOORD0 is present, NONMEM automatically superimposes a zero line on the ordinate axis if it is appropriate for the type of data item.

ABS0
A zero line is superimposed on the abscissa axis of the scatterplots. May also be coded ABSZERO, AB0, or ABZERO.

NOABS0
Prevents a zero line from being superimposed on the abscissa axis of the scatterplots. May also be coded NOABSZERO, NOABS0, NOABZERO, NOAB0.
If neither of ABS0 and NOABS0 is present, NONMEM automatically superimposes a zero line on the abscissa axis if it is appropriate for the type of data item.

FIRSTONLY
Only the first data record from each individual record may contribute a point to the scatterplot. May also be coded FIRSTRECORDONLY or FIRSTRECONLY.

OBSONLY
The scatterplot will only use data records with MDV=0. This option applies independently of FIRSTONLY. It is not necessary when either DV, RES, or WRES is plotted.

FROM n_1
n_1 is the number of the first data record which may "contribute" to the scatterplot. Default: n_1 is 1.

TO n_2
n_2 is the number of the last data record which may "contribute" to the scatterplot. Default: There is no upper limit. All appropriate records will contribute. To restore the NONMEM VI behavior, use TO n_1+899.

With FIRSTONLY, n_1 and n_2 refer to (first records of) individual records. The remaining options apply to all $SCATTERPLOT records.

NOSUB=[0|1]
With NOSUB=0, label substitution will be performed scatterpots. (See **$ABBREVIATED**). This is the default. With NOSUB=1, label substitution will not be performed.

CONDITIONAL
The Scatterplot Step is implemented only when the Estimation Step terminates successfully or is not implemented. This is the default.

UNCONDITIONAL
The Scatterplot Step is always implemented. This is the default.

OMITTED
The Scatterplot Step is not implemented.

When DV, RES, or WRES is plotted, records having MDV=1 are not plotted.

The following symbols are used in scatterplots:

*	1 point
Overstriking:	
2-9	2-9 points
A-Z	10-35 points (A=10, B=11, ... , Z=35)
$	more than 35 points

REFERENCES: Guide IV, section III.B.17
REFERENCES: Guide V, section 9.5.2 , 10.7.2

$SIGMA

MEANING: Supplies initial estimates for the NONMEM SIGMA Matrix
CONTEXT: NM-TRAN Control Record

USAGE:

```
$SIGMA [DIAGONAL(n)|BLOCK(n)|BLOCK(n) SAME(m) | BLOCK SAME(m)]
       [[value1]  [value2]  [value3] ...
       [(value,value...) xn]
       [BLOCK(n) VALUES(diag,odiag)]
       [label=value] ...
       [BLOCK(n) [NAMES (label1,...,labeln)] [VALUES (diag,odiag)]
       [FIXED] [UNINT]
       [VARIANCE|STANDARD] [COVARIANCE|CORRELATON] [CHOLESKY]
```

SAMPLE:

```
$SIGMA BLOCK(3)  6. .005 .3 .0002 .006 .4
```

DISCUSSION:

Gives initial estimates and constraints for elements of one or several blocks of the SIGMA matrix, i.e., the matrix of variances and covariances of the eps variables in the statistical model. This record should appear only if the statistical model contains eps variables. Multiple $SIGMA records may be used to define multiple blocks of SIGMA. The order of the appearance of all blocks over all records corresponds to the order of the blocks in SIGMA.

If the initial estimates are omitted for any element(s) of SIGMA, then NONMEM will try to obtain the initial estimates.

OPTIONS:

There are six forms:

1. `$SIGMA [DIAGONAL(n)] [ v11 v22 v33 ... vnn ]`

   This gives the initial estimates of the diagonal elements of a diagonal block of SIGMA. E.g.,

   ```
   $SIGMA .04 .12
   ```

   Initial estimate of variance of eps(1) = .04
   Initial estimate of variance of eps(2) = .12

   Each initial estimate may optionally be coded with one of the forms:
   init options (init options) (options init)

   With NONMEM 7.3 (*value*,*value*...)x*n* is permitted, so that repeated inputs of $SIGMA may be entered easily. Any initial value or group of initial values may be enclosed in parentheses and followed by "x*n*", which means to replicate the values within parentheses *n* times ("repeated value").

   The following options apply only to a single initial estimate (i.e., a single 1x1 block) and must follow the initial estimate unless within parentheses.

   Option `FIXED` indicates that the variance is to be constrained to be fixed to the given initial estimate. (When `FIXED` appears anywhere, then the block is described by NONMEM as consisting of separate blocks, each of dimension one.)

   Option `UNINT` is used with NONMEM 7.5. UNINT is used during the Optimal

Design Step to identify an eps as uninteresting. UNINT may be used anywhere that FIXED may be used.

Option VARIANCE indicates that the initial estimate is understood to be a variance of the eps. This is the default.

Option STANDARD indicates that the initial estimate is understood to be a standard deviation of the eps. May also be coded SD.

An initial estimate may be 0 only if the variance or standard deviation is fixed to this estimate.

2. $SIGMA BLOCK(n) [v11 v21 v22 v31 v32 v33 ... vn1 vn2 ... vnn]

This gives the initial estimates of all the elements of a nondiagonal ("full") block of SIGMA. E.g.,

```
$SIGMA BLOCK(2) .04 .002 .12
```
Initial estimate of variance of eps(1) = .04
Initial estimate of covariance of eps(2), eps(1) = .002
Initial estimate of variance of eps(2) = .12

Any initial value or group of initial values may be enclosed in parentheses and followed by "x*n*", which means to replicate the values within parentheses *n* times ("repeated value").

The following options apply to the entire block and may appear anywhere among the list of initial estimates:

FIXED indicates that the entire block is constrained to be fixed to its initial estimate.

Option UNINT is used with NONMEM 7.5. UNINT is used during the Optimal Design Step to identify an eps as uninteresting. UNINT may be used anywhere that FIXED may be used.

VARIANCE indicates that all initial estimates given for diagonal elements are understood to be initial estimates of variances of eps. This is the default.

STANDARD indicates that all initial estimates given for diagonal elements are understood to be initial estimates of standard deviations of epsilons. May also be coded SD.

COVARIANCE indicates that all initial estmates given for off-diagonal elements are understood to be initial estimates of covariances of epsilons. This is the default.

CORRELATON indicates that all initial estmates given for off-diagonal elements are understood to be initial estimates of correlations of epsilons.

CHOLESKY indicates that the block is specified in its Cholesky form.

Options VARIANCE or STANDARD may be combined with COVARIANCE or CORRELATON.

Note that NONMEM converts all initial estimates to variance and covariances. The values desplayed in the NONMEM report and in the raw and additional output files are always variances and covariances.

Examples:

The following describe the same block (within rounding errors):

```
$SIGMA BLOCK(2) ; or $SIGMA VARIANCE COVARIANCE BLOCK(2)
0.64
-0.24 0.58
$SIGMA STANDARD BLOCK(2)
0.8
-0.24 0.762
$SIGMA STANDARD CORRELATION BLOCK(2)
0.8
-0.394 0.762
$SIGMA VARIANCE CORRELATION BLOCK(2)
0.64
-0.394 0.58
$SIGMA CHOLESKY BLOCK(2)
0.8
-0.3 0.7
```

The (entire) initial estimate of the block must be positive definite. The only exception is when the entire initial estimate of the block is 0, in which case it must be fixed to this estimate. Initial estimates of some of the elements of the block may be 0, while initial estimates of some other elements may be nonzero, but only in the case where the block is constrained to be of band symmetric form. That is, given the diagonal and a group of contiguous subdiagonals symmetrically ocurring across the diagonal, the elements off both the diagonal and the subdiagonals are constrained to be zero. To specify the initial estimates of such a block, the initial estimates of those elements that are to be constrained to 0 should be given as 0, while all other initial estimates should be given as nonzero. E.g., with these structures for $SIGMA BLOCK(3), the 0's are preserved:

x
0x
00x

x
xx
0xx

With NONMEM 7.3 if the initial estimate of a block is not positive definite because of rounding errors, a value will be added to the diagonal elements to make it positive definite. A message in the NONMEM report file will indicate that this was done. E.g.,

```
DIAGONAL SHIFT OF  1.1000E-03 WAS IMPOSED TO ENSURE POS-
ITIVE DEFINITENESS
```

3. `$SIGMA BLOCK(n) SAME(m)`

This describes a block whose initial estimates, as well as final estimates, are constrained to be equal to those of the preceding block. Values may not be given. "(n)" may be omitted.

With NONMEM 7.3 (m) is permitted. If (m) is present, then this record is equivalent to m identical records without (m). E.g.,

```
$SIGMA BLOCK(2) SAME(3)
```

is equivalent to

```
$SIGMA BLOCK(2) SAME
$SIGMA BLOCK(2) SAME
$SIGMA BLOCK(2) SAME
```

4.

`$SIGMA BLOCK(n) VALUES(diag,odiag)`

This supplies initial values for a block such that the initial estimates of the diagonal elements are all the same, specified by "diag", and the initial estimates of the off-diagonal elements are all the same, specified by "odiag". If present, `VALUES` must follow BLOCK. Other options (such as FIXED, CHOLESKY, VARIANCE,STANDARD,COVARIANCE,CORRELATON, UNINT) may follow VALUES or be placed between BLOCK and VALUES.
E.g.,

```
$SIGMA BLOCK(6) VALUES(0.1,0.01)
is the same as
$SIGMA BLOCK(6)
0.1
0.01 0.1
(0.01)x2 0.1
(0.01)x3 0.1
(0.01)x4 0.1
(0.01)x5 0.1
```

For fixed block (such as for sigma priors):

```
$SIGMA BLOCK(6) FIX VALUES(0.15,0.0)
```

5.

`$SIGMA label=`*value* `(NM75)`

The symbolic label substitution feature is new with NONMEM 7.5. This is a compact method of defining an EPS (an element of SIGMA) specifying its initial estimate, and specifying a label for the subscript for this element of SIGMA. The label may be used as a subscript for EPS in abbreviated code, and will also identify this element of SIGMA in the NONMEM output. If new $SIGMA records change the ordering, the abbreviated code does not have to be changed. For example, suppose the first element of SIGMA that is defined happens to be

```
$SIGMA RSW=.06
```

The NONMEM report will describe the relationship, e.g.,

```
LABELS FOR EPS
EPS(1)=EPS(RSW)
```

and EPS(RSW) rather than EPS1 will appear in the NONMEM report. The abbreviated code can use this symbolic subscript instead of the numeric subscript.

As with $OMEGA, $SIGMA and $THETA records (if elements of THETA are used) must be placed ahead of any records that use the symbolic label.

Another example defines symbolic labels for a block of SIGMA:

```
$SIGMA BLOCK(2)
RSW=  0.3
EX=   0.01 0.35
```

Or, for diagonals,

```
$SIGMA
RSW= 0.3
EX= 0.35
```

6.

$SIGMA `BLOCK(n) NAMES` (*label*$_1$,...,*label*$_n$) `VALUES` (odiag,diag) (NM75)
With NONMEM 7.5, Symbolic label substitution may be specified for an entire block using the NAMES option. This is a compact way of defining one or more epsilons with labels and, when combined with VALUES, with initial values. For example

```
$SIGMA BLOCK(2) NAMES(RSW,EX) VALUES(0.03,0.01)
```

This is equivalent to

```
$SIGMA BLOCK(2)
RSW=  0.03
EX=   0.01 0.03
```

If both are present, VALUES() must come after NAMES().

REFERENCES: Guide I, section C.3.4.6 , D.5.2 , D.5.3
REFERENCES: Guide IV, section III.B.10 , V.C.6
REFERENCES: Guide V, section 9.3

$SIGMAP,$SIGMAPD

MEANING: Gives prior information for sigmas
CONTEXT: NM-TRAN Control Record

USAGE:
`$SIGMAP` *value*$_1$ [*value*$_2$] [*value*$_3$] ...
`$SIGMAPD` *value*$_1$ [*value*$_2$] [*value*$_3$] ...

SAMPLE:
```
; Prior to SIGMA (NEPPxNEPP=1x1 of them)
$SIGMAP 0.05 FIX
; Set degrees of freedom of SIGMA Prior (one value per SIGMA block)
$SIGMAPD (1 FIX)
```

DISCUSSION:

These are called the informative forms of the $SIGMA record, for use with the NWPRI utility.

$SIGMAP gives prior information for elements of the SIGMA matrix.
$SIGMAPD gives degrees of freedom (also called the dispersion factor) for SIGMA priors.

The name of the record describes the kind of information it gives, rather than the structure of the information. E.g., in the example above, $SIGMAPD is implemented in FCON with a $THETA record because it is always a vector of values. These records may be located anywhere in the control stream. NM-TRAN inserts the corresponding records in the control stream in the correct order. When the informative forms are used, the options of $PRIOR NWPRI need not be specified. However, if options are listed in $PRIOR NWPRI, then these values are chosen over what is surmised from the informatively labeled theta/omega/sigma records.
(See **nwpri**).

OPTIONS:

The option FIXED should be used. Other appropriate options are BLOCK and VALUES.

REFERENCES: Guide Introduction_7

$SIMULATION

MEANING: Instructions for the NONMEM Simulation Step
CONTEXT: NM-TRAN Control Record

USAGE:

```
$SIMULATION (seed1 [seed2] [NORMAL|UNIFORM|NONPARAMETRIC] [NEW]) ...
            [CLOCKSEED=[0|1]]
            [SOURCE_EPS=n]
            [SUBPROBLEMS=n] [ONLYSIMULATION] [OMITTED]
            [REQUESTFIRST] [REQUESTSECOND] [PREDICTION|NOPREDICTION]
            [TRUE=INITIAL|FINAL|PRIOR] [TTDF=n]
            [BOOTSTRAP=n [REPLACE|NOREPLACE] [STRAT=label] [STRATF=label]]
            [NOREWIND|REWIND] [SUPRESET|NOSUPRESET]
            [RANMETHOD=[n|S|m|P] ]
            [PARAFILE=[filename|ON|OFF]
```

SAMPLE:

```
$SIMULATION     (889215690) (2239177789 UNIFORM)
```

DISCUSSION:

Optional. Requests that the NONMEM Simulation Step be implemented. May also be coded `$SIMULATE` or `$SIML`.

Usually, when the Simulation Step is implemented, the simulated observation is taken to be the quantity to which the Y variable (with NM-TRAN abbreviated code) or F variable (with a user-supplied PRED or ERROR routine) is set. This is the default behaviour. However, see option `NOPREDICTION` below.

If a $ESTIM record appears in the problem specification, then unless the `LIKELIHOOD` or `-2LOGLIKELIHOOD` option appears on the $ESTIM record, etas (if any) are understood to be single-subject etas, except when epsilons also appear, in which case the etas are understood to be population etas. If a $ESTIM record does not appear, but a $SIMUL record appears, then unless the `NOPREDICTION` option appears on the $SIMUL record, etas are understood in the same way. When the `NOPREDICTION` option is used, the etas are understood to be population etas.

In NM-TRAN abbreviated code, there can be a special block of code that is implemented only during the simulation task (See **ICALL**, **Simulation Block**). It is called a "simulation block".

OPTIONS:

The information coded within each set of parentheses defines the attributes of a single random source. A source of random numbers is an "infinite" stream of random numbers. Each pair of parentheses on the record defines a separate source of random numbers, and the information coded within the parentheses defines the attributes of the source. The sources are ordered as they are defined on the $SIMUL record. The numbers from a source are explicitly available to the user via the NONMEM utility routine: RANDOM (See **RANDOM**). By default, the first source is earmarked for the simulation of etas and epsilons, and then the numbers from this source are not explicitly available to the user. However, see option `SOURCE_EPS` below.

seed1

Seed1 is the first seed for the random source, an integer between 0 and 2147483647. If this is not the first problem specification in the control stream, then seed1 can be

-1, indicating that the random source is to be continued from the previous problem.

seed2

Seed2 is the second seed for the random source, an integer between 0 and 2147483647. For use of a second seed, see NONMEM Users Guide, Part IV.

NORMAL

The random numbers of the source are to be pseudo-normal with mean 0 and variance 1 (unless the source is the first and used to generate eta and epsilon realizations, in which case the variance-covariance of these variables is that specified in the $OMEGA and $SIGMA records). This is the default.

UNIFORM

The random numbers of the source are to be pseudo-uniform on the interval [0,1].

CLOCKSEED=[0|1] (NM75)

If CLOCKSEED=1 (default is 0), actual starting seed will be 10000*(seconds after midnight)+SEED (applies to both seed1 and seed2, if specified.) This allows a control stream to produce different stochastic results for automated replications, without the need to modify the seed value in the control stream file in each replication.

SOURCE_EPS=n (NM75)

May be used to specify that the simulation of epsilons should use a different source than the default, which is the first. n is the number of the source for epsilons. This source must have the NORMAL attribute.

NONPARAMETRIC

The random numbers from the first source defined with the NONPARAMETRIC attribute are used to generate realizations of random vectors from a (multivariate) nonparametric distribution on eta, obtained from the Nonparametric Step of an earlier problem. An input MSF must also be present. It is understood that the etas are to be simulated from the nonparametric distribution rather than from the pseudo-normal distribution associated with the first source. The NONPARAMETRIC attribute can only be used in the definition of the second or subsequent source.

NEW

If the NEW option is used, the vector of eta's (epsilon's) changes with each call to SIMETA (SIMEPS), rather than only at the start of the next individual record (next level-two record). That is, with NEW, a call to SIMETA will obtain new eta's, even if ID has not changed. A call to SIMEPS will obtain new eps's, even if L2 has not changed. The output NONMEM report describes this as "DIFFERENT ETA AND EPS WITH EACH CALL TO SIMETA AND SIMEPS".

ONLYSIMULATION

NONMEM is being asked to simulate data but not to evaluate an objective function on these data. WRES values in tables and scatterplots will be 0. PRED-defined data items in tables and scatterplots will be computed using simulated etas and initial thetas.

$ESTIM, $COV and $NONP cannot be used with ONLYSIMULATION. Also, see the PREDICTION and NOPREDICTION options.

SUBPROBLEMS=n

Requests that the entire NONMEM problem is to be repeated n times in succession (including all NONMEM steps: simulation, estimation, covariance, table, scatterplot). Each subproblem includes the Simulation Step, but the random sources are simply continued from subproblem to subproblem. If n=0 or n=1, there is only one

subproblem; this is the default. May also be coded SUBPROBS, NSUBPROBLEMS, NSUBPROBS. With all versions of NONMEM, the data set for each subproblem after the first is the same data set used by the previous subproblem, and includes any changes (transgeneration) made by the previous subproblem.
With NONMEM 7.4 and higher, see REWIND, below.
With NONMEM 7.3 and higher, the maximum number of subproblems is increased from 9999 to 2147483647.

REQUESTFIRST
NONMEM sets a variable IFIRSTEM in Module ROCM_INT (referenced as FIRSTEM in abbreviated code) informing PRED whether or not PRED needs to compute first-partial derivatives with respect to eta. Normally, during the Simulation Step, these derivatives are not needed, either by NONMEM or by the user. However, the user may want the first-partial eta derivatives of a PRED-defined item and may want FIRSTEM to reflect this. With the REQUESTFIRST option, the FIRSTEM variable is set so to inform PRED that the derivatives need to be computed. In this case, if an abbreviated code is used to compute the PRED-defined item, the item should not be computed within a simulation block, because NM-TRAN does not provide derivatives for PRED-defined items in a simulation block.

REQUESTSECOND
NONMEM sets a variable ISECDER in Module ROCM_INT (referenced as MSEC in abbreviated code) informing PRED whether or not PRED needs to compute second-partial derivatives with respect to eta. Normally, during the Simulation Step, these derivatives are not needed, either by NONMEM or by the user. However, the user may want the second-partial eta derivatives of a PRED-defined item and may want the MSEC variable to reflect this. With the REQUESTSECOND option, the MSEC variable is set so to inform PRED that the derivatives need to be computed. In this case, if an abbreviated code is used to compute the PRED-defined item, the item should not be computed within a simulation block, because NM-TRAN does not provide derivatives for PRED-defined items in a simulation block. REQUEST-SECOND implies REQUESTFIRST.

PREDICTION
Permitted only with ONLYSIM, and is the default.
With or without ONLYSIM, unless the NOPREDICTION is used, the simulated observation is taken to be the quantity to which the Y variable (with NM-TRAN abbreviated code) or F variable (with a user-supplied PRED or ERROR routine) is set. In a simulation block, the DV variable may be directly set to the simulated observation, but the Y (or F) variable should also be set to this observation. E.g., if a line of code DV=... is used in a simulation block, be sure to follow this line with the additional line Y=DV.

NOPREDICTION
Permitted only with ONLYSIM.
Indicates that the simulated observation will be taken to be the value to which the DV variable is set. The code Y=... is permitted inside or outside a simulation block, but if such code appears in a simulation block, be sure to also include e.g. DV=Y. Also, etas (if any) are understood to be population etas, even if epsilons do not appear.

TRUE=INITIAL
The initial estimates given in the control stream are used as the parameter values ("true values") in the simulation, except when a model specification file is input.

When a model specification file is input, the initial estimates with the previous run are used as the parameter values ("true values") in the simulation, and the final estimates with the previous run are used as the initial estimates in all tasks other than the simulation. The UCP used with these other tasks are the same as with the previous run. This is the default. May not be used with $MSFI in conjunction with SUBPROBLEMS=n (n > 1).

TRUE=FINAL

An input model specification file must be used. The final estimates with the previous run are used as the parameter values ("true values") in the simulation and as the initial estimates in all tasks other than the simulation. The UCP used with these other tasks are new, so that a UCP value of 0.1 now corresponds to a final estimate from the previous run.

TRUE=PRIOR

The values stored in THET_P, OMEG_P, SIGM_P by the PRIOR routine are used as the true parameter values ("true values") in the simulation.

TTDF=n (NM75)

If TTDF is set to a non-zero integer value n, thetas are simulated with n degrees of freedom t-distribution. TRUE=PRIOR must be specified as well. Priors must also be specified appropriately.

See also $TTDF control record (NM75) to specify degrees of freedom for each theta. The value of TTDF overrides $TTDF

See INTRODUCTION TO NONMEM 7, Simulating THETAS with t-Distribution

BOOTSTRAP=n

With the BOOTSTRAP option, NONMEM does not perform the usual simulation activity of randomly creating DV values for a new data set, but rather selects a random set of subjects from an existing "template" data set (which must already have legitimate DV values). The BOOTSTRAP number n refers to how many subjects are to be randomly selected from the data set. Setting -1 means to randomly select as many subjects as are in the data set. For example, if 400 subjects are in the simulation template data set, then 400 subjects are randomly selected. The random source is, in effect, uniform, because any subject is equally probable. If n is greater than the number of subjects, NONMEM will use the number of subjects.

The BOOTSTRAP option in $SIML is most suitable for multi-subject data, in which there is an ID data column identifying the subjects. However, see the example "BOOTSTRAPPING SINGLE SUBJECT DATA" in the INTRODUCTION TO NONMEM 7.

REPLACE

Subjects are selected with replacement. This results in some subjects not being selected at all, and some subjects selected more than once. This is the default.

NOREPLACE

Subjects are selected without replacement, that is, without repeating a subject. The NOREPLACE feature is reasonable if there are many more than n subjects to choose from in the template dataset (for example, 1000 subjects in the template, and for each sub-problem, 50 of them are randomly chosen without replacement, that is, without repeating a subject).

STRAT

The label of a data item that serves as the stratification. This splits the data set into distinct sub-sets, guaranteeing a specific number of subjects will be selected from

each category.

STRATF

The label of a data item that contains the fraction that should represent a category in the bootstrapped data set. Without STRATF, the number of subjects to be taken from a given category is proportional to the number of subjects in the base data set.

NOREWIND|REWIND

The **NOREWIND** option requests that if any input data item(s) are changed (transgenerated) during a the simulation step of a sub-problem, they remain changed at the start of the next sub-problem. This is the default. With NONMEM 7.4, the REWIND option requests that the original data set be used for all sub-problems. If the data set is not changed during simulation, the NOREWIND and REWIND options give the same results.

Keep in mind that transgeneration performed on the data set using $INFN with ICALL=1 affects the original data set and these changes are unaffected by REWIND and NOREWIND options. For example:

```
$INFN
IF (ICALL==1) THEN
DOWHILE(DATA)
 ... data transgeneration statements here
ENDDO
ENDIF
```

SUPRESET|NOSUPRESET

This option affects seeds for random sources when a $SIMULATION record is included in the scope of a $SUPER problem. The SUPRESET option requests that, with subsequent iterations of a super-problem, the seed(s) for all random sources are reset back to that listed in the $SIMULATION record of the control stream file. This is the default. With NONMEM 7.4, the NOSUPRESET option may be used to prevent resetting the random seeds, so that each iteration serves as a new random instance.

RANMETHOD=[n|S|m|P]

As of NONMEM 7.3, the RANMETHOD option is available for the $SIM record, to use alternative random numbers generators. N is the random number generator type, S is Sobol sequence, and m is the Sobol scrambler. The default is NONMEM's traditional one, n=4. Among the Sobol sequence methods, the S2 method appears to provide the least biased random samples, that is nearly uniform distribution, with good mixing in multi-dimensional spaces. As of NONMEM 7.4, RANMETHOD will also act on the P modifier , which will retain separate seed sequences for each subject, so that the random variable patterns are retained regardless of whether the simulation is done in single computing or parallel computing mode.

(See **$ESTIMATION**).

PARAFILE=filename

As of NONMEM 7.4, the Simulation Step computation may be parallelized. by default. By default, parallelization is not turned on, because simulation is very rapid anyway, and often does not need to be accelerated. When modeling with super-ID nested ETA levels ($LEVEL record is present), parallelization will not occur, since these etas are shared across individuals, and there is no guarantee that all subjects sharing the same etas will be simulated by the same process.

PARAFILE=filename specifies a different parafile than was used for the previous step.

PARAFILE=ON turns on parallelization for the Simulation Step. Set RANMETHOD=P to permit assure consistient seed patterns regardless of whether or not parallelization is performed. E.g., $SIML ... PARAFILE=ON RANMETHOD=P
PARAFILE=OFF turns off parallelization for the Simulation Step. This is the default.

OMITTED
The Simulation Step is not implemented.

REFERENCES: Guide IV, section III.B.13 , IV.I
REFERENCES: Guide V, section 12.4.8
REFERENCES: Guide VI, section III.C , III.E , IV.B , IV.G.1

$SIZES

MEANING: Array sizes for NONMEM and PREDPP
CONTEXT: NM-TRAN Control Record

USAGE:
`$SIZES` [constant=value] [constant=value] ...

SAMPLE:
```
$SIZES LIM1=30000 MAXFCN=2000000 NO=500
```

DISCUSSION:
$SIZES is optional. If present, it must precede the first $PROBLEM or $SUPER record.

Certain constants are used in NM-TRAN, NONMEM and PREDPP. With NONMEM 7.2 and higher, the user may override many of the constants with the $SIZES record.

See the discusion of sizes for a discussion of how these constants are determined, and how they are communicated to NONMEM and PREDPP.
(See **SIZES FSIZES prsizes**)

Any non-zero value specified on the $SIZES record overrides both the default and the value that NM-TRAN would have specified. (A value of 0 is ignored.) As of NONMEM 7.3, as an alternative to modifying sizes.f90 to very large maximum sizes, you can tell NMTRAN the maximum size that may be needed by specifying a $SIZES constant as a negative value. Thus, a user can give NMTRAN permission to deal with all problems that have data input files that have up to 1000 data items, and up to 150 etas and epsilons, and up to 200 thetas, by the following:

```
$SIZES PD=-1000 LVR=-150 LTH=-200
```

but the values of these constants when the NONMEM executable is constructed will be only what is needed for the particular problem. In contrast,

```
$SIZES PD=1000 LVR=150 LTH=200
```

will result in sizing the NONMEM executable with these values, and won't make a "tailor fit". This would result in a very large executable regardless of the model size. Thus,
```
$SIZES PD=-1000
```
tells NMTRAN that you may need as many as 1000 data items in a data file, whereas
```
$SIZES PD=1000
```
tells NMTRAN that you need exactly that size.

List of $SIZES Record Options and Their Default Values

LTH=100
LVR=30
LVR2=20
NO=250
MMX=10
LNP4=4000
LSUPP=4050
LIM7=2
LWS3=9000
MAXIDS=10000
LIM1=10000
LIM2=100000

LIM3=10000
LIM4=1000
LIM5=200
LIM6=400
LIM8=200
LIM11=25
LIM13=1000
LIM15=1000
LIM16=400
MAXRECID=200
PC=30
PCT=30
PIR=700
PD=50
PDT=50
PAL=50
MAXFCN=1000000
DIMTMP=500
DIMCNS=500
DIMNEW=1000
DIMVRB=200
PL=10
NPOPMIXMAX=10
MAXOMEG=70
MAXPTHETA=90
MAXITER=210
ISAMPLEMAX=10
MAXSIDL=0
PNM_MAXNODES=100
MAXNRDS=PC
PAST_SIZE=4000

Additional constants that may be set with $SIZES:

`LADD_MAX (See resource/SIZES.f90).`

(See **SIZES FSIZES prsizes**)

REFERENCES: Guide Introduction_7

$SLKJDF

MEANING: Specifies LKJ decorrelation degrees of freedom for each SIGMA block
CONTEXT: NM-TRAN Control Record

USAGE:

```
$SLKJDF 0 (default)
$SLKJDF [value ... ]
$SLKJDF [(value)[xn]] ...
```

DISCUSSION:
SLKJDF is an option of the $ESTIMATION record. $SLKJDF is a separate record that allows the user to specify LKJ decorrelation degrees of freedom for each SIGMA block.

SAMPLE:

```
$SLKJDF 4.5 3.5 -2.0
```

Where 4.5 degrees of freedom are specified for the first sigma block, 3.5 for the second, and - 2.0 specifies 2 degrees of freedom for the third sigma block, but a user-defined definition of the standard deviations of the diagonals for the third sigma block. Use the SIGMA_STD_PRIORU.f90 file in the ..\source directory as a template to modify the SIGMA_STD_PRIORU to provide the desired probability density function. Set $SUBR OTHER=SIGMA_STD_PRIORU.f90 to use the user modified template (you may rename the file for organizational purposes).

RECORD ORDER:

Follows $SIGMA

REFERENCES: Guide Introduction_7

$SUBROUTINES

MEANING: Lists certain subroutine choices for the NONMEM Executable
CONTEXT: NM-TRAN Control Record

USAGE:

```
$SUBROUTINES [subname1 = name1] [subname2 = name2] ...
             [TOL=n] [ATOL=n] [SSTOL=n] [SSATOL=n]
             [SUBROUTINES=kind]
```

SAMPLE:

```
$SUBROUTINES    PRED=pred
```

DISCUSSION:
Optional. Describes the choice of subroutines for the NONMEM executable (also called the NONMEM load module). May also be coded `$SUBS`.

One of the following is required:
`ADVAN`=name (also specifies the use of PREDPP).
`PRED`=name (specifies a user-supplied PRED routine).

OPTIONS:

`subname=name`

Subname is the entry name of a user-supplied subroutine to be included in the NONMEM executable. Name is the name of a file containing FORTRAN source code for the subroutine. Name is used by NM-TRAN as documentation (in FREPORT) and for inclusion of source code (in FSUBS). More than one such option may be specified. The names need not be unique. Name may not contain embedded spaces.
Name may contain as many characters as fit a single line. It must not start with a digit. If name contains commas, semicolons, equal signs, or parentheses, it must be surrounded by single quotes ' or double quotes ".

Subname may be chosen from the following categories.

User-supplied NONMEM routines:

```
CRIT MIX PRED CRIT MIX PRIOR CONTR CCONTR USMETA SPTWO
```

User-supplied PRED routine:

```
PRED
```

Subroutines from the PREDPP library:

```
ADVAN TRANS SS
```

(See **ss_option**).

User-supplied PREDPP routines:

```
PK ERROR MODEL DES AES TOL INFN
```

Other user-supplied subroutines:

```
OTHER
```

OTHER specifies the name of a file that will be copied into FSUBS (e.g., `OTHER=filename`). A subroutine or function in file filename might be called by a user-supplied routine or by verbatim code. With NONMEM 7.4, it might be listed on a $ABBR FUNCTION record. The OTHER option may be used up to 40 times, to specify the names of up to 40 such files; each file may contain multiple subroutines and functions. These routines must be in Fortran

90 format.

`SUBROUTINES=kind`

Specifies the kind of subroutines to be included in the NONMEM executable ("kind" must be `DP` - double precision).

`TOL=n`

When PREDPP is specified with an ADVAN that requires a TOL routine, this option can be used to supply a NRD ("number of required digits") value. "n" is an integer. This is a relative tolerance.

For TOL and the options that follow (ATOL, SSTOL, SSATOL), it is also possible to code `TOL`=name to specify the name of a user-supplied TOL routine, or to include $TOL abbreviated code, either of which allows all these values to be assigned by compartment. A user-supplied TOL routine also allows values to be assigned by compartment and for each NONMEM step. See also the TOL option of the $COVARIANCE record. Required.

`ATOL=n`

Specifies the absolute tolerance for ADVAN9, ADVAN13, ADVAN14, ADVAN15, ADVAN16, ADVAN17, and ADVAN18. Optional. Default is 1.0E-12. See also the ATOL option of the $ESTIMATION and $COVARIANCE records.

`SSTOL=n`

Specifies the relative tolerance for Steady State evaluation. Optional. Default is TOL.

`SSATOL=n`

Specifies the absolute tolerance for Steady State evaluation. Optional. Default is ATOL.

REFERENCES: Guide IV, section III.B.6

$SUPER

MEANING: Identifies the start of a NONMEM Superproblem
CONTEXT: NM-TRAN Control Record

USAGE:
`$SUPER [SCOPE=`n_1`][ITERATIONS=`n_2`]`
`[`**`NOPRINT`**`|PRINT]`

SAMPLE:
```
$SUPER      SCOPE=2 ITERATIONS=10
```

DISCUSSION:
Optional.

OPTIONS:

`SCOPE=`n_1
Number of problems in the superproblem. Required. Must be at least 1.

`ITERATIONS=`n_2
Number of iterations of the superproblem. Required. Must be at least 2. `ITERATIONS` may also be coded `NITERATIONS`.

`NOPRINT`
NONMEM printout displaying the input information for each of the problems of the superproblem will be generated only during the first iteration. This is the default.

`PRINT`
NONMEM printout displaying the input information for each of the problems of the superproblem will be generated during all iterations.

To define a sequence of problems as a superproblem, precede the $PROBLEM record of the first problem of the sequence with a $SUPER record. More than one $SUPER problem can be present in the control stream, but the level of nesting can be at most 2.

The pattern for nested super problems is

```
$SUPER SCOPE=s1
$PROBLEM 1
 ...
$PROBLEM n
$SUPER SCOPE=s2
$PROBLEM n+1
 ...
$PROBLEM n+m
```

The second super problem must be totally contained with in the first.

n >= 1 and m >= 1 is required (each $SUPER problem must contain at least one $PROBLEM).

If s1 > n, then the scope of the first $SUPER includes the second $SUPER, and the scope of the second must be entirely contained within the scope of the first: s1 >= n+s2.

(See **Problem_Iteration_Counters**).

REFERENCES: None.

$SVARF

MEANING: Specifies the weighting to the standard deviations of SIGMA
CONTEXT: NM-TRAN Control Record

USAGE:
`$SVARF 0 (default)`
`$SVARF [`*value* `... ]`
`$SVARF [(`*value*`)[x`*n*`]] ...`
SAMPLE:

```
$SVARF 2.0 5.0
```

Where 2.0 is specified for the first sigma block, 5.0 for the second. If the corresponding $SLKJDF value is negative then this is argument STDSSP in user-defined SIGMA_STD_PRIORU.f90.

DISCUSSION:

The $SVARF is a separate record that allows the user to specify the weighting (inverse variance) to the standard deviations LKJ decorrelation degrees of freedom for each SIGMA block. Used with NUTS method.

$EST SVARF over-rides $SVARF.

REFERENCES: Guide Introduction_7

$TABLE

MEANING: Requests that NONMEM generate a table
CONTEXT: NM-TRAN Control Record

USAGE:

```
$TABLE [list1] [BY list2]
       [PRINT|NOPRINT] [FILE=filename]
       [NOHEADER|ONEHEADER] [ONEHEADERALL]
       [NOTITLE|NOLABEL]
       [FIRSTONLY|LASTONLY|FIRSTLASTONLY] [NOFORWARD|FORWARD]
       [APPEND|NOAPPEND]
       [FORMAT=s] [LFORMAT=s] [RFORMAT=s]
       [IDFORMAT=s]
       [NOSUB=[0|1]]
       [EXCLUDE_BY list3]
       [PARAFILE=[filename|ON|OFF]
       [ESAMPLE=n1][WRESCHOL]
       [SEED=n2] [CLOCKSEED=[0|1]]
       [RANMETHOD=[n|S|m] ]
       [VARCALC=[0|1|2|3]]
       [FIXEDETAS= (list) ]
       [NPDTYPE=[0|1]]
       [UNCONDITIONAL|CONDITIONAL] [OMITTED]
```

SAMPLE:

```
$TABLE           ID DOSE WT TIME
```

DISCUSSION:

Requests that a NONMEM table be produced. Up to 10 $TABLE records may be included in a given problem.

OPTIONS:

`list1`

A list of item labels (i.e., user-chosen item types) to be tabled along with DV and the special items `PRED`, `RES`, and `WRES`.

The user may request the following additional special diagnostic items by including their name in the list.

`NPRED`, `NRES`, `NWRES`

Calculated assuming non-conditional estimation and no eta-epsilon interaction. NPRED and NRES are same as PRED and RES. NWRES is same as WRES when INTERACTION is not set in $EST.

`PREDI`, `RESI`, `WRESI`

Calculated assuming non-conditional estimation with eta-epsilon interaction. Always same as PRED, RES, and WRES.

`CPRED`, `CRES`, `CWRES`

Calculated assuming conditional estimation and no eta-epsilon interaction.

`CPREDI`, `CRESI`, `CWRESI`

Calculated assuming conditional estimation with eta-epsilon interaction.

`CIPRED`, `CIRES`,`CIWRES`
: Conditional individual values

`CIPREDI`, `CIRESI`,`CIWRESI`
: Conditional individual values with eta-epsilon interaction.

`EPRED`, `ERES`, `EWRES`
: Monte-Carlo generated diagnostics and are not linearized approximations like the other diagnostic types. EWRES is the Monte-Carlo version of CWRESI.

`ECWRES`
: Monte-Carlo version of CWRES.

`NPDE`
: Monte-Carlo generated normalized probability distribution error.

`NPD`
: The correlated (or non-decorrelated) NPDE value.

`OBJI`
: Objective function values for each individual (same as given in the root.phi file). The sum of the individual objective function values is equal to the total objective function.

In addition the list may include:

Data item labels.

Labels `ETA(1)`, `ETA(2)`, ... ,`ETA(10)`, ... , `ETA(70)`, etc.,
or alternatively, labels `ETA1`, `ETA2`, ... ,`ETA10`, ... , `ETA70`, etc., corresponding to eta(1), eta(2), etc.
The labels in the output will be ETA1, ETA2, ... , ET10, ... ET70, etc.

With NONMEM 7.3, a range of etas may be requested:
`ETAS(k:n)`
is equivalent to
`ETAk, ..., ETAn`
where n > k. LAST can be used in place of n, and requests the last (highest numbered) eta in the problem. E.g. `ETAS(1:LAST)`
With NONMEM 7.4, more flexible syntax is available:
The word `TO` may be used in place of ":".
The `BY` expression may be used:
`ETAS(1 TO 10 by 3)` prints out etas 1,4,7,10
`ETAS(LAST TO 1 by -3)` prints out etas 10,7,4,1 (assuming LAST=10)
A number list may be given:
`ETAS(1,5,12,4)` prints out etas 1, 5, 12, 4.
`ETAS(4:1)` prints etas 4, 3, 2, 1
`ETAS(4:1 by -2)` prints etas 4, 2
`ETAS(1:4 by -1)` prints etas 4, 3, 2, 1 (the by value sets the direction).

With NONMEM 7.4, a symbolic label specified in $ABBR REPLACE may be listed in $TABLE. For example:

```
$ABBR REPLACE ETA(CL)=ETA(1)
 ...
$TABLE ETA(CL)
```

Reserved positions of MODULE NMPRD4 (See **$ABBREVIATED**). `COM(k)` or :k denotes the kth reserved position. (There must be exactly 4 digits after ":". Use leading 0's as necessary.) E.g., `COM(3)` or `:003`.
Labels of the form :k will be used in the output.

Labels of PRED-defined items in MODULE NMPRD4 if abbreviated code is present (up to PDT distinct such labels in any one problem, for all tables and scatterplots. PDT is a constant in resource/SIZES.f90; default value is 500.) These may include labels of the NM-TRAN-defined items:

0nnn	e.g., 0010	stands for A00nnn
1nnn	e.g., 1010	stands for A01nnn
2nnn	e.g., 2010	stands for C00nnn
3nnn	e.g., 3010	stands for D00nnn
4nnn	e.g., 4010	stands for E00nnn
5nnn	e.g., 5010	stands for F00nnn
6nnn	e.g., 6010	stands for P00nnn

These may also include:

labels `VECTRA(1)`, `VECTRA(2)`, ... ,`VECTRA(9)`, or alternatively, labels `VA_1`, `VA_2`, ... ,`VA_9`, corresponding to VECTRA(1), VECTRA(2), ..., VECTRA(9).
The labels in the output will be VA_1, VA_2, ... , VA_9.
Similarly, for `VECTRB` and `VECTRC`.

Synonyms may be defined on either the $TABLE or $SCATTER record for special items `PRED`, `RES`, `WRES`; for PRED-defined items; for NM-TRAN-defined items; and for reserved positions of MODULE NMPRD4.

E.g., assume that IWRES is a PRED-defined label, that at least 3 positions of NMPRD4 are reserved, and that NM-TRAN has generated A00032 as the label for a derivative in the generated FSUBS routine. Either $TABLE or $SCATTER records may include:

```
WRES=RES1,IWRES=RES2,COM(3)=ABC,0032=DK.
```

For a discussion of the values of ETAs and PRED-defined items (e.g., are they based on initial or final values of theta? Simulated or zero or conditional values of eta?), see **values**.

Elements of G and H
E.g., `$TABLE G11 G21 G31 H11 H21`
The format is Gk1 or Hk1, where k is an integer value, e.g. 1-9 or 01-99 or 001-999. Gk1 requests the value of G(k,1), and Hk1 requests the value of H(k,1), where G and H are arguments of subroutine PRED. G(k,1) is the partial of F (the prediction) with respect to ETA(1), and H(k,1) is the partial of F with respect to EPS(1). (HH may be coded instead of H, but it is treated as if it were H.) A variable of the form Gk1 or Hk1 is not a reserved variable. If it is previously defined (i.e., if it is listed in $INPUT, or used on the left in abbreviated code, or used as a synonym e.g., $TABLE G11=COM(1)), then that definition of the variable is used, and there is no change from previous versions of NM-TRAN. Only if there is no other previous definition of the variable will it be understood to be an element of G or H.
What NM-TRAN actually displays is the variable in MODULE NMPRD4 that was used to compute the derivative of interest (a variable in the series A00nnn, C00nn, or D00nnn) with the appropriate synonym such as G11. If there is no such variable, this is an error. NMTRAN will not display variables that are not computed , e.g., G41 when there are only 3 etas in the problem, or when there are 4 etas but ETA(4) does not contribute to the

value of Y.
[There is a workaround if the zero is wanted as a place holder in the table. In abbreviated code ($ERROR or $PRED or $PK)
G41=0.
Now G41 may be listed in $TABLE or $SCATTER.]
This feature is designed so that the verbatim code in the "compute.cwres" R documentation is unnecessary.
E.g., Instead of:

```
  $ABB COMRES=5
  "LAST
  "  COM(1)=G(1,1)
  "  COM(2)=G(2,1)
  "  COM(3)=G(3,1)
  "  COM(4)=HH(1,1)  (or H(1,1) with $PRED)
  "  COM(5)=HH(2,1)  (or H(2,1) with $PRED)
$TABLE ID COM(1)=G11 COM(2)=G21 COM(3)=G31
 COM(4)=H11 COM(4)=H21
 IPRED MDV NOPRINT ONEHEADER FILE=cwtab1
```

Use only:

```
$TABLE ID G11 G21 G31 H11 H21
 IPRED MDV NOPRINT ONEHEADER FILE=cwtab1
```

(See **Displayed PRED-Defined Items**).

When tables are printed, the maximum number of labels permitted in list1 is 8; otherwise, it is PDT. (But see the NOAPPEND option.)

`list2`

A list comprised of one or more labels from list1. The rows of the table are sorted on the data items in list2. List2 may not appear when the number of labels in list1 is greater than 8. That is, a table with more than 8 data items also may not be sorted.

`NOSUB=[0|1]`

With `NOSUB=0`, label substitution will be performed for final estimates in table files. (See **$ABBREVIATED**). This is the default. With `NOSUB=1`, label substitution will not be performed.

`list3`

A list comprised of one or more items that are permitted in list1, e.g., data item labels and labels of PRED-defined items in MODULE NMPRD4. They follow option `EXCLUDE_BY`. Labels in list3 are called exclusion variables. If one or more of them have a non-zero value for a given data record, the row of the table corresponding to the data record will be excluded from the table file. Exclusion variables are not listed in the table file. They have no effect on the printed table or scatters in the NONMEM output, e.g., they do not cause any rows to be deleted from the printed table and are displayed in the printed table.

`PARAFILE`=filename

Weighted residuals are evaluated before the first $TABLE record. As of NONMEM 7.4, this computation is parallelized if parallelization is on when the first Table Step is implemented.

`PARAFILE`=filename specifies a different parafile than was used for the previous step.

`PARAFILE=ON` turns on parallelization for the weighted residuals.

PARAFILE=OFF turns off parallelization for the weighted residuals.
The PARAFILE option may be specified on any $TABLE record, but applies to all $TABLE records.

ESAMPLE=n_1
n_1 specifies the number of random samples used to calculate the Monte-Carlo diagnostics. Should be specified only on the first $TABLE record. Default is 300.

WRESCHOL (NM73)
Use the Cholesky square root of the variance, rather than the eigenvalue square root, when computing weighted residuals. Should be specified only on the first $TABLE record. This can speed up the Table Step when there are a large number of observations per individual.

SEED=n_2
n_2 specifies the starting seed for generating the Monte-Carlo diagnostics. Should be specified only on the first $TABLE record. Default is 11456.

CLOCKSEED=[0|1] (NM75)
If CLOCKSEED=1 (default is 0), actual starting seed will be 10000*(seconds after midnight)+SEED. This allows a control stream to produce different stochastic results for automated replications, without the need to modify the seed value in the control stream file in each replication.

RANMETHOD=[n|S|m]
n: the random number generator used for the Monte-Carlo simuations of weighted residual items.
0: ran0 of reference [5], minimal standard generator
1: ran1 of reference [5], Bays and Durham.
2: ran2 of reference [5].
3: ran3 of reference [5], Knuth. (Default)
4: NONMEM's traditional random number generator used in $SIMULATION
S: sobol sequence
m: the type of scrambling desired
0: no scrambing (S0 is the same as S)
1: Owen type scrambling
2: Faure-Tezuka type scrambling
3: Owen plus Faure-Tezuka type scrambling.
See the description of RANMETHOD for $ESTIM. Among the Sobol sequence methods, the S2 method appears to provide the least biased random samples, that is nearly uniform distribution, with good mixing in multi-dimensional spaces.
See **INTRODUCTION TO NONMEM 7, Reference [5]**
See **INTRODUCTION TO NONMEM 7, Monte Carlo Importance Sampling EM**
RANMETHOD should be specified only on the first $TABLE command. The RANMETHOD set in the $TABLE command does not propagate to $EST or $CHAIN.

Options PRINT, NOPRINT, HEADER, NOHEADER, NOLABEL, NOTITLE, FILE, FIRSTONLY, LASTONLY,FIRSTLASTONLY, FORWARD, NOFORWARD, APPEND, NOAPPEND, FORMAT, VARCALC apply to the individual $TABLE record. They must be specified for each table to which they apply.

PRINT
A printed table appears in the NONMEM output. This is the default.

NOPRINT
No printed table appears in the NONMEM output.

FILE=filename
The table is written to the given file in character form, e.g., ASCII or EBCDIC, according to the hardware platform. Filename may not contain embedded spaces. If it contains commas, semicolons, or parentheses, then it must be surrounded by single quotes ' or double quotes ". Filename may also contain equal signs if it is enclosed in quotes. Filename may contain at most 71 characters. If filename is the same as any option of the $TABLE record, it must be enclosed in quotes. Filename can differ between $TABLE records.
Default: No table file is output. Required with NOPRINT.

NOHEADER
Used only with the FILE option. No header lines are included in the table file.

ONEHEADER
Used only with the FILE option. Only the first line of the table is a header line.

ONEHEADERALL (NM74)
Used only with the FILE option and FORWARD. Only the first line of the table file is a header line. May also be coded ONEHEADERPERFILE.

NOLABEL
Used only with the FILE option. Do not print column labels. It may be combined with ONEHEADER to print only the title at the beginning of each table.

NOTITLE
Used only with the FILE option. Do not print table titles. It may be combined with ONEHEADER to print only the column labels at the beginning of each table. NOLABEL NOTITLE is equivalent to NOHEADER.

FIRSTONLY
Only information corresponding to the first data record from each individual record appears in the table. May also be coded FIRSTRECORDONLY or FIRSTRECONLY.

LASTONLY
Only information corresponding to the last data record from each individual record appears in the table. May also be coded LASTRECORDONLY or LASTRECONLY.

FIRSTLASTONLY
Only information corresponding to the first and last data record from each individual record appears in the table.

NOFORWARD
Used only with the FILE option. When the table file is opened during a given (sub)problem, it is positioned at the start of the file. This is the default. However, when there are multiple $TABLE records within the same problem and having the same filename, the situation is a little more complicated; see the text describing the FORWARD option.

FORWARD
Used only with the FILE option. When a table file is opened during a given (sub)problem, it is forwarded to the end of the file. This allows a table file to accumulate tables from multiple subproblems and superproblems. Moreover, if in the same (sub)problem the $TABLE record is followed by a contiguous succession of additional $TABLE records having the same filename as the given record, then even

though some of these additional records may have the NOFORWARD option, or have neither the NOFORWARD nor the FORWARD options, the FORWARD option will apply to each of the records in the succession.

APPEND

Items DV, PRED, RES, WRES appear automatically as the last 4 columns of the table. This is the default.

NOAPPEND

Requests that items DV, PRED, RES, WRES not appear automatically. When this is specified, the number of labels (i.e., user-chosen item types) that may appear in the table can be as large as 12 (rather than 8) for a printed table, and as large as PDT=4 (rather than PDT) for a table file. If items PRED, RES, and/or WRES are explicitly coded in list1, then they appear in the table, exactly as listed. (Previously to NONMEM VI 2.0, they could be included in the list, but were suppressed from the portion of the table described by list1 in favor of the automatically-generated items.)

FORMAT=*s*

This option defines the delimiter and number format for table files. It affects table files until a different FORMAT is specified. *s* defines the delimiter [,|s(pace)|t(ab)] followed by a Fortran format specification. The default is s1PE11.4. There are many more options for FORMAT.
For more details, see the format help item:
(See **format**).

Alternately, use LFORMAT and/or RFORMAT.
See **INTRODUCTION TO NONMEM 7, FORMAT=s1PE11.4**

LFORMAT=*s*

Specifies the format of the full label record of a table. Allows different formats for different columns. Sample:

```
LFORMAT="(4X,A4,4(',',4X,A8))"
```

RFORMAT=*s*

Specifies the format of the full numeric record of a table. Allows different formats for different columns. Sample:

```
RFORMAT="(F8.0,,4(',',1PE12.5))"
```

Multiple LFORMAT options and RFORMAT options may be present and will be concatenated. The format specifications are not checked by NMTRAN. If either is invalid, the run will fail in NONMEM. Both LFORMAT and RFORMAT affect table files until a different format is specified. Use LFORMAT="NONE" or RFORMAT="NONE" to resume use of the default format (which may have been specified by the FORMAT option) in a subsequent problem.

IDFORMAT=*s*(NM75)

By default the ID column has the same format as specified by FORMAT. IDFORMAT specifies the format for the ID column in table files. If an improper format is given, it defaults to that of FORMAT. Some examples:

IDFORMAT=I

Integer value, left adjusted in the field.

IDFORMAT=I6

Integer value, right adjusted in the first 6 characters of the field.

IDFORMAT=F6.1

Floating value, with single digit to the right of the decimal.

VARCALC=[0|1|2|3](NM74,NM75)

To report standard errors associated with etas (individual parameters) in the tables for user-defined variables, set $TABLE ...VARCALC=1. See setest.ctl in the examples directory. This appends an item named item_SE following each user-defined item in list1. If using RFORMAT formatting option, make sure to allow enough format fields to include reported standard errors. In addition, full variances-covariances among all user-defined variables and PREDPP parameters will be outputted to file root.vpd (the FORMAT used for this file is that defined in the $EST statement). With NONMEM 7.5, file root.vpt is also created. This contains variance-covariances associated with thetas (or omegas and sigmas) as well as those associated with etas/individual parameters. To append the comparable total standard errors item_SE to tables, set $TABLE ...VARCALC=3. To only create the vpd and vpt files, and not report SE's to the table, set VARCALC=2. This option must be recoded for each $TABLE record for which SE's are wanted. If VARCALC=1 or 2 or 3 is requested at least once among any of the tables, then the variance items are written to the vpd and vpt files. VARCALC=0 requests neither SE's nor vpd nor vpt, and is the default. Values of COMACT, which identify copying passes, may need to be tested in abbreviated code when this feature is used.

See **INTRODUCTION TO NONMEM 7, Requesting Standard Errors to User-Defined and PREDPP Parameters**

FIXEDETAS= (list) (NM74)

Specified etas may be treated as if they are fixed effects when evaluating population diagnostics during the $TABLE step. This is particularly suitable for super-ID $LEVEL etas that span groups of subjects, as if they were a fixed effect when evaluating populations characteristics during the $TABLE step, such as PRED, CWRES, NPDE, etc. In this way, the PRED evaluated will be, not of the total population, but of a given site level for that subject. List is a number-list of etas. For example, FIXEDETAS=(3-6)

A number-list may contain a single integer, a range of integers (with -), or a series of integers and ranges separated by comma.

NPDTYPE=1

The strict stochastic (Monte Carlo) method over the data y domain as well as etas is implemented for NPD diagnostics.

NPDTYPE=0

An asymptotic assessment of the residual variability is used. This is the default.

UNCONDITIONAL

The Table Step is always implemented. This is the default.

CONDITIONAL

The Table Step is implemented only when the Estimation Step terminates successfully or is not implemented.

OMITTED

The Table Step is not implemented.

REFERENCES: Guide IV, section III.B.16
REFERENCES: Guide V, section 9.5.1 , 10.7.1

$THETA

MEANING: Gives initial estimates and bounds for thetas
CONTEXT: NM-TRAN Control Record

USAGE:

```
$THETA value1 [value2] [value3] ...
      [(value)xn] ...
      [label=value] ...
      [NAMES (label ...)value ...]
      [NUMBERPOINTS=n]
      [ABORT|NOABORT|NOABORTFIRST]
```

SAMPLE:

```
$THETA  (0,3) 2 FIXED (0,.6,1) 10 (-INF,-2.7,0) (37 FIXED)
```

DISCUSSION:
Gives initial estimates and bounds for elements of the THETA matrix. Thetas are numbered in the order in which they are defined.

OPTIONS:

Each value defines a theta and gives its initial estimate and bounds. A value has one of 4 forms:

init [FIXED]
: Init is the initial estimate. If FIXED is used, the final parameter estimate is to be constrained to equal the initial parameter estimate.

([low,] init [,up] [FIXED])
: Low and up are lower and upper bounds respectively. They are the boundaries for the minimization search. Commas are optional. If an upper bound is used, a lower bound must also be used (e.g., -INF; see below). The lower and upper bounds (or if just the lower bound is used, then just this bound) may be omitted, in which case this form differs from the one described above only in so far as with this form, the initial estimate and the FIXED attribute are enclosed in parentheses. When FIXED is used, and a bound appears the bound must equal the initial estimate. FIXED is implied when all three values are equal. The lower bound can be -INF (i.e., -infinity), and the upper bound can be INF (i.e., +infinity). These are the defaults for lower and upper bounds. They are communicated to NONMEM as numeric values -1000000 and 1000000.

([low,] init [,up]) [FIXED]
: This is just like the form described above except that if the FIXED attribute is used, the attribute occurs outside the parentheses, and then if moreover, a bound appears, the bound need not equal the initial estimate.

(low,,up)
: The commas are required. Because no initial estimate is given, a search for an initial estimate (the Initial Estimates Step) is undertaken by NONMEM. With NONMEM 7.4, when initial thetas are to be estimated, evaluations can now be done for FOCE and Laplace, not just for FO.

(*value*)x*n*
: Any initial value or group of initial values may be enclosed in parentheses and followed by "x*n*", which means to replicate the values within parentheses *n* times

("repeated value"). The values within the parenthesis may have any of the above forms. For example, the following two are equivalent:

```
$THETA 2 2 2 2 (0.001,0.1,1000) (0.001,0.1,1000) (0.001,0.1,1000)
       (0.5 FIXED) (0.5 FIXED)
$THETA (2)x4 (0.001,0.1,1000)x3 (0.5 FIXED)x2
```

UNINT (NM75)

UNINT is used during the Optimal Design Step to identify a theta as uninteresting. UNINT may be used anywhere that FIXED may be used.

label=*value* [FIXED] (NM75)

This is a compact method of defining an element of theta, specifying its initial estimate, and specifying a symbolic subscript for this element of THETA. The symbolic subscript may be used for THETA in abbreviated code, and will also identify this element of THETA in the NONMEM output. (Only the first 9 characters of the label will appear). If new $THETA records change the ordering, the abbreviated code does not have to be changed. For example, suppose the third element of THETA that is defined happens to be

```
$THETA CL=(0.0,7.0)
```

The abbreviated code can use this symbolic subscript instead of the numeric subscript, e.g.,

```
TVCL=THETA(CL)
```

The NONMEM report will describe the relationship, e.g.,
LABELS FOR THETAS THETA(3)=THETA(CL)
and THETA(CL) rather than TH 3 will appear in the NONMEM report.

It is also possible to specify the label using the

```
$ABBR REPLACE THETA(CL)=THETA(3)
```

control record but the NONMEM report will not identify the relationship and will not generate the symbolic label THETA(CL) in the report. For dynamic, or implicit mapping of labels, such as for various occasions, these still need to be done via the $ABBR REPLACE record.

$THETA records must be placed ahead of any records that use the symbolic label. Therefore these records must be placed before $THETAI, $THETAR, and abbreviated code.

`NAMES` (*label ...*)*value ...* (NM75)

A compact way of defining one or more thetas with labels and initial values. For example

```
$THETA NAMES(V1,CL,Q,V2) (0.0,7.0) (0.0,7.0) (0.0,7.0) 7
```

`NUMBERPOINTS=n`

During NONMEM's search for an initial estimate, a number of points will be examined. This number will be automatically determined by NONMEM, or it can be specified with this option. May also be coded `NUMBERPTS`, `NUMPOINTS`, `NUMPTS`.

`ABORT`

During the Initial Estimates step, NONMEM is to abort when PRED sets the error return code to 1. (The PRED error return code n is set by the statement "`EXIT` n [k]" in abbreviated code, or by the statement `IERPRD=n` in user-supplied code, or by PREDPP when it detects an error.) This is the default.

`NOABORT`

During the Initial Estimates step, NONMEM is simply to ignore values of the theta

vector that result in PRED error return code 1. (Ordinarily the first value of the theta vector is never ignored.) These will not be feasible values for an initial estimate.

`NOABORTFIRST`
Same as NOABORT option, but also applies to the first value of the theta vector that is tried. It cannot be shortened; all characters must be coded. May be used with the NOABORT option, in which case the stronger condition (NOABORTFIRST) takes precedence.

REFERENCES: Guide IV, section III.B.9

$THETAI

MEANING: Gives Instructions for Transforming Initial Thetas
CONTEXT: NM-TRAN Control Record

USAGE:
`$THETAI` Fortran statements

SAMPLE:

```
$THI
THETA(1:NTHETA)=LOG(THETAI(1:NTHETA))
THETAP(1:NTHP)=LOG(THETAPI(1:NTHP))

THETA=LOG(THETA)
```

DISCUSSION:

The purpose of $THETAI is to transform the initial values in the $THETA and $THETAP records. The record name may also be coded as $THI.

In the above sample code, it is desired that the thetas by estimated within NONMEM in the log domain, but the user wants the convenience of inputting and outputting them in the natural domain, such as when performing linear MU referencing.

The $THETAI record will convert any initial thetas in a $THETA record, or thetas obtained from a chain file, but will not convert thetas from an MSF file. The variance to the theta priors will be appropriately converted when using $PRIOR NWPRI.

The assignment statements may be any Fortran 95 statements. They are copied unchanged to subroutine SUBROUTINE THETAISUB in FSUBS (also found in thetair.f90).

They may include array assigment statements specifying the whole arrays or sections of arrays.

If the initial estimate for an element of theta is transformed, so is the upper and lower bounds for that theta, if any.

Arguments of the subroutine are as follows.

THETAI
: Values of THETA specified on $THETA records. Input.

THETAPI
: Values of THETA specified on $THETAP records (or, if the informative names are not used, thetas corresponding to priors, if any). Input.

THETA
: New values of THETA. Output.

THETAP
: New values of THETA's for priors. Output.

Other reserved variables that may be used are as follows:

NTHETA
: Number of thetas to be estimated.

NTHP
: Number of theta priors.

NPROB IPROB

These can be tested in IF statements so that values may be assigned diffently for different problems.

If the range is not specified, NONMEM to supply the range (which is by default NTHETA+NTHP).

When appropriate, for reporting thetas, the inverse function should be supplied, e.g., with the samples above:

```
$THR
THETAR(1:NTHETA)=EXP(THETA(1:NTHETA))
THETAPR(1:NTHP)=EXP(THETAP(1:NTHP))
or
THETAR=EXP(THETA)
```

Note that the assignment occurs after the NONMEM control stream has been processed, so that errors in assignment of THETA's are not found. E.g.

```
$THI
THETA=0.
```

This will set all theta's to 0, and there will be no specific error message from NONMEM, though most likely the run will fail.

If initial estimates of all or part of THETA are omitted, NONMEM performs a search for missing initial estimates as its first task. (These are printed on a separate page under the heading "`INITIAL PARAMETER ESTIMATE`"). This search occurs before the transformation by $THETAI.

Values of INITIAL ESTIMATE OF THETA in NONMEM output are those from the $THETA/$THETAP records. Values of THETA in the root.ext files are as set in $THI.

$THI may be used with $THR, but not necessarily.

Another example is rescaling thetas. E.g., suppose in CONTROL5

```
$THETA  (.1,3,5) (.008,.08,.5) (.004,.04,.9)
```

is replaced with

```
$THETAI
THETA=THETAI/10.
$THETA  (1,30,50) (.08,.8,5) (.04,.4,9)
```

The results will be identical, except for the values of INITIAL ESTIMATE OF THETA in the NONMEM report. The values in .ext will not be affected.

REFERENCES: Guide Introduction_7

$THETAP,$THETAPV

MEANING: Gives prior information for thetas
CONTEXT: NM-TRAN Control Record

USAGE:
`$THETAP` *value*$_1$ [*value*$_2$] [*value*$_3$] ...
`$THETAPV` *value*$_1$ [*value*$_2$] [*value*$_3$] ...

SAMPLE:

```
; Prior information of THETAS (NTHP=4 of them)
$THETAP (2.0 FIX) (2.0 FIX) (2.0 FIX) (2.0 FIX)
; Variance to prior information of THETAS (NTHPxNTHP=4x4 of them).
$THETAPV BLOCK(4)
10000 FIX
0.00 10000
0.00  0.00 10000
0.00  0.00 0.0 10000
```

DISCUSSION:

These are called the informative forms of the $THETA record, for use with the NWPRI utility.

$THETAP gives prior information for elements of the THETA matrix.
$THETAPV gives variance information for THETA priors.

The name of the record describes the kind of information it gives, rather than the structure of the information. E.g., in the example above, $THETAPV is implemented in FCON with an $OMEGA record because it is in general an array of values. These records may be located anywhere in the control stream. NM-TRAN inserts the corresponding records in the control stream in the correct order. When the informative forms are used, the options of $PRIOR NWPRI need not be specified. However, if options are listed in $PRIOR NWPRI, then these values are chosen over what is surmised from the informatively labeled theta/omega/sigma records.
(See **nwpri**).

OPTIONS:

The option FIXED should be used. Other appropriate options are BLOCK and VALUES.

REFERENCES: Guide Introduction_7

$THETAR

MEANING: Gives Instructions for Transforming Final Thetas
CONTEXT: NM-TRAN Control Record

USAGE:
`$THETAR` Fortran statements

SAMPLE:
```
$THETAR
THETAR=EXP(THETA)
Or
$THETAR
 THETAR(1:NTHETA)=EXP(THETA(1:NTHETA))
 THETAPR(1:NTHP)=EXP(THETAP(1:NTHP))
```

DISCUSSION:

The purpose of $THETAR is to transform the final theta values for the NONMEM report and additional output files. The record name may also be coded as $THR.

In the above sample code, it is assumed that $THETAI was used to transform thetas from the natural domain to the log domain for estimation within NONMEM, such as when performing linear MU referencing. $THETAR causes them to be output in the natural domain as well.

This affects the following data in the output report:
```
NPARAMETR
FINAL PARAMETER ESTIMATE for THETA.
FIRST ORDER STANDARD ERROR OF ESTIMATE
 THETA - VECTOR OF FIXED EFFECTS PARAMETERS
FIRST ORDER COVARIANCE MATRIX OF ESTIMATE
 THETA - VECTOR OF FIXED EFFECTS PARAMETERS
```

Contents of additional output files such as .coi and .cov and .cor are also affected.

The assignment statements may be any Fortran 95 statements. They are copied unchanged to subroutine SUBROUTINE THETARSUB in FSUBS (also found in thetair.f90).

They may include array assigment statements specifying the whole arrays or sections of arrays.

Arguments of the subroutine are as follows.

THETA
: Final estimates of theta. Input.

THETAP
: Final estimates of thetas specified on $THETAP records (or, if the informative names are not used, thetas corresponding to priors, if any). Input.

THETAR
: New values of THETA. Output

THETAPR
: New values of THETA's for priors. Output.

Other reserved variables that may be used are as follows:

NTHETA
Number of thetas to be estimated.

NTHP
Number of theta priors.

NPROB IPROB
These can be tested in IF statements so that values may be assigned diffently for different problems.

If the range is not specified, NONMEM to supply the range (which is by default NTHETA+NTHP).

$THR may be used with $THI, but not necessarily.

Another example is rescaling thetas. E.g., suppose that

```
$THETAI
THETA=THETAI/10.
```

was used to rescale thetas on input.

Then

```
$THETAR
THETAR=THETA*10.
```

can be used to report them in the original domain.

REFERENCES: Guide Introduction_7

$TOL

MEANING: Marks the beginning of abbreviated code for the TOL routine
CONTEXT: NM-TRAN Control Record

USAGE:
```
$TOL
abbreviated code
```

DISCUSSION:
The $TOL record is used to specify compartment-specific NRD values. It is used with PREDPP's general non-linear models (ADVAN6, ADVAN8, ADVAN9, ADVAN13, ADVAN14, ADVAN15, ADVAN16, ADVAN17, ADVAN18, and SS6 and SS9). NRD stands for "Number of Required Digits," although the precise meaning depends on the particular ADVAN or SS routine that uses it.

$TOL cannot be present if the $SUBROUTINES record includes any of the corresponding options.

The following are equivalent:
```
$SUBROUTINES ... TOL=n
```
and
```
$SUBROUTINES ...
$TOL NRD=n
```

Left-hand quantities:

These are identical to the left-hand variables that may be set in subroutine TOL. If subscript "(I)" is omitted, it defaults to 1.

`NRD(I)`
: The number of digits that are required to be accurate in the computation of the drug amount in compartment I, i.e., the relative tolerance. ADVAN 9, 13, 16, 17, and 18 have the capability of using different values of NRD for different compartments. For compartments not specified, the tolerance of the last compartment specified will be used.

  However, all the other ADVAN routines requiring TOL take the relative tolerance to be the same for all compartments; NRD(I), $I > 1$, is ignored, and only NRD(1) is used. With NONMEM 7.4, NRD(0) is the relative tolerance for the Steady State computations. If NRD(0) is not specified, NRD(1) is used.

  The value of NRD(1) can also be specified using $SUBROUTINES option TOL.
  The value of NRD(0) can also be specified using $SUBROUTINES option SSTOL.

`ANRD(I)` (NM74)
: The absolute tolerance in the computation of the drug amount in compartment I. The default is 12 (that is, accuracy is 10**(-12)). Used by ADVAN 9, 13, 14, and 15, 16, 17, and 18 which have the capability of using different values of ANRD for different compartments. For compartments not specified, the tolerance of the last compartment specified will be used.

  ANRD(0) is the absolute tolerance for the Steady State computations. If ANRD(0) is not specified, ANRD(1) is used.

  The value of ANRD(1) can also be specified using $SUBROUTINES option ATOL.
  The value of ANRD(0) can also be specified using $SUBROUTINES option SSATOL.

`NRDC(I)` (NM74)

Same as NRD(I), but used for the FOCE/LAPLACE covariance step. Used with ADVAN 9, 13, 14, 15, 16, 17, and 18. If not set, NRDC defaults to the value of NRD. NRDC(0) is used for Steady State computations during the FOCE/LAPLACE covariance step.

The value of NRDC(1) can also be specified using $SUBROUTINES option TOLC.
The value of NRDC(0) can also be specified using $SUBROUTINES option SSTOLC.

`ANRDC(I)` (NM74)

Same as ANRD(I), but used for the FOCE/LAPLACE covariance step. Used with ADVAN 9, 13, 14, 15, 16, 17, 18. If not set, ANRDC defaults to the value of ANRD. ANRDC(0) is used for Steady State computations during the FOCE/LAPLACE covariance step.

The value of ANRDC(1) can also be specified using $SUBROUTINES option ATOLC.
The value of ANRDC(0) can also be specified using $SUBROUTINES option SSATOLC.

Right-hand quantities:

Integers, e.g., `NRD(1)=4`

Forbidden Variable Names:

All other variables.

RECORD ORDER:

Follows $SUBROUTINES and $MODEL

REFERENCES: Guide IV, section V.C.3 , V.C.10
REFERENCES: Guide VI, section VI.D , Figure 41

$TTDF (NM75)

MEANING: Specifies t-distribution degrees of freedom for theta
CONTEXT: NM-TRAN Control Record

USAGE:

```
$TTDF 0 (default)
$TTDF [value ... ]
$TTDF [(value)[xn]] ...
```

SAMPLE:

```
$TTDF (2.0)x2 (3.0)x2
```

sets t-distribution degrees of freedom for thetas 1 and 2 to value of 2, and degrees of freedom for thetas 3 and 4 to value of 3.0.

DISCUSSION:
The ttdf is a separate record that allows the user to specify the t-distribution degrees of freedom for each theta. It can be used with $ESTIMATION METHOD=NUTS, and with $SIMULATION. By default $TTDF value is 0. These values will be used in estimation and simulation unless $EST TTDF or $SIM TTDF is set, respectively, which set the t degrees of freedom to a single TTDF value for all thetas for that step.

REFERENCES: Guide IV, section V.C.3 , V.C.10
REFERENCES: Guide VI, section VI.D , Figure 41

$WARNINGS

MEANING: Control Display of NM-TRAN Warning, Data Warning and Data Error messages

CONTEXT: NM-TRAN Control Record

USAGE:

```
$WARNINGS [NONE] [n] [RESET | NORESET ]
          [WARNINGMAXIMUM= [ NONE | n | (list) ] ]
          [DATAMAXIMUM= [ NONE | n | (list) ] ]
          [ERRORMAXIMUM=n ]
```

SAMPLE:

```
$WARNING      WMAX=1,EMAX=9999
```

DISCUSSION:

Optional. Controls the display of NM-TRAN warning messages and data error messages in the current and all subsequent problems until another $WARNING record appears. May appear anywhere in the control stream following $SIZES, $SUPER (if present) and $PROBLEM records, but cannot affect the display of warning messages associated with the records that precede it.

OPTIONS:

`NONE`

Suppress all warning messages.

n Sets `WARNINGMAXIMUM`=n, `ERRORMAXIMUM`=n, `DATAMAXIMUM`=n. May be 0, in which case it is the same as `NONE`.

`LIST`

Print a list of all possible warning messages. When this option is present, the control stream must consist **only** of the single record:

```
$WARNING LIST
```

`WARNINGMAXIMUM`

Refers to general (non-data) warning messages. May also be coded `WARNMAXIMUM` and shorter substrings, e.g., `WMAX`, `WARN`. May appear more than once, with cumulative effect.

`WARNINGMAXIMUM=NONE`

Suppress warning messages.

`WARNINGMAXIMUM=`*n*

Each type of warning message will be displayed no more than n times ($n \geq 0$). The default is 20.

`WARNINGMAXIMUM=` (list)

The list is a series of integers n_1, n_2, n_3 ... ($n_i \geq 0$). n_i gives the the maximum number for the *i*th type of warning message.

`NONE` may coded for any n_i if it is the last in the list. Sets the maximum number of messages for the *i*th and all higher-numbered messages to zero.

Example:

```
$WARNING      WMAX=1
```

Sets the maximum for all non-data messages to 1.

DATAMAXIMUM
: Refers to data warning messages. May also be coded DMAXIMUM and shorter substrings, e.g. DMAX, DATA. May appear more than once, with cumulative effect. Forms and default are identical to those for the WARNINGMAXIMUM option.

ERRORMAXIMUM=*n*
: Refers to data error messages. It can also be coded ERRMAXIMUM or EMAXIMUM, and shorter strings, e.g., EMAX. Each type of data error message will be displayed no more than n times ($n \geq 0$). The default is 20. Data error messages can never be totally suppressed. With $n = 0$ and $n = 1$, NM-TRAN displays the first data error message and then terminates.

  Example:

```
$WARNING EMAX=99999
```

  Sets the maximum number of data error messages displayed from 20 to a much higher value, so that NM-TRAN will not terminate until all data error conditions have been identified.

RESET
: All counts reset to 0 with each new problem. This is the default.

NORESET
: Counts of how many times a given message has been displayed are accumulated over problems. (A new $WARNING record always resets all counters, but they will not be automatically reset again with subsequent problems.)

A $WARNING record with no options re-establishes all default conditions as if no $WARNING records had been present.

REFERENCES: None.

ABBREVIATED CODE

MEANING: FORTRAN-like statements
CONTEXT: Abbreviated code

SAMPLE:
```
$PK     CL=THETA(1)+ETA(1)
```

DISCUSSION:
Abbreviated code is FORTRAN-like language within an NM-TRAN control stream that is used to specify a model. The $PRED, $PK, $ERROR, $DES, $AES, $AESINITIAL, $TOL, $MIX, and $INFN records each begin a block of abbreviated code. Such a block ends with either the appearance of another record beginning with a "$", or with the absence of any further records in the control stream.

The general rules are:

One statement of abbreviated per line, starting anywhere on the line.
No statement numbers (unlike with FORTRAN).
Comments may be included on any line after the semicolon character ";".
No statement types other than assignment,
`IF`, `THEN`, `ELSE`, `ELSEIF`, `ENDIF`, `DO WHILE`,
`ENDDO`, `CALL`, `WRITE`, `PRINT`, `RETURN`, `OPEN`, `CLOSE`, `REWIND`.
(e.g., no `GOTO`, `READ`, `FORMAT`.)
With NONMEM 7.2 and higher, both lower and upper case may be used for all user-defined and reserved words.
Continuation lines are permitted using the character & at the end of the line.
A special statement type, `EXIT`, is permitted.

ASSIGNMENT STATEMENTS

An assignment statement (A=B) uses right-hand quantities in an expression B in order to define a quantity A.

Permissible right-hand quantities are:
previously defined quantities, including declared variables that have also been used on the left side of an assignment statement.
(See **$abbreviated**).
constants, e.g 1, 1.1, 3E+1 3E1, 3E01, 3E-1 3E-01, 3D+1, 3D1, 3D01, 3D-1 3D-01
THETA, OMEGA and SIGMA elements, e.g. `THETA(1)`, `OMEGA(2,2)`. When the two subscripts for an OMEGA or SIGMA element are the same, only one subscript is needed, e.g. `OMEGA(2)`.
labels of data items defined in the $INPUT record
variables `ETA(n)`, `EPS(n)`, `ERR(n)`, which are understood to have mean 0 and variances and covariances given by elements of the OMEGA and SIGMA arrays.
operators `+ - * / **`
parentheses ()
built-in functions: `LOG` (natural log), `LOG10`, `EXP`, `SQRT`, `SIN`, `COS`, `ABS`, `TAN`, `ASIN`, `ACOS`, `ATAN`, `INT`, `MIN`, `MAX`, `MOD`
NONMEM functions: `PHI`, `GAMLN`

FORTRAN functions may have random variable arguments and the appropriate partial derivatives are computed. The partial derivative of ABS(X) with respect to η is mathematically undefined at X=0. We are arbitrarily defining it to be

dX/*deta*. If the prediction depends on X, X must always be either positive or negative. If the argument of GAMLN is a random variable, it must always be positive. Function PHI may have a random argument but no partial derivates are computed.

The INT, MOD, MIN, and MAX functions produce discontinuous results. No partial derivatives are computed. If they are used outside of a simulation block and the function value affects the value of the objective function, then an error in the NONMEM Estimation Step will probably occur. If they are needed in the computations of DADT (differential equations; with ADVAN 6, 8 ,9, 13, 14, 15, 16, 17, 18), then the funcions should be used instead in the PK routine, and model event time (MTIME) parameters be used to set flags for DES.
(See **Model Time examples**)
(See **Cirdadian example**)

abbreviated functions (see below)
subscripted variables (see below)

With previous versions of NONMEM, constants were limited to 12 characters. With NONMEM 7 a constant may be up to 30 characters. (The length is specified by constant SCO in SIZES)

Further information about Assignment statements:

Parentheses may be nested in the right-hand side of an assignment statement.

Except in certain blocks of abbreviated code (i.e. a simulation block, an initialization block, a finalization block or a pred_ignore_data block), the variables `ETA(n)`, `EPS(n)`, `ERR(n)` are considered to be (the base) random variables, and quantities defined in terms of these variables (directly or indirectly) are themselves considered to be random variables. NM-TRAN sets up computations for derivatives of random variables with respect to the base random variables.

Left-hand variables that have no reserved meaning are called user-defined variables. With previous versions of NONMEM they were restricted to 8 characters in length and could not include the character _. With NONMEM 7 a user-defined variable name consists of 1-20 letters (A-Z), numerals (0-9), and the character '_', beginning with a letter. (The length 20 is specified by constant SD in SIZES)
(See **SIZES**).

With NONMEM 7, variables MU_1, MU_2, etc., are reserved and are used for mu modelling.
See **INTRODUCTION TO NONMEM 7, MU Referencing**
(See **MU_Model**).

Left-hand quantities `VECTRA(n)`, `VECTRB(n)`, `VECTRC(n)` may be defined, and these become elements of pre-defined vectors `VECTRA`, `VECTRB`, `VECTRC`. If `VECTRA`, `VECTRB`, `VECTRC` is defined without a subscript, it is understood to be a simple variable. A subscript must be an integer constant, or a character string that is replaced by an integer constant using the $ABBR REPLACE feature. Whereas once it is defined, a vector element may be used as a right-hand quantity - in the same

way as any previously defined variable - the vector itself may be used only as an argument to an abbreviated function.

(See VECTORS and ABBREVIATED FUNCTIONS, below.)

Values may be assigned to certain special variables: RPTO, RPTON and PRDFL - with the repetition feature
(See **Repetition Variables**);
SKIP_ (but only in a finalization block) - with superproblems or subproblems
(See **SKIP**).

Variables, constants, and functions are all of double precision floating point type. (E.g., K=.5 gives K the value .5, not 0 as it would be under the FORTRAN convention that K is integer type.)

Except in certain blocks of abbreviated code (i.e. a simulation block, an initialization block, or a finalization block), a label in an $INPUT record may not appear on the left in an assignment statement.

There is a reserved storage area of module NMPRD4, and when defining the kth variable in this area, it may be referenced as `COM(k)`. Then, even if the variable is defined in terms of random variables, it is not regarded as being a random variable; derivatives of `COM(k)` with respect to the base random variables are not computed.

CONDITIONAL STATEMENTS

Conditional statements have one of two forms:

`IF` (condition) assignment statement

```
IF (condition) THEN
  abbreviated code ...
ELSEIF (condition) THEN
  abbreviated code ...
ELSE
  abbreviated code ...
ENDIF
```

See below for a restriction on `ELSEIF`.
Conditions may include the operators .EQ., .NE., .LE., .GE., .LT., .GT., .AND., and .OR.. The arthmetic operators may be coded as ==, /=, <=, >=, <, and >, respectively. They may include expressions that can be used as right-hand quantities with assignment statements. They may not include parentheses except in such expressions.
Valid: `IF (Q.EQ.(R+C)/D) ...`
Invalid: `IF (Q.EQ.R.AND.(C.GT.D.OR.E.EQ.F)) ...`

With $PRED, $PK, and $ERROR records, a condition may test the ICALL argument. However, if the test is `IF (ICALL.EQ.2)`, then `ELSE` may not be used. Special rules apply. For more detail, see SPECIAL STATEMENTS (below), and help entries for these records.

A condition may also test certain variables defined in modules
(See **Variables_in_modules**, **NONMEM_modules**, **PREDPP_modules**).

Abbreviated code in a particular THEN or ELSE clause may not contain multiple definitions of the same random variable. The code below is invalid:

```
IF (condition) THEN
X=...an expression involving a random variable
X= ....
ENDIF
```

Random variables cannot be defined within nested conditionals, i.e., within a conditional structure beginning with an IF and containing another IF. The use of ELSEIF ... THEN implies a nested conditional.

A special rule applies when random variables are defined via conditional statements. If a random variable is multiply defined within a series of IF ... THEN structures, but all conditions are false, then the value of the random variable is set to zero. If an ELSE appears, then not all conditions are false.
Consider two cases in which the following statements are the only ones defining TVK and K, respectively:

```
IF (WT.GT.0) TVK=THETA(1)*WT
```

If the condition is false, the non-random variable TVK retains the value set with the previous data record.

```
IF (WT.GT.0) K=THETA(1)*WT*EXP(ETA(1))
```

If the condition is false, the value of the random variable K is set to zero.
NM-TRAN prints a warning message when it detects such code.

In $PK, $ERROR, and $PRED records, recursion code may be used in an explicit manner, as in this example:

```
IF (WT.GT.0) THEN
  K=THETA(1)*WT*EXP(ETA(1))
ELSE
  K=K
ENDIF
```

If the condition is false, K retains its value set with the previous data record.

Recursion code can be used in $PRED, $PK, and $ERROR records for other purposes as well. The following two fragments of code illustrate how one can use abbreviated code to implement recursive kinetics in $PRED. The first example works with a single bolus dose and the second example works with single or multiple bolus doses. Similar code can be used in $PK and $ERROR.

```
K=THETA(1)*EXP(ETA(1))
IF (TIME.EQ.0) THEN
  OLDA=AMT
  T=TIME
ENDIF
A=OLDA*EXP(-K*(TIME-T))
```

```
OLDA=A
T=TIME

K=THETA(1)*EXP(ETA(1))
IF (TIME.EQ.0) THEN
  A=AMT
  T=TIME
ELSE
  A=A*EXP(-K*(TIME-T))+AMT
ENDIF
T=TIME
```

The above forms of recursion work for recursion from one data record to the next ("inter-record" recursion). It is also possible to use recursion in a do-while loop ("intra-record", or "do-while" recursion).

Example of a do-while recursive loop using a random variable:

```
TERM=THETA(1)*EXP(ETA(1))
SUM=0
DO WHILE(condition)
SUM=SUM+TERM
...
ENDDO
```

A product loop such as

```
PROD=PROD*TERM
```

is also possible, as are other ways the dowhile recursive variable can be used, so long as the variable appears on both sides of the equal sign within the DOWHILE loop: V= ... V ...

EXIT STATEMENT

The exit statement has three forms:

```
EXIT
EXIT n
EXIT n k
```

n is called the "PRED error return code"; (also called the "PRED exit code.") It must be 1 or 2. Default is 1.
k is the user error code. It may be omitted; if present, it must be integer-valued in the range 0-999. With NONMEM 7.5, it may also be in the range 1000-9999 when issued during the Simulation Step. Default is 0.

With all versions of NONMEM, the value of k is part of the error message in ETEXT, which is reported in the NONMEM output report and in file PRDERR. With NONMEM 7.5 and later, k tells NONMEM how to handle EXIT statements during the Simulation Step.
(See **Simulation block**).
(See **nmprd1**)

The EXIT statement causes an immediate exit from the routine and, if PREDPP is being used, a subsequent immediate exit from PREDPP, with a return to NONMEM. It is typically used in an IF statement to avoid further computation of the users code when the values of theta/eta's set by NONMEM are inappropriate or would lead to an arithmetic exception. If such an exit occurs during a Covariance, Table or Scatterplot Step, or during computation of the initial value of the objective function, NONMEM will abort. If the exit occurs during an Estimation or Initial Estimates Step, NONMEM's action depends on the error return code value:

n=1
Suppose first that a search for eta is not underway. Then a search for theta is underway, and when a search is underway for theta, the following happens: If the NOABORT option is used, NONMEM will try to continue using different values for theta (i.e. "theta-recovery"); otherwise, NONMEM aborts. If theta-recovery fails, NONMEM aborts.
Next, suppose a search for eta is underway. NONMEM will try to continue using different values for eta (i.e. "eta-recovery"). Suppose eta-recovery fails. If a search for theta is not also underway, NONMEM aborts. Otherwise, the above procedure regarding theta-recovery applies.
(See **Guide VI, section III.K**)

n=2
NONMEM aborts immediately.

If NONMEM aborts, and k>0, a user message such as the following is printed in the output:
"PK SUBROUTINE: USER ERROR CODE = k"
This message is intended to help the user distinguish which EXIT statement caused NONMEM to abort when more than one EXIT statement is present in the abbreviated code.

WRITE AND PRINT STATEMENTS

(See **write print**).

SPECIAL STATEMENTS

Optional. Each of these special statements is permitted in special blocks of abbreviated code.
(See **Initialization-Finalization block**, **Simulation block**).
(See **Expectation block**, **Data_Average block**).
(See **Data_ingnore block**).

```
CALL PASS(MODE)
```

Must be coded exactly as shown. May be used only in $PRED initialization-finalization blocks, and in $INFN records. If present, `MODE` becomes a reserved variable with type INTEGER, and may not be used outside the block(s).

```
CALL SUPP(ie,ic)
```

Must be coded as shown, with constant integer values 0 or 1 in place of ie and ic. A

value 1 for ie (ic) suppresses output from the Estimation (Covariance) step. A value 0 does not suppress the output from the step. May be used only in $PRED initialization-finalization blocks, and in $INFN records. The ie (ic) value remains in force until changed by a call to SUPP.

```
CALL RANDOM(n,R)
```

Must be coded as shown, with a constant integer value 1-10 in place of n. The value is the number of the random source. `R` is a random number from this source. May be used only in Simulation and Expectation blocks. If present, `R` becomes a reserved variable with type REAL, and may not be used outside the block(s).

```
CALL SIMETA(ETA)
```

Must be coded exactly as shown. May only be used in Simulation blocks. Note that NM-TRAN itself provides the minimum necessary call to SIMETA. This statement is used in abbreviated code only to obtain a different value of `ETA`, e.g., so that the eta distribution may be truncated:

```
DO WHILE (ETA(1).GT.5)
   CALL SIMETA(ETA)
ENDDO
```

```
CALL SIMEPS(EPS)
```

Must be coded exactly as shown. May only be used in Simulation blocks. Note that NM-TRAN provides the minimum necessary call to SIMEPS. This statement is used in abbreviated code only to obtain a different value of `EPS`, e.g., so that the distribution may be truncated (see SIMETA above).

```
DO WHILE( condition )
```

With NONMEM 7, DOWHILE may be used in all blocks of abbreviated code. DOWHILE marks the beginning of a set of statements that are executed repeatedly until the condition is false. The ending of the set of statements is marked by the statement `ENDDO`. The non-FORTRAN syntax `DO WHILE(DATA)` is permitted in $PRED initialization and finalization blocks, and in the $INFN record.
(See **Initialization-Finalization block**).

```
RETURN
```

May be used in all special blocks. RETURN statements must be used with caution because they by-pass certain normal final actions of the routine.

Another kind of special block is the pred_ignore_data block, which may be part of $PRED,$PK, or $INFN blocks. It uses the reserved variables PRED_IGNORE_DATA_TEST and PRED_IGNORE_DATA. By setting values of PRED_IGNORE_DATA=1 for certain data records, it causes these records to be dropped from the NONMEM data set. Ordinary Fortran syntax is used in such blocks.

(See **PRED_IGNORE_DATA block**)
(See Guide Introduction_7 "Extension to $DATA IGNORE=st filtering")

VECTORS and ABBREVIATED FUNCTIONS

Reserved variable names VECTRA, VECTRB, VECTRC, etc. may be used for predefined vectors. elements. Elements can be defined in any order.

Vectors that are defined outside of $DES (or $AES) may not be used in the $DES (or $AES) code. I.e., a vector may not be an implicit basic PK parameter.

If FUNCA, FUNCB, FUNCC, etc. is defined without a subscript, it is understood to be a simple variable, and then it may appear as such as a right-hand quantity. If `FUNCx` appears for the first time as a right-hand quantity, it is understood to be the reserved name of an abbreviated function, and then it must include a single argument, e.g. `FUNCA(VECTRA))` or `FUNCA(THETA(1))`. Complete FORTRAN code for an abbreviated function is supplied by the user (See **abbreviated function**).

The argument of an abbreviated function may be any expression that can be used in an assignment statement (possibly involving explicit vector elements), or it can be simply a vector. In the latter case, any VECTR may be used with any FUNC, e.g. `X=FUNCA(VECTRB)`.

With NONMEM VI 2.0:

Function names may be FUNCA through FUNCC.
Reserved argument vector names may be VECTRA through VECTRC.
A given reserved function may have at most 9 elements in the argument vector, and may appear in abbreviated code at most 9 times.

With NONMEM 7.3:

Function names may be FUNCA through FUNCI, and constant NFUNCX in SIZES may be used to increase the number.
Reserved argument vector names may be VECTRA through VECTRC.

With NONMEM 7.4:

Function names may be FUNCA through FUNCZ.
Reserved argument vector names may be VECTRA through VECTRZ.
Extended reserved names are recognized. These are FUNCxy and FUNCxyz, where each of x, y, z stands for an alphabetic character A-Z, e.g., FUNCAB or FUNCABC. Similar extended reserved names for vectors are also recognized: e.g, VECTRAB or VECTRABC.
A reserved function may appear in abbreviated code more than 9 times.

The $ABBR FUNCTION and $ABBR VECTOR feature allows the user to declare the names of user-defined functions and the names and sizes of user-defined argument vectors. Restrictions such as the number of user functions, the number of elements in a vector, and the number of times a given function may appear in

abbreviated code are more flexible and may be modified by the user.
(See **$abbreviated**).
(See **abbreviated function**).

PSEUDO ASSIGNMENT STATEMENTS

Pseudo-assignment statements are optional and provide certain special instructions. If present, they must appear (in any order) at the beginning of a record, before the rest of the essential abbreviated code comprising that record. They may be enclosed in parentheses, but two or more pseudo-assignment statements enclosed by parentheses must be separated by a semicolon ";". The following pseudo-statement may be used in all records except $MIX.

`COMRES=-1` : All quantities defined in abbreviated code are to be stored locally; they are not to be stored in the (global) MODULE NMPRD4.
(Variables in MODULE NMPRD4 can be used for communication between various user-routines, and can be displayed in tables and scatterplots.)

See the entries for specific records for information about pseudo-assignment statements special to those records.

RESERVED VARIABLE NAMES

Reserved variable names include names of those variables that have special meaning for the user, as well as names of variables that are used internally by NM-TRAN. Entries for $PRED, $PK, $ERROR, $DES, $AES, $AESINITIAL, and $TOL describe the reserved variable names for these abbreviated codes.

A variable name that is reserved in an abbreviated code may not be used as the name of a variable defined in this code, or in any other abbreviated code unless the pseudo-assignment statement `COMRES=-1` is used (in some code). If a reserved variable name is inappropriately used, NM-TRAN will generate an error message.

The following names are reserved in all abbreviated codes for internal use by NM-TRAN:

Names of modules and of variables in the modules
(See **Variables_in_modules**).
`GETETA SIMETA SIMEPS`
`COMSAV NWIND ETEXT IERPRD IERPRDU MSEC MFIRST NETEXT`
`ETA1-ETA9,ETA10-ETA99`
`Annnnn Bnnnnn Cnnnnn Dnnnnn Ennnnn Fnnnnn Pnnnnn Qnnnnn`
`MCnnnn MEnnnn MGnnnn MTnnnn`
(where nnnnn is numeric 00000-99999, and nnnn is numeric 0000-9999)

SUBSCRIPTED VARIABLES

Subscripts may be used with user-defined variables that are declared to be arrays using the $ABBR DECLARE record, and also with certain reserved variables:

`THETA,THETAFR,SETHET,THSIMP,SETHETR,THSIMPR`
`OMEGA,OMEGAF,SEOMEG,OMSIMP`
`SIGMA,SIGMAF,SESIGM,SGSIMP`
`CNTID,IIDX`
`CDEN_`
`A, A_0` (when used in a WRITE or PRINT statement)

`ETA, EPS, ERR` (when used in a WRITE OR PRINT statement)

Where a subscript is permitted or necessary, it may be an *integer expression*. An integer expression is an integer constant, a declared integer variable that has appeared on the left (i.e., has been given a value), a constant defined in SIZES, an expression involving integer constants and integer variables and arithmetic operators +, -, *, /, **. It may also be a character string that is replaced by an integer expression using the $ABBR REPLACE feature. The number of subscripts must be equal to the number of dimensions. If there are two subscripts, either or both may be constants or expressions. If a subscripted variable may be used on the left side, then the subscript may also be an integer expression. E.g.,

```
$ABBR DECLARE INTEGER IND
 ...
$PK
 IND=1
 X=THETA(IND+1)
```

With reserved random variables such as
`ETA(n) EPS(n) ERR(n) A(n) A_0(n)`
integer expressions may be used only if they are listed in in WRITE or PRINT statements. Otherwise `n` must be an integer constant, or a character string that is replaced by an integer constant using the $ABBR REPLACE feature.

NONMEM RESERVED VARIABLES

There are many variables that are generally internal to NONMEM, and often are not needed by users except occasionally, which are not explicitly recognized by NMTRAN, and so cannot be used in abbreviated code, but must be used with verbatim code (" at beginning of line). A convenient means of accessing such variables, as well as letting NMTRAN allow you to use the variables in abbreviated code is to place the MODULE definitions in an include file that begins with the name NONMEM_RESERVED (case insensitive). Insert include statements for such files at the beginning of the block(s) of abbreviated code in which you want to use them. The user may use any of these variables without using verbatim code. NMTRAN "reads" the nonmem_reserved file, and remembers the variables declared in there as acceptable to use.

A list of useful variables and their meanings are listed in ..\guides\useful_variables.pdf. Be careful in their use, as you have the ability to change the values of these reserved variables, and this could crash the system if you change the wrong thing.

NONMEM_RESERVED_GENERAL in the ..\util directory has a few of the quite useful variables listed. It needs to be copied to the present run directory so that NMTRAN has access to it. Note that there may be user-defined variable names in the control stream that inadvertently have the same names as variables in NONMEM_RESERVED_GENERAL, which may cause compiler errors. If so, make a smaller version of NONMEM_RESERVED_GENERAL with a name also starting with "NONMEM_RESERVED" that contains only those reserved variables of interest.

As an example,

NONMEM_RESERVED_GENERAL contains

" USE NMBAYES_INT, ONLY: ITER_REPORT,BAYES_EXTRA_REQUEST,BAYES_EXTRA

These variables can be used as shown in example 8:

```
$PK
include nonmem_reserved_general
BAYES_EXTRA_REQUEST=1
 ....
IF(BAYES_EXTRA==1 .AND. ITER_REPORT>=0 .AND. TIME==0.0) THEN
WRITE(50,*) ITER_REPORT,ID,CL,V1,Q,V2
ENDIF
```

ITER_REPORT contains the present iteration number as reported to the console or NONMEM report file. BAYES_EXTRA and BAYES_EXTRA_REQUEST are described in example8.ctl.

RESERVED AND OTHER FUNCTIONS

The nonmem_reserved_general file contains function declarations, such as TFI and TFD, which are convenient functions to easily convert an integer to text ("text from integer" TFI) or double precision value to text ("text from double" TFD).

If you wish to define your own function for use in abbreviated code and have the information about its proper use of arguments be conveyed upon its execution, so the compiler may detect errors, then one method is to package the definition of the function in a USE module, such as is done in the example User-defined Reserved Function.

REFERENCES: Guide IV, section IV
REFERENCES: Guide VI, section III.K
REFERENCES: Guide Introduction_7

ABBREVIATED FUNCTION

MEANING: User-supplied function that may be used in abbreviated code.
CONTEXT: Fortran coded function

The following applies to all versions of NONMEM.

USAGE:

```
FUNCTION FUNCA(X,X1,X2)
DOUBLE PRECISION X,X1,X2,FUNCA
DIMENSION X(9),X1(9),X2(9,9)
```

The constant "9" must be used exactly as shown.

DISCUSSION:
The FORTRAN function FUNCA may be used in abbreviated code. Similarly, the functions FUNCB, FUNCC, etc. may be used; their constructions would be similar to that of FUNCA. In abbreviated code, the function is referenced with a single argument, either a (scalar) expression or a vector. Certain reserved names such as VECTRA, VECTRB, etc. may be used in the abbreviated code, but the name of the argument is not in the code defining the function. (See **abbreviated code**). In the code defining the function, the function has three arguments.

Input Argument:

`X` The value of the argument (which may be a vector).

Output Argument:

`FUNCA`
The value of the function is to be stored in `FUNCA`.

There are two other outputs.

`X1` `X1(n)` is the first-partial derivative of the function with respect to the nth element of the argument.
If the value of `X(n)` will not be a value of a random variable, `X1(n)` need not be set.
If the argument `X` is a (scalar) expression, only `X1(1)` is needed.

`X2` `X2(n,m)` is the second-partial derivative of the function with respect to the nth and mth elements of the argument.
If the value of either `X(n)` or `X(m)` will not be a value of a random variable, `X2(n,m)` need not be set. Nor need any value of `X2` be set if a test of MSEC=1 is false (See **Partial Derivative Indicators**).
If the argument is a (scalar) expression, only `X2(1,1)` may be needed. If the argument is a vector and if the value of `X2(n,m)` is needed, then the value of `X2(m,n)` is needed as well, even though these two values will be identical.

The Fortran code for the functions should be placed in one or more files. Suppose there is one such file and its name is funcfile. It should be listed on the $SUBROUTINES record. It may contain more than one FUNCTION. E.g.,

```
$SUBROUTINES ... OTHER=funcfile
```

With NONMEM VI 2.0, reserved function names are FUNCA through FUNCC. reserved names for vectors are VECTRA through VECTRC.

With NONMEM 7.3, reserved function names are FUNCA through FUNCI.

With NONMEM 7.4, reserved function names are FUNCA through FUNCZ. Reserved names for vectors are VECTRA through VECTRZ.

If the abbreviated code uses one of these reserved function names, but the user does not supply the code via $SUBR OTHER, the NONMEM executable will be created, but there will be an error message in the NONMEM report file such as:

```
FUNCA WAS CALLED, BUT NO CODE WAS SUPPLIED.
```

With NONMEM 7.4, extended reserved names are recognized. These are FUNCxy and FUNCxyz, where each of x, y, z stands for an alphabetic character A-Z, e.g., FUNCAB or FUNCABC. Similar extended reserved names for vectors are also recognized: e.g, VECTRAB or VECTRABC.
If the abbreviated code uses one of these extended reserved function names, but the user does not supply the code via $SUBR OTHER, the NONMEM executable cannot be created, and there will be an error message from the Fortran compiler such as:

```
Undefined Symbols
_funcaaa_
```

(See **Abbreviated function example**).

The following applies only to NONMEM 7.4 and higher.

The $ABBR FUNCTION allows the user to declare the name of the function, the name of the vector of input arguments (which is of no use to the function), and the length of the vector of input arguments, which is passed to the function as argument NDIM. Constant NVECX in SIZES gives the maximum value size of the vector of input arguments.

In the NONMEM control stream:

```
$ABBR FUNCTION function_name(input_vector_name,dimension,usage)
```

(See **$ABBREVIATED**).
(See VECTORS and ABBREVIATED FUNCTIONS)

The function should be coded as follows:

```
FUNCTION function_name(X,X1,X2,NDIM)
USE SIZES, ONLY: DPSIZE,ISIZE
INTEGER(KIND=ISIZE) :: NDIM
REAL(KIND=DPSIZE), INTENT(IN)     :: X
REAL(KIND=DPSIZE), INTENT(IN OUT) :: X1,X2
REAL(KIND=DPSIZE):: function_name
DIMENSION X(NDIM),X1(NDIM),X2(NDIM,NDIM)
```

Function_name must be exactly as specified via $ABBR FUNCTION. The value coded as "dimension" is passed to the function as NDIM. (The value coded as "usage" is optional and does not appear in code for the function.)

The value of the function is to be stored in `function_name`. Otherwise, the function is coded exactly as described above, with X1 and X2 as output arguments. The input_vector_name is never used in the Fortran code for the function.

REFERENCES: none

ABSORPTION LAG PARAMETER

MEANING: Absorption lag (ALAG) parameter for PREDPP
CONTEXT: Additional PK Parameters

USAGE:

```
$PK
 ALAG1= ....
```

DISCUSSION:

Absorption lag parameters are used with PREDPP. They are optional additional PK parameters. With NM-TRAN, they are symbolized in the $PK block by reserved variables `ALAG`n, where n is the compartment number to which the parameter applies.

There is one absorption lag time (parameter) associated with every possible dose compartment of the kinetic model (the output compartment is not a possible dose compartment) and the absorption lag time used for a given dose is that one associated with the compartment into which the dose is given (the dose compartment).

The event time t on a dose record refers to the recorded time the dose was administered. In the case of a regular infusion, t is the time the infusion was initiated. An absorption lag time is an increment of time L such that the time that the dose is regarded (by PREDPP) as entering (or starting to enter) the system is t+L.

Absorption lag times are optional in the sense that absorption lag times associated with compartments never used as dose compartments may be ignored. The values of absorption lag times that are not computed in PK are always understood to be 0.

An absorption lag time for a dose is computed by the PK routine using, if needed, information in the dose record. When additional doses are specified on a dose event record, the absorption lag time applies to the dose and to all the additional doses. With a steady-state multiple dose the absorption lag time applies not only to this dose, but also to all the preceding implied doses. With a steady-state dose, the lag time should be less than the interdose interval.

(See **cmt**, **pk**, **$pk**, **default_compartment**).

REFERENCES: Guide IV, section V.C.5
REFERENCES: Guide VI, section III.F.6 , V.F.1 , V.H

ACTIVE ETA LIST FOR PREDPP

MEANING: PREDPP read-only global variables
CONTEXT: PK, ERROR, TRANS routines

USAGE:

```
USE PROCM_INT, ONLY: NACTIV,M=>IDXETA
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: LVR
INTEGER(KIND=ISIZE) :: NACTIV,IDXETA(LVR)
```

DISCUSSION:

`NACTIV`

`NACTIV` = # of etas in the problem - `NRETA`.
(See **Non-Active_ETA_list_for_PRED**).

`NACTIV` tells how many of the etas in the problem have partial derivatives that NONMEM is not currently ignoring.

`M` `M(k)` is the index of the kth 0-valued element of LVOUT (for k=1,...,NACTIV).

Suppose a user-written TRANS routine modifies the kth. element of GG. Here is code that might be used to modify only those of its first eta derivatives that are of current interest to NONMEM:

```
      DO 100 I=1,NACTIV
  100 GG(K,M(I)+1,1)=GG(K,M(I)+1,1)  + ...
```

Location prior to NONMEM 7: procm5

REFERENCES: None.

ADDITIONAL OUTPUT FILES

MEANING: NONMEM Additional Output files
CONTEXT: NONMEM Output Files

DISCUSSION:

With NONMEM 7, a NONMEM run produces several additional output files. These provide a more efficient way of extracting numerical results from the analysis than by obtaining them from the NONMEM report file.
(See **raw_output_file**).

Names of the files start with "root", where root is the root name (not including extension) of the NM-TRAN control stream file given at the nmfe command line, or root="nmbayes" if the control stream file name is not passed as an argument to the NONMEM executable. In this example the root is myexample, and NONMEM 7.4 is invoked.

```
nmfe74 myexample.ctl myexample.res
```

The files have names with extensions (suffixes) .ext, .cov, .clt, .coi, .cor, .cpu, .phi, .phm, .grd, .shk, .shm, .cnv, smt, .rmt, .vpd, .ets, .bfm. In this example, they will be myexample.cov, myexample.coi, etc.

Each type of file accumulates information from each step that generates the type of output in question.

The first two lines of each file is the same. First is a header line that begins with the word TABLE, such as:

```
TABLE NO.      n:
```

These tables are not related to the ones produced by the $TABLE record. The value of n is incremented each time the Estimation Step is implemented, i.e., once per $ESTIM record, when MAXEVAL is not 0 and $EST is not omitted. The $EST records may be in the same problem, or in subsequent problems. If $COV is present in a problem without $EST (e.g., when $MSFI is used), then n is also incremented.

Next on the header line is the analysis text (same as given on the #METH: line in the report file), e.g., "First Order".

Also on this line are:

```
Problem=1 Subproblem=0 Superproblem1=0 Iteration1=0
          Superproblem2=0 Iteration2=0
```

The values of Problem, Subproblem, Superproblem1, Iteration1, Superproblem2, Iteration2 are the same values as would be found in the following variables in modules during the run:
IPROB IREP S1NUM S1IT S2NUM S2IT

The second line contains the column headers for the current additional file, all on one line.

Options NOTITLE and NOLABEL and FORMAT of the $ESTIMATION record can be used to supress the first and/or second lines, and to modify the format of the values in the table. They apply to raw and additional output files, but only for the current Estimation Step. The options must be re-specified with each $EST record.

The additional output files are:

root.ext
: Always. (See **raw_output_file**).

root.phi
: Always. Individual phi parameters (phi(i)=mu(i)+eta(i), for ith parameter), and their variances phc(,). For parameters not MU referenced phi(i)=eta(i). When a classical method is performed (FOCE, Laplace), then mode of posterior eta(i) are printed out, along with their Fisher information (first order expected value for FOCE, second order for Laplace) assessed variances etc(,). For ITS, these parameters are the modes of the posterior density, with first-order approximated expected variances (or second order variances if $EST METHOD=ITS LAPLCE is used). For IMP, IMPMAP, SAEM methods, they are the Monte Carlo evaluated conditional mean parameters and variances of the parameters under the posterior density. For MCMC Bayesian, they are random single samples of phi(), as of the last position. Their variances are zero. Individual objective function values (obji) are also produced. As of NONMEM 7.4, if $EST PHITYPE=1 is specified, then conditional mean etas are reported in the phi table. regardless of the analysis method.

root.iph (NM75)
: As of nm75, samples of phi/eta are collected at each BAYES iteration, and summarized to provide conditional mean phi and phc() in the root.phi table, as described above. By default, the individual phi values from each iteration are not stored. However, if you set
$EST ... BAYES_PHI_STORE=1
then phi and eta values from each BAYES iteration will be stored in root.iph. For non-mixture problems, only records of SUBP=0 are recorded, as there are no sub-population divisions. For mixture problems, the SUBP=0 records contain the composite phis and etas (the average of these across all non-negative iterations are in the root.phi table), and the SUBP>0 records contain the phis and etas appropriate to each sub-population SUBP (the average of these across all non- negative iterations are in the root.phm table). The root.iph file can become quite large, so it should be used only on the final analysis.

root.phm (NM72)
: Individual phi/eta/obji parameters per sub-population. This file is only produced in $MIXTURE problems. As of nm75, for MCMC Baysian analysis, the items listed in this table consist of the average values among all the iterations during the stationary distribution phase. Before nm75, the values consisted of values from the last MCMC iteration.

root.shk (NM72)
: This file presents composite eta shrinkage and epsilon shrinkage information, the same as given in the NONMEM report file between the #TERM: and #TERE: tags, but in rows/column format, and with adjustable formatting.
(See **shrinkage**).

root.shm (NM73)
: This file is a shrinkage map describing which etas were included or excluded in the eta shrinkage assessment. (See **shrinkage**).

root.grd (NM72)
: Gradient values for classical NONMEM methods. Always.

root.xml (NM72)
: An XML markup version of the contents of the NONMEM report file. The rules by

which it is constructed are given in output.xsd and output.dtd.

root.cov
: Full variance-covariance error matrix of thetas, sigmas, and omegas. Same as "COVARIANCE MATRIX OF ESTIMATE" in the NONMEM report except that it is full (NONMEM displays only the lower triangle), elements that are fixed or not estimated are displayed as 0 (NONMEM displays them as .....), and, when the data are individual, sigma is displayed as 0 (NONMEM does not display sigma at all.) Only generated if the $COVARIANCE record is present.
(See **covariance matrix of estimate**).
(See **Covariance_Output_files**).
(See **order_option**).

root.clt (NM74)
: The lower-triangular portion of the variance-covariance of the parameter estimates reported in root.cov. This is provided for easier pasting of the information as theta priors for a subsequent analysis. Only generated if the $COVARIANCE record is present.

root.cor
: Full correlation matrix of thetas, sigmas, and omegas. Same as "CORRELATION MATRIX OF ESTIMATE", except as described for root.cov.
(See **correlation matrix of estimate**).

root.coi
: Full inverse covariance matrix (Fischer information matrix) for thetas, sigmas, and omegas. Same as "INVERSE COVARIANCE MATRIX OF ESTIMATE" except as described for root.cov.

root.cnv (NM72)
: Convergence information for the Monte Carlo/EM methods, if CTYPE>0.
(See **raw_output_file**).

root.smt (NM72)
: Contains the S matrix, if $COV step failed.

root.rmt (NM72)
: Contains the R matrix, if $COV step failed.

root.imp (NM73)
: Produced if the user selects importance sampling with option IACCEPT=0.0. Contains the final IACCEPT and DF values that NONMEM selected for each subject.

root.npd (NM73)
: For NONPARAMETRIC method. Each row contains information about a support point.

root.npe (NM73)
: For NONPARAMETRIC method. The expected value etas and expected value eta covariances (ETC) are listed for each problem or sub-problem.

root.npi (NM73)
: For NONPARAMETRIC method. The individual probabilities are listed. Each row contains information about a support point.

root.npl (NM74)
: For NONPARAMETRIC method. The individual data likelihoods (not including the parameter density) are listed in this file.

root.fgh (NM73)

This file is produced if the user selects $EST NUMDER=1 or NUMDER=3. The file lists the numerically evaluated derivatives of Y with respect to eta, where

F=Y (the prediction)
G(I,1)=partial Y with respect to eta(i))
G(I,J+1)=Second derivatives of Y with respect to eta(i),eta(j)
H(I,1)=partial Y with respect to eps(i)
H(i,j+1)=partial Y with respect to eps(i),eta(j)

The values are those used in the Estimation Step. Values of F are computed by PRED with eta=0 if METHOD=0 is used; otherwise they are computed with non-zero etas: the Conditional (CPE) etas, empirical bayes estimates (EBE), mode a posteriori (MAP) estimates) or conditional mean etas, depending on the Estimation Method. Values of G are computed by NONMEM via finite differences using these values of F and eta. Values of H are computed by NONMEM with the same values of F and eta if INTERACTION is specified; otherwise, they are computed with eta=0.

root.agh (NM73)

This file is produced if the user selects $EST NUMDER=2 or NUMDER=3. The file lists the analytically evaluated derivatives of Y with respect to eta. The format and order is the same as file root.fgh. The values are those returned by subroutine PRED in arguments F, G, and H. These are 0 if not computed by PRED. (E.g. second derivatves are not computed analytically with a method other than LAPLACIAN.) Values in root.agh correspond exactly to those in root.fgh. That is, values of F and G are from calls to PRED with eta=0 or with non-zero etas, as described above. Values of H are from calls to PRED with the same values of eta if INTERACTION is specified; otherwise, they are from calls to PRED with eta=0, regardless of the Estimation method.

root.vpd (NM74)

This file is produced if the $TABLE file option VARCALC=1 or VARCALC=2 is used. It is the full variances- covariances among all user-defined and PREDPP parameters. The FORMAT used for this file is that defined in the $EST record.

root.ets (NM74)

As of NM74, one can obtain random samples of individual etas, and uses these for covariate and model diagnostics. $EST option ETASAMPLES=1 causes individual ISAMPLE random eta samples per subject, to be written to root.ets, where root is the root name of the control stream file.

See "Stochastic Approximation Expectation Maximization (SAEM) Method" in Guide Introduction_7.

root.bfm (NM75)

When OFVTYPE=8 during optimal design ($DESIGN OFVTYPE=8) the progress of average individual conditional variances (average empirical Bayes conditional variance) are shown in the root.bfm file (where root is name of control stream file), and the final one on the -1000000000 line, during an optimization. The RAW file (by default named root.ext) which shows only the starting, and final values of the standard errors of population parameters, as extra information.

root.cpu (NM73)

The cpu time in seconds is reported in this file. It is an accurate representation of the computer usage, whether single or parallel process. The same problem when run

singly or in parallel will report a similar cpu time. This is in contrast with elapsed time, which is improved with parallelization.

With NONMEM 7.3 and higher, the $THETAI and $THETAR records may be used to transform initial and final estimates of THETA, respectively. This will affect values in the .ext, .cov, etc. files
(See **$thetai**, **$thetar**)

REFERENCES: Guide Introduction_7

ADDITIONAL RECORD COUNTERS

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

USAGE:

```
include nonmem_reserved_general
```

DISCUSSION:

In addition to the record counters NDREC and NIREC, NONMEM 7.4 provides additional record counters. All except IRECIDX refers to a record number within the current individual record.

`FIRSTREC`
: First record of the individual.

`LASTREC`
: Last record of the individual.

`FIRSTOBS`
: First observation record of the individual (for which MDV=0 or 100)

`LASTOBS`
: Last observation record of the individual (for which MDV=0 or 100)

`FIRSTDOS`
: With PREDPP, first dose record of the individual (for which EVID=1 or 4, and MDV=1 or MDV=101).
Without PREDPP, -1.

`LASTDOS`
: With PREDPP, last dose record of the individual (for which EVID-1 or 4, and MDV=1 or MDV=101).
Without PREDPP, -1.

`EFIRSTREC`
: First record of the individual during Estimation or Covariance Steps (for which MDV=0 or MDV=1)

`ELASTREC`
: Last record of the individual during Estimation or Covariance Steps (for which MDV=0 or MDV=1)

`EFIRSTOBS`
: First observation record of the individual during Estimation or Covariance Steps (for which MDV=0)

`ELASTOBS`
: Last observation record of the individual during Estimation or Covariance Steps (for which MDV=0)

`EFIRSTDOS`
: With PREDPP, first dose record of the individual during Estimation or Covariance Steps (for which EVID=1 or 4, and MDV=1).
Without PREDPP, -1.

`ELASTDOS`
: With PREDPP, last dose record of the individual during Estimation or Covariance Steps (for which EVID=1 or 4, and MDV=1).

Without PREDPP, -1.

IRECIDX

IRECIDX+1 is the starting record number in the NONMEM data set for the current individual record. NDREC starts at 1 for the first record of each individual record, so that NDREC+IRECIDX is the number of the current data record within the NONMEM data set.

Like NIREC and NDREC, the additional record counters also change value in conjunction with calls to PASS, a NONMEM utility routine. They may all be used as right-hand quantities in $PRED, $PK, and $ERROR blocks, and in a $INFN block in conjunction with PASS.

REFERENCES: Guide Introduction_7

ADDL DATA ITEM

MEANING: Additional Dose (ADDL) data item for PREDPP
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... ADDL ...`

DISCUSSION:
`ADDL` labels PREDPP's additional dose (ADDL) data item. The additional dose data item is optional. In a dose or reset-dose event record it describes how many additional doses, exactly like the present ("initiating") dose, should be given. The interdose interval (II) data item gives the time between the additional doses (See **ii**).

When ADDL is 0, no additional doses are given.

For non-steady-state doses, ADDL should be a positive number if and only if the II data item is a positive number. II gives the time between additional doses.

For steady-state infusions, ADDL must be 0.

For other steady-state doses, ADDL is optional. If it is a positive number, it continues the pattern of implied doses beyond the steady-state dose. The additional doses of the pattern are non-steady state doses.

With NM75 there is a new way of computing SS, the Empirical method, in which there is no SS data item, and a negative value of ADDL requests the computation. This is described separately.
(See **empirical_SS**).
(See Guide Introduction_7 "An Empirical Method of Achieving Steady State")

For observation, other-type and reset event records, ADDL should be zero.

The following two data record fragments specify the same set of doses:

TIME	AMT	ADDL	II	DV
10.	15.	0	0	.
15.	.	.	.	25.7
20.	10.	0	0	.
22.	15.	0	0	.
34.	15.	0	0	.
36.	.	.	.	30.2

TIME	AMT	ADDL	II	DV
10.	15.	2	12.	.
15.	.	.	.	25.7
20.	10.	0	0	.
36.	.	.	.	30.2

The observation at time 15 and the extra dose at time 20 do not affect the regular dosing regimen (doses at times 10, 22, and 34).

There is a discussion of user (concomitant) data items used to compute PK parameters that are relevant to ADDL doses; (See **$BIND**).

REFERENCES: Guide VI, section V.K

ADVAN (GENERAL)

MEANING: Choice of Pharmacokinetic Model for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [ADVAN=]ADVANn
```

SAMPLE:
```
$SUBROUTINE   ADVAN1
```

DISCUSSION:
PREDPP contains a library of routines, called ADVAN routines, which implement specific kinetic models. Exactly one ADVAN routine must be selected for each NONMEM/PREDPP run. Its function is to "advance" the kinetic system from one state time point to the next. The ADVAN routines are:

ADVAN1	One Compartment Linear Model
ADVAN2	One Compartment Linear Model with First Order Absorption
ADVAN3	Two Compartment Linear Model
ADVAN4	Two Compartment Linear Model with First Order Absorption
ADVAN5	General Linear Model
ADVAN6	General Nonlinear Model
ADVAN7	General Linear Model with Real Eigenvalues
ADVAN8	General Nonlinear Model with Stiff Differential Equations
ADVAN9	General Nonlinear Model with Equilibrium Compartments
ADVAN10	One Compartment Model with Michaelis-Menten Elimination
ADVAN11	Three Compartment Linear Model
ADVAN12	Three Compartment Linear Model with First Order Absorption
ADVAN13	General Nonlinear Model using LSODA
ADVAN14	General Nonlinear Model using CVODA
ADVAN15	General Nonlinear Model with Equilibrium Compartments using IDA
ADVAN16	Delay Differential Equation Model using RADAR5
ADVAN17	Delay Differential Equation Model with Equilibrium Compartments using RADAR5
ADVAN18	Delay Differential Equation Model using DDE_SOLVER

With a general ADVAN (numbered 5 and above, except for 10, 11, 12), a $MODEL record (or user-supplied MODEL subroutine) must be provided to specify certain details of the model. With ADVAN 6, 8, 9, 13, 14, 15, 16, 17, 18, a $DES block (or DES subroutine) must be provided to evaluate the differential equations. With ADVAN9 and ADVAN15 and ADVAN17, $AES and $AESINITIAL blocks (or AES subroutine) may also be provided to evaluate the algebraic expressions. With any ADVAN for which $DES is required, and with ADVAN10 or SS6, TOL must be specified by means of the TOL option of the $SUBROUTINES record (or a $TOL block or TOL subroutine).

To use a given ADVAN routine, follow two steps:

1. Specify this ADVAN to NM-TRAN on the $SUBROUTINES record. This enables NM-TRAN to interpret other records such as $PK.
2. Include the specified ADVAN routine when building the NONMEM executable. NM-TRAN itself places the name of the specified ADVAN routine (and any other routines from the PREDPP library required by that ADVAN) in its output file FREPORT.

The nmfe and nmfe.bat commands automatically include the correct ADVAN routine in the NONMEM executable.

REFERENCES: Guide VI, section I , III.A , VII.C
REFERENCES: Guide IV, section V.C.3 , V.C.5

ADVAN1

MEANING: Choice of Pharmacokinetic Model for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [ADVAN=]ADVAN1
```

SAMPLE:
```
$SUBROUTINE ADVAN1
```

DISCUSSION:
ADVAN1 is a routine in PREDPP's library which implements the kinetic equations for the One Compartment Linear Model.

Compt. No.	Function	Initial Status	On/Off Allowed	Dose Allowed	Default for Dose	Default for Obs.
1	Central	On	No	Yes	Yes	Yes
2	Output	Off	Yes	No	No	No

TRANS routines that may be used: TRANS1, TRANS2

Basic PK parameter with TRANS1:

`K` (rate constant of elimination)

Basic PK parameters with TRANS2: (See **TRANS2**).

Additional PK parameters:

`S1` - Scale for central compartment (also called `SC`)
`S2` - Scale for output compartment (also called `S0`)
`F1` - Bioavailability for central compartment
`R1` - Rate for central compartment
`D1` - Duration for central compartment
`ALAG1` - Absorption lag for central compartment
`F0` - Output fraction (also called `F2`, `FO`)
`XSCALE` - Time scale
`MTIME(i)` - Model event times

REFERENCES: Guide VI, section VII.C.1

ADVAN2

MEANING: Choice of Pharmacokinetic Model for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [ADVAN=]ADVAN2
```

SAMPLE:
```
$SUBROUTINE ADVAN2
```

DISCUSSION:
ADVAN2 is a routine in PREDPP's library which implements the kinetic equations for the One Compartment Linear Model with First Order Absorption.

Compt. No.	Function	Initial Status	On/Off Allowed	Dose Allowed	Default for Dose	Default for Obs.
1	Depot	Off	Yes	Yes	Yes	No
2	Central	On	No	Yes	No	Yes
3	Output	Off	Yes	No	No	No

TRANS routines that may be used: TRANS1, TRANS2

Basic PK parameters with TRANS1:

`K` (rate constant of elimination)
`KA` (rate constant of absorption)

Basic PK parameters with TRANS2: (See **TRANS2**).

Additional PK parameters:

`S1` - Scale for depot compartment
`S2` - Scale for central compartment (also called `SC`)
`S3` - Scale for output compartment (also called `S0`)
`F1` - Bioavailability for depot compartment
`F2` - Bioavailability for central compartment
`R1` - Rate for depot compartment
`R2` - Rate for central compartment
`D1` - Duration for depot compartment
`D2` - Duration for central compartment
`ALAG1` - Absorption lag for depot compartment
`ALAG2` - Absorption lag for central compartment
`F0` - Output fraction (also called `F3`, `FO`)
`XSCALE` - Time scale
`MTIME(i)` - Model event times

REFERENCES: Guide VI, section VII.C.2

ADVAN3

MEANING: Choice of Pharmacokinetic Model for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [ADVAN=]ADVAN3
```

SAMPLE:
```
$SUBROUTINE ADVAN3
```

DISCUSSION:
ADVAN3 is a routine in PREDPP's library which implements the kinetic equations for the Two Compartment Linear Model.

Compt. No.	Function	Initial Status	On/Off Allowed	Dose Allowed	Default for Dose	Default for Obs.
1	Central	On	No	Yes	Yes	Yes
2	Peripheral	On	No	Yes	No	No
3	Output	Off	Yes	No	No	No

TRANS routines that may be used: TRANS1, TRANS3, TRANS4, TRANS5, TRANS6
Basic PK parameters with TRANS1:

- `K` (rate constant of elimination)
- `K12` (rate constant from central to peripheral)
- `K21` (rate constant from peripheral to central)

Basic PK parameters with TRANS3: (See **TRANS3**).

Basic PK parameters with TRANS4: (See **TRANS4**).

Basic PK parameters with TRANS5: (See **TRANS5**).

Basic PK parameters with TRANS6: (See **TRANS6**).

Additional PK parameters:

- `S1` - Scale for central compartment (also called `SC`)
- `S2` - Scale for peripheral compartment
- `S3` - Scale for output compartment (also called `S0`)
- `F1` - Bioavailability for central compartment
- `F2` - Bioavailability for peripheral compartment
- `R1` - Rate for central compartment
- `R2` - Rate for peripheral compartment
- `D1` - Duration for central compartment
- `D2` - Duration for peripheral compartment
- `ALAG1` - Absorption lag for central compartment
- `ALAG2` - Absorption lag for peripheral compartment
- `F0` - Output fraction (also called `F3`, `FO`)
- `XSCALE` - Time scale
- `MTIME(i)` - Model event times

REFERENCES: Guide VI, section VII.C.3

ADVAN4

MEANING: Choice of Pharmacokinetic Model for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [ADVAN=]ADVAN4
```

SAMPLE:
```
$SUBROUTINE ADVAN4
```

DISCUSSION:
ADVAN4 is a routine in PREDPP's library which implements the kinetic equations for the Two Compartment Linear Model with First Order Absorption.

Compt. No.	Function	Initial Status	On/Off Allowed	Dose Allowed	Default for Dose	Default for Obs.
1	Depot	Off	Yes	Yes	Yes	No
2	Central	On	No	Yes	No	Yes
3	Peripheral	On	No	Yes	No	No
4	Output	Off	Yes	No	No	No

TRANS routines that may be used: TRANS1, TRANS3, TRANS4, TRANS5, TRANS6

Basic PK parameters with TRANS1:

- `K` (rate constant of elimination)
- `K23` (rate constant from central to peripheral)
- `K32` (rate constant from peripheral to central)
- `KA` (rate constant of absorption)

Basic PK parameters with TRANS3: (See **TRANS3**).

Basic PK parameters with TRANS4: (See **TRANS4**).

Basic PK parameters with TRANS5: (See **TRANS5**).

Basic PK parameters with TRANS6: (See **TRANS6**).

Additional PK parameters:

- `S1` - Scale for depot compartment
- `S2` - Scale for central compartment (also called `SC`)
- `S3` - Scale for peripheral compartment
- `S4` - Scale for output compartment (also called `S0`)
- `F1` - Bioavailability for depot compartment
- `F2` - Bioavailability for central compartment
- `F3` - Bioavailability for peripheral compartment
- `R1` - Rate for depot compartment
- `R2` - Rate for central compartment
- `R3` - Rate for peripheral compartment
- `D1` - Duration for depot compartment
- `D2` - Duration for central compartment
- `D3` - Duration for peripheral compartment
- `ALAG1` - Absorption lag for depot compartment
- `ALAG2` - Absorption lag for central compartment
- `ALAG3` - Absorption lag for peripheral compartment
- `F0` - Output fraction (also called `F4`, `FO`)

`XSCALE` - Time scale
`MTIME(i)` - Model event times

REFERENCES: Guide VI, section VII.C.4

ADVAN5 ADVAN7

MEANING: Choice of Pharmacokinetic Model for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [ADVAN=]ADVAN5
```

SAMPLE:
```
$SUBROUTINE ADVAN5
```

DISCUSSION:
ADVAN5 and ADVAN7 are routines in PREDPP's library which implement the general linear model. The general linear model is used for systems in which a drug is distributed between compartments according to linear processes. ADVAN7 may be used when the eigenvalues of the rate constant matrix are known to be real (which is true for many pharmacokinetic systems such as mammillary models). It is generally faster than ADVAN5.

A $MODEL record is required to describe the compartments and their attributes. The $PK record (or, if a user-supplied PK routine is used, the $MODEL record) describes how the compartments are linked.

TRANS routines that may be used: TRANS1

Suppose there are m compartments in the system, including the output compartment.

Basic PK parameters with TRANS1:

`Kij` (rate constant from compartment i to compartment j)
`Ki0` (alternate name for `Kim`)

The letter T may be used as a separator between the two compartment numbers, e.g., KiTj. The letter T is optional when there is no ambiguity, but required when there are two possible interpretations of the numbers that follow K. E.g., with 12 compartments, `K111` is ambiguous. It should be coded `K1T11` or `K11T1`, depending if it symbolizes the rate constant from compartment 1 to compartment 11 or from compartment 11 to compartment 1.

Additional PK parameters:

For each compartment n in the system (n=1, ..., m):

`Sn` - Scale for nth compartment
`S0` - Alternate name for scale for output compartment

For each dosable compartment n in the system:

`Fn` - Bioavailability for nth compartment
`Rn` - Rate for nth compartment
`Dn` - Duration for nth compartment
`ALAGn` - Absorption lag for nth compartment

Other additional PK parameters:

`F0` - Output fraction (also called `Fm`, `FO`)
`XSCALE` - Time scale
`MTIME(i)` - Model event times

Required PREDPP Library subroutines:
ADVAN5 - REXPON, RXSUBS
ADVAN7 - RRXPON, RXSUBS

Notes:

1 Analytical second derivatives used with the Laplacian method are not obtained with ADVAN5 and ADVAN7. Numerical second derivatives must be used.

REFERENCES: Guide VI, section VII.C.5 , VII.C.7
REFERENCES: Guide IV, section V.C.4

ADVAN6 ADVAN8 ADVAN13 ADVAN14 ADVAN16 ADVAN18

MEANING: Choice of Pharmacokinetic Model for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [ADVAN=]ADVAN6
```

SAMPLE:
```
$SUBROUTINE ADVAN6
```

DISCUSSION:
ADVAN6, ADVAN8 ADVAN13, ADVAN14, ADVAN16, ADVAN18 are routines in PREDPP's library which implement the general non-linear model. The general non-linear model is used for systems in which a drug is distributed between compartments according to first-order processes. The differences are in the package of code that performs the integration of the differential equations.

ADVAN6 uses IMSL's DVERK, a Runge-Kutta-Verner fifth and sixth order method of integration, for nonstiff problems.

ADVAN8 uses IMSL's DGEAR, the Gear method of numerical integration, for stiff problems.

ADVAN13 uses the LLNL solver for ordinary differential equations (LSODA), with automatic method switching for stiff (BDF) and non-stiff (Adams method) problems. It is a decendent of ADVAN8.

ADVAN14 uses CVODES from the LLNL SUNDIALS system for ordinary differential equations and is a descendent of LSODA (ADVAN13). ADVAN14 also has a root-finding algorithm.
The user may choose to make modifications to ..\pr\CVODEU.f90. See ..\guides\cvs_guide.pdf and ..\guides\cv_guide.pdf.

ADVAN16 uses the RADAR5 delay differential equation solver.
See **INTRODUCTION TO NONMEM 7, Using the Delay differential equation Solvers with the ddexpand program.**
See also ..\guides\manrad5-v2.pdf.
See also ..\examples\dde\advan16_dde.pdf

ADVAN18 uses the DDE_SOLVER delay differential equation solver.
See **INTRODUCTION TO NONMEM 7, Using the Delay differential equation Solvers with the ddexpand program.**
See also ..\guides\ddes_f90.pdf.
See also ..\examples\dde\advan16_dde.pdf

ADVAN8 may be used in preference to ADVAN6 when the differential equations describing the processes are stiff. ADVAN13 may be preferable with a mixed system (both stiff and non-stiff). For discussion of ADVAN14 as well as changes to ADVAN13, see Guide INTRODUCTION TO NONMEM 7.

A $MODEL record is required to describe the compartments and their attributes. The $DES record is required to describe the differential equations.

TRANS routines that may be used: TRANS1

Explicit Basic PK parameters:

`P(n)` (nth basic PK parameter)

Implicit basic PK parameters:

PK-defined variables used also in $DES block

Additional PK parameters:

Suppose there are m compartments in the system, including the output compartment. For each compartment n in the system (n=1, ..., m):

`Sn` - Scale for nth compartment
`S0` - Alternate name for scale for output compartment

For each dosable compartment n in the system:

`Fn` - Bioavailability for nth compartment
`Rn` - Rate for nth compartment
`Dn` - Duration for nth compartment
`ALAGn` - Absorption lag for nth compartment

Other additional PK parameters:

`F0` - Output fraction (also called `Fm`, `FO`)
`XSCALE` - Time scale
`MTIME(i)` - Model event times

TOLERANCE values

TOL (Relative Tolerance) is required; this can be supplied with the $SUBROUTINE or $TOL record, or a user-supplied TOL routine, which sets NRD=TOL.
(See **$SUBROUTINES**).
(See **$TOL**).

With NONMEM 7.4, ATOL (Absolute tolerance) may be specified for ADVAN9, ADVAN13, ADVAN14, ADVAN16, ADVAN18.
For example, $SUBROUTINE ... ATOL=n.
The default is 12 (that is, accuracy is 10**(-12)). Usually the problem runs quickly when using this setting. On occasion, however, you may want to reduce ATOL (usually set it equal to that of TOL), and improve speeds of up to 3 to 4 fold. Values of TOL and ATOL may also be specified for evaluations of Steady State amounts, and also for the Covariance Step.

See options ATOL, TOLC, ATOLC, SSATOL, SSATOLC of the $SUBROUTINE and $TOL records.

A user-supplied TOL routine may assign values of NRD and ANRD specifically for the initial (base) setting and each NONMEM step (estimation, covariance, simulation, table/scatter step, simulation, initial parameters estimate, nonparametric). It may also supply specific values for each compartment, and for Steady State amounts.
(See **TOL**).

With ADVAN13, the user may find that the TOL option should specify larger NRD values than for other ADVANs (e.g., ADVAN6). Values of 7 or 8 may not be unreasonable with double precision.

See **INTRODUCTION TO NONMEM 7, Controlling the Accuracy of the Gradient Evaluation and Individual Objective Function Evaluation**

Reserved variable MXSTEP (the maximum number of integration steps) can be specified for ADVAN13 and ADVAN14 and ADVAN16.

(See **mxstep**).

With ADVAN13, if there is a period of time during which some compartment's amount should be zero, that compartment should be turned off. Otherwise, very small amounts can appear in the compartment, which can cause difficulties during subsequent time periods.

By default, the initial conditions (i.e., compartment amounts) are zero at the start of each individual record. Different initial conditions may be specified using a compartment initialization block in PK. When endogenous drug is specified in the differential equations, non-zero initial conditions may also be computed using a steady-state dose event record with SS>0 and AMT=0 and RATE=0, or using the I_SS (Initial Steady State) feature of MODEL and/or PK.
(See **Compartment_Initialization_Block**, **SS_dose**).
(See **$model**, **$pk**).
(See **model**, **pk**).
(See **Initial_Steady_State:_I_SS,ISSMOD**).

Required PREDPP Library subroutines:
ADVAN6 - DVERK1, FCN1
ADVAN8 - DGEAR1, FCN1, FCN3
ADVAN13 - LSODA, FCN1, JAC2
ADVAN14 - CVODE, FCN1, JAC2
ADVAN16 - RADAR5NM, FCN1, JAC2
ADVAN18 - ADDELSV, FCN1, JAC2

REFERENCES: Guide VI, section VII.C.6 , VII.C.8
REFERENCES: Guide IV, section V.C.4 , V.C.5

ADVAN9 15

MEANING: Choice of Pharmacokinetic Model for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:

```
$SUBROUTINES [ADVAN=]ADVAN9
$SUBROUTINES [ADVAN=]ADVAN15
$SUBROUTINES [ADVAN=]ADVAN17
```

SAMPLE:

```
$SUBROUTINE ADVAN9
$SUBROUTINE ADVAN15
$SUBROUTINE ADVAN17
```

DISCUSSION:
ADVAN9 and ADVAN15 and ADVAN17 are routines in PREDPP's library which implement the general non-linear model with equilibrium compartments. This general non-linear model is used for systems in which a drug is distributed between compartments according to a system of first-order differential-algebraic processes. It may be useful when the system consists of only stiff differential equations, and there are no equilibrium compartments (i.e., no algebraic equations). It may also be used when the system consists of only algebraic equations (i.e., no differential equations).

A $MODEL record is required to describe the compartments and their attributes. The $DES record describes the differential equations, if any. The $AES and $AESINITIAL records describe the algebraic expressions, if any.

TOL is required; (See **$SUBROUTINES**).

TRANS routines that may be used: TRANS1

Explicit Basic PK parameters:

`P(n)` (nth basic PK parameter)

Implicit basic PK parameters:

PK-defined variables used also in $AES or $DES blocks

Additional PK parameters:

Suppose there are m compartments in the system, including the output compartment. For each compartment n in the system (n=1, ..., m):

`Sn` - Scale for nth compartment
`S0` - Alternate name for scale for output compartment

For each dosable compartment n in the system:

`Fn` - Bioavailability for nth compartment
`Rn` - Rate for nth compartment
`Dn` - Duration for nth compartment
`ALAGn` - Absorption lag for nth compartment

Other additional PK parameters:

`F0` - Output fraction (also called `Fm`, `FO`)
`XSCALE` - Time scale
`MTIME(i)` - Model event times

With NONMEM 7.4, ATOL may be specified for ADVAN9 and ADVAN15. (ADVAN17 is new with NONMEM 7.5. The following applies to ADVAN17 as well.) TOL (relative tolerance) and ATOL (Absolute tolerance) may be specified on the $SUBROUTINE record, the $TOL record, or by a user-supplied TOL subroutine. See also the ATOL option of the $ESTIMATION and $COVARIANCE records.

ATOL is not required. The default is 12 (that is, accuracy is 10**(-12)). Usually the problem runs quickly when using this setting. On occasion, however, you may want to reduce ATOL (usually set it equal to that of TOL), and improve speeds of up to 3 to 4 fold.

With ADVAN9, the user may find that the TOL option should specify larger NRD values than for other ADVANs (e.g., ADVAN6). Values of 7 or 8 may not be unreasonable.

Reserved variable MXSTEP (the maximum number of integration steps) can be specified for ADVAN9 and ADVAN15 and ADVAN17, as well as for ADVAN13 and ADVAN14 and ADVAN16.
(See **mxstep**).

With ADVAN9, if there is a period of time during which some compartment's amount should be zero, that compartment should be turned off. Otherwise, very small amounts can appear in the compartment, which can cause difficulties during subsequent time periods.

By default, the initial conditions (i.e., compartment amounts) are zero at the start of each individual record. Different initial conditions may be specified using a compartment initialization block in PK. When endogenous drug is provided in the differential equations, non-zero initial conditions can be computed using a steady-state dose event record with SS>0 and AMT=0 and RATE=0, or using the I_SS (Initial Steady State) feature of MODEL and/or PK. Note that with either method, steady-state is computed only for compartments specified by differential equations. The equilibrium compartments are not computed (i.e., the algebraic equations are not evaluated) until the system is advanced beyond the initial value of TIME.
The CALL data item may be used to force a call to ADVAN, which will then evaluate the algebraic equations.
(See **Compartment_Initialization_Block**, **SS_dose**).
(See **$model**, **$pk**, **model**, **pk**).
(See **Initial_Steady_State:_I_SS,ISSMOD**).

ADVAN9 uses the Lawrence Livermore National Laboratory (LLNL) solver for differential-algebraic equations, implicit form (LSODI), using the backward differentiation formulas (BDF) for stiff problems.

ADVAN15 uses IDA from the LLNL SUNDIALS system for differential-algebraic equations and is a descendent of the LSODI1 (ADVAN9) system. ADVAN15 also has a root-finding algorithm. The user may choose to make modifications to parameters in ..\pr\IDAU.f90.
See ..\guides\idas_guide.pdf and ..\guides\ida_guide.pdf.
See **INTRODUCTION TO NONMEM 7, $SUBROUTINES: Yet Another New Differential Equation Solving Method: IDAS (ADVAN15)**

ADVAN16 and ADVAN17 use the RADAR5 Delay Differential Equation Solver. ADVAN17 may be used when there are equilibrium compartments.
See **INTRODUCTION TO NONMEM 7, Using the RADAR5 Delay differential equation Solver with the ddexpand program.**
See also ..\guides\manrad5-v2.pdf.

Required PREDPP Library subroutines:
ADVAN9 - ADDA, FCN5, JAC, LEQT22, LSODI1, RES, ZSPOW2
ADVAN15 - ADDA, FCN5, JAC, LEQT22, IDA, RES, ZSPOW2
ADVAN17 - JAC, LEQT22, RADAR5NM, RADAR5U, RES, ZSPOW2, FCN2, FCN4, FCN5

1 Analytical second derivatives used with the Laplacian method are not obtained with ADVAN9 and ADVAN15 and ADVAN17. Numerical second derivatives must be used.

REFERENCES: Guide VI, section VI.B , VII.C.9
REFERENCES: Guide IV, section V.C.8

ADVAN10

MEANING: Choice of Pharmacokinetic Model for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [ADVAN=]ADVAN10
```

SAMPLE:
```
$SUBROUTINE ADVAN10
```

DISCUSSION:
ADVAN10 is a routine in PREDPP's library which implements the kinetic equations for the One Compartment Model with Michaelis-Menten Elimination.

Compt. No.	Function	Initial Status	On/Off Allowed	Dose Allowed	Default for Dose	Default for Obs.
1	Central	On	No	Yes	Yes	Yes
2	Output	Off	Yes	No	No	No

TOL is required; (See **$SUBROUTINES**).

If steady-state doses are present, steady-state routine SS6 is used, and consequently a $DES block (or user-supplied DES routine) is required. Unless there are steady-state infusions, it may be a "dummy routine".
(See **Guide VI, section VII.C.6**)

TRANS routines that may be used: TRANS1

Basic PK parameter with TRANS1:

`VM` (maximum rate)
`KM` (Michaelis constant)

Additional PK parameters:

`S1` - Scale for central compartment (also called `SC`)
`S2` - Scale for output compartment (also called `S0`)
`F1` - Bioavailability for central compartment
`R1` - Rate for central compartment
`D1` - Duration for central compartment
`ALAG1` - Absorption lag for central compartment
`F0` - Output fraction (also called `F2`, `FO`)
`XSCALE` - Time scale
`MTIME(i)` - Model event times

Required PREDPP Library subroutines:
ADVAN10 - MMPHI

REFERENCES: Guide VI, section VII.C.6 , VII.C.10

ADVAN11

MEANING: Choice of Pharmacokinetic Model for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [ADVAN=]ADVAN11
```

SAMPLE:
```
$SUBROUTINE ADVAN11
```

DISCUSSION:
ADVAN11 is a routine in PREDPP's library which implements the kinetic equations for the Three Compartment Linear Model.

Compt. No.	Function	Initial Status	On/Off Allowed	Dose Allowed	Default for Dose	Default for Obs.
1	Central	On	No	Yes	Yes	Yes
2	Peripheral 1	On	No	Yes	No	No
3	Peripheral 2	On	No	Yes	No	No
4	Output	Off	Yes	No	No	No

TRANS routines that may be used: TRANS1, TRANS4, TRANS6

Basic PK parameters with TRANS1:

- `K` (rate constant of elimination)
- `K12` (rate constant from central to peripheral 1)
- `K21` (rate constant from peripheral 1 to central)
- `K13` (rate constant from central to peripheral 2)
- `K31` (rate constant from peripheral 2 to central)

Basic PK parameters with TRANS4: (See **TRANS4**).

Basic PK parameters with TRANS6: (See **TRANS6**).

Additional PK parameters:

- `S1` - Scale for central compartment (also called `SC`)
- `S2` - Scale for peripheral compartment 1
- `S3` - Scale for peripheral compartment 2
- `S4` - Scale for output compartment (also called `S0`)
- `F1` - Bioavailability for central compartment
- `F2` - Bioavailability for peripheral compartment 1
- `F3` - Bioavailability for peripheral compartment 2
- `R1` - Rate for central compartment
- `R2` - Rate for peripheral compartment 1
- `R3` - Rate for peripheral compartment 2
- `D1` - Duration for central compartment
- `D2` - Duration for peripheral compartment 1
- `D3` - Duration for peripheral compartment 2
- `ALAG1` - Absorption lag for central compartment
- `ALAG2` - Absorption lag for peripheral compartment 1
- `ALAG3` - Absorption lag for peripheral compartment 2
- `F0` - Output fraction (also called `F4`, `FO`)
- `XSCALE` - Time scale

`MTIME(i)` - Model event times

Steady-state subroutines: SS11, SS6

Other required user or library routines:

Library: EXP3

Notes:

1 Analytical second derivatives used with the Laplacian method are not obtained with ADVAN11. Numerical second derivatives must be used.

2 Can be greatly speeded up if calls to PK can be limited. (Use CALLFL=1 in $PK to call once per IR, and use the CALL data item to force additional calls if and when necessary; e.g., when covariates used in the PK model change.)

REFERENCES: None.

ADVAN12

MEANING: Choice of Pharmacokinetic Model for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [ADVAN=]ADVAN12
```

SAMPLE:
```
$SUBROUTINE ADVAN12
```

DISCUSSION:
ADVAN12 is a routine in PREDPP's library which implements the kinetic equations for the Three Compartment Linear Model with First Order Absorption.

Compt. No.	Function	Initial Status	On/Off Allowed	Dose Allowed	Default for Dose	Default for Obs.
1	Depot	Off	Yes	Yes	Yes	No
2	Central	On	No	Yes	No	Yes
3	Peripheral 1	On	No	Yes	No	No
4	Peripheral 2	On	No	Yes	No	No
5	Output	Off	Yes	No	No	No

TRANS routines that may be used: TRANS1, TRANS4, TRANS6

Basic PK parameters with TRANS1:

- `K` (rate constant of elimination)
- `K23` (rate constant from central to peripheral 1)
- `K32` (rate constant from peripheral 1 to central)
- `K24` (rate constant from central to peripheral 2)
- `K42` (rate constant from peripheral 2 to central)
- `KA` (rate constant of absorption)

Basic PK parameters with TRANS4: (See **TRANS4**).

Basic PK parameters with TRANS6: (See **TRANS6**).

Additional PK parameters:

- `S1` - Scale for depot compartment
- `S2` - Scale for central compartment (also called `SC`)
- `S3` - Scale for peripheral compartment 1
- `S4` - Scale for peripheral compartment 2
- `S5` - Scale for output compartment (also called `S0`)
- `F1` - Bioavailability for depot compartment
- `F2` - Bioavailability for central compartment
- `F3` - Bioavailability for peripheral compartment 1
- `F4` - Bioavailability for peripheral compartment 2
- `R1` - Rate for depot compartment
- `R2` - Rate for central compartment
- `R3` - Rate for peripheral compartment 1
- `R4` - Rate for peripheral compartment 2
- `D1` - Duration for depot compartment
- `D2` - Duration for central compartment
- `D3` - Duration for peripheral compartment 1

D4 - Duration for peripheral compartment 2
ALAG1 - Absorption lag for depot compartment
ALAG2 - Absorption lag for central compartment
ALAG3 - Absorption lag for peripheral compartment 1
ALAG4 - Absorption lag for peripheral compartment 2
F0 - Output fraction (also called F5, FO)
XSCALE - Time scale
MTIME(i) - Model event times

Steady-state subroutines: SS12, SS6

Other required user or library routines:

Library: EXP3 EXP4

Notes:

1 Analytical second derivatives used with the Laplacian method are not obtained with ADVAN12. Numerical second derivatives must be used.

2 Can be greatly speeded up if calls to PK can be limited. (Use CALLFL=1 in $PK to call once per IR, and use the CALL data item to force additional calls if and when necessary; e.g., when covariates used in the PK model change.)

REFERENCES: None.

ADVAN7: FACTOR

MEANING: PREDPP global variables
CONTEXT: For use with PREDPP

USAGE:

```
$PK
"FIRST
" USE PRCOM_REAL, ONLY: FACTOR=>FAC
"MAIN
" FACTOR=10.
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: DPSIZE
REAL(KIND=DPSIZE) :: FAC
```

DISCUSSION:

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of PREDPP code.

USAGE:

A message similar to the following is sometimes produced by ADVAN7:

```
NUMERICAL DIFFICULTIES OBTAINING THE SOLUTION.  NON-REAL
EIGENVALUES. PERHAPS ADVAN5 SHOULD BE USED FOR THIS PROBLEM.
"LARGEST" VALUE ALONG SUBDIAGONAL OF SCHUR MATRIX:  0.423E-01
(DIAG: 0.84E+00 0.84E+00)
```

FACTOR is a "fudge factor" that affects ADVAN7's determination of how large the subdiagonal element is relative to the two diagonal elements. The default value of FACTOR is 1.

These error messages arise when the computations of the eigenvalues produce unexpected values. The question is whether the subdiagonal elements are too large to ignore (e.g., when the eigenvalues are truly complex) in which case ADVAN5 must be used, or whether trivially small quantities have been produced by the accumulation of rounding error in the floating point computation, in which ADVAN7 can be used. (Non-real eigenvalues result when the system is non-mamillary, and may also occasionally occur when there are etas associated with rate constants Kij).

If the maximum value printed is very small (e.g., E-15), then the computations will proceed correctly using a value of FACTOR such as 10 or 100. Values such as FACTOR=10 or FACTOR=100 are more permissive, and allow larger subdiagonal elements. If no reasonable value of FACTOR eliminates the error messages, then ADVAN5 should be used.

Location prior to NONMEM 7: prcomw

REFERENCES: None.

AES

MEANING: AES subroutine
CONTEXT: User-supplied subroutine; for use with PREDPP's ADVAN9 and ADVAN15 and ADVAN17

USAGE:

```
SUBROUTINE AES (INIT,A,P,T,E,IR,DA,DP,DT)
USE SIZES, ONLY: DPSIZE,ISIZE
INTEGER (KIND=ISIZE) :: INIT,IR
REAL(KIND=DPSIZE) :: A,P,E,DA,DP,DT
DIMENSION :: A(*),P(*),E(*),DA(IR,*),DP(IR,*),DT(*)
```

DISCUSSION:
The AES subroutine is called by PREDPP to evaluate algebraic expressions for ADVAN9, ADVAN15, and ADVAN17.

Input argument:

`P(n)`
: The value of the nth PK parameter.

`T` Time. `T` takes values continuously over an integration interval.

Input/output arguments:

`INIT`
: When AES is called with INIT=1, this is an initial condition call at the start of an integration interval. Approximate amounts in each equilibrium compartment n at time T must be stored in A(n). (The amounts in the non-equilibrium compartments are already stored in the lower-numbered elements of A.) DA, DP, DT need not be computed.

  If AES stores approximate values in A, it must set INIT=0.
  If AES stores exact values in A, it must leave INIT unchanged.

  When AES is called with INIT=2, the values of the algebraic expressions must be stored in E and the derivatives in DA, DP, and DT.

`A(n)`
: The amount in the nth compartment at time T.

Output argument:

`E(k)`
: The value of the algebraic expression g(k).

`DA(k,j)`
: The derivative of g(k) with respect to A(j).

`DP(k,j)`
: The derivative of g(k) with respect to P(j).

`DT(k)`
: The derivative of g(k) with respect to T.

Also see variables in NONMEM modules, NONMEM-PRED modules, and PREDPP modules.
(See **variables in modules**)
In particular,
(See **DES AES: ICALL,IDEFD,IDEFA**)

REFERENCES: Guide IV, section V.C.8 , V.C.9
REFERENCES: Guide VI, section VI.E

AMT DATA ITEM

MEANING: Dose Amount (AMT) data item for PREDPP
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... AMT ...`

DISCUSSION:
`AMT` labels PREDPP's dose amount (AMT) data item. The dose amount data item is optional.

AMT must be 0 for non-dose event records.

AMT must be 0 for "steady-state with constant infusion" dose or reset-dose event records.

AMT must be a positive number in all other dose or reset-dose event records.

(See **dose**, **steady_state_dose**).

REFERENCES: Guide VI, section V.D
REFERENCES: Guide V, section 6.8

ANEAL

MEANING: CONSTRAINT subroutine
CONTEXT: NONMEM utility routine

USAGE:

```
$SUBROUTINES ADVAN3 TRANS4 OTHER=aneal.f90
```

DISCUSSION:

Additional algorithmic constraints may be imposed upon the model parameters, by use of the subroutine CONSTRAINT. This feature is available only for the EM and Bayesian algorithms.

One use would be to slow the rate of reduction of the diagonal elements of the OMEGA values during the burn-in phase of the SAEM method. This is shown in example 9, where a user supplied annealing algorithm is used to replace the built-in one described earlier. By specifying OTHER=ANEAL.f90, where ANEAL.f90 was originally derived from a template of CONSTRAINT.f90 in the source directory, the user supplied CONSTRAINT subroutine can be incorporated into the model.

In example 9, whenever the argument iteration number (ITER_NO) of CONSTRAINT changes, a new OMEGA is evaluated that is larger than what was determined by the SAEM update. Typically, this expansion algorithm should be such that its impact decreases with each iteration. This example is given in example 9.

See **INTRODUCTION TO NONMEM 7, $ANNEAL to facilitate EM search methods**

REFERENCES: Guide Introduction_7

BIOAVAILABILITY BEHAVIOR

MEANING: PREDPP global variables
CONTEXT: For use with PREDPP

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of PREDPP code.

USAGE:

```
$PK
"FIRST
" USE PRCOM_LOG, ONLY: NEWBIO=>NEWWAY
"MAIN
" NEWBIO=.FALSE.
```

GLOBAL DECLARATION:

```
LOGICAL NEWWAY
```

DISCUSSION:
This variable allows the user to change the way bioavailability fraction parameters are used by PREDPP. Prior to 1990 (i.e., with NONMEM III / PREDPP I) bio-availability fractions applied to all transient doses *except* infusions (doses with AMT > 0 and RATE > 0). Similarly, it did not apply to steady-state doses with multiple infusion and with a positive rate data item.

After the change, bio-availability fractions apply to all transient (non steady-state) doses, and to steady-state doses for which the AMT is specified. The only doses to which bio-availability fractions do not apply are steady-state with constant infusion.

By default, NEWBIO is true and the new rules apply. If NEWBIO is set to false, bioavailability fraction parameters revert to their pre-1990 behavior.

Location prior to NONMEM 7: prcomu

REFERENCES: None.

BIOAVAILABILITY FRACTION PARAMETER

MEANING: Bioavailability fraction (F) parameter for PREDPP
CONTEXT: Additional PK Parameters

USAGE:

```
$PK
 F1= ....
```

DISCUSSION:

Bioavailability fraction parameters are used with PREDPP. They are optional additional PK parameters. With each possible dose compartment of the kinetic model (the output compartment is not a possible dose compartment), there is associated one bioavailability fraction (parameter), symbolized in the $PK record by the reserved variable `F`n, where n is the number of the compartment.

If the amount on a dose record is A, and the bioavailability fraction is F, only the fraction F of A actually appears in the dose compartment (either instantaneously at the time the dose enters the compartment - with a bolus dose, or over a period of time - with a regular infusion; the bioavailability fraction does not apply to a steady-state infusion).

Bioavailability fractions are optional in the sense that bioavailability fractions associated with compartments never used as dose compartments may be ignored. The values of bioavailability fractions that are not computed in PK are always understood to be 1.

When additional doses are specified on a dose event record, the bioavailability fraction applies to the dose and to all the additional doses. When a steady-state dose is specified on a dose event record, the bioavailability fraction applies to the dose and to all implied doses leading to the steady-state.

(See **cmt**, **pk**, **$pk**, **default_compartment**).

REFERENCES: Guide IV, section V.C.5
REFERENCES: Guide V, section 6.8 , 7.4.4
REFERENCES: Guide VI, section III.F.6 , V.F.1 , V.H

CALL DATA ITEM

MEANING: Call (CALL) data item for PREDPP
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... CALL ...`

DISCUSSION:
`CALL` labels PREDPP's call (CALL) data item. The call data item is optional. It is used to force calls to PK and/or ERROR subroutines with event records for which such calls would not normally occur. (When $PK and/or $ERROR blocks are present, the effect of calling these routines with an event record is to evaluate the $PK and/or $ERROR abbreviated code for that particular event record.)

Values are:

0 (Default) No forced call; PREDPP takes its normal action.

1 Force a call to ERROR with the event record.

2 Force a call to PK with the event record.

3 Force calls to both PK and ERROR with the event record.

10 Call ADVAN9, ADVAN15, or ADVAN17 with the event record. May be used whether or not the TIME data item is defined. May be combined with values 1, 2, and 3. e.g., CALL = 11 forces calls to ADVAN9, ADVAN15, or ADVAN17 and to ERROR with the event record.

If the PK routine is not called with a given event record, then the basic and additional PK parameters retain the values they had after the last call to PK. This would cause a problem if some parameter depends on an data item whose value is different on the given event record than on the last event record for which PK was called.

If the ERROR routine is not called with a given event record, then NONMEM's PRED value is a scaled compartment amount. This would cause a problem if the event record is an other-type event which was included in the data set in order to obtain a prediction of (say) a pharmacodynamic effect.

The pattern of calls to ADVAN9, ADVAN15, ADVAN17 depends on the presence or absence of the TIME data item.

If TIME is defined, ADVAN9, ADVAN15, or ADVAN17 is called by default exactly as other ADVAN routines are called: when TIME increases. CALL=10 may be used to obtain additional calls. For example, immediately following a bolus dose event record one might include an other-type event record having the same value of TIME and CALL=10. ADVAN9, ADVAN15, or ADVAN17 is called, and the amounts in the equilibrium compartments are computed based on the amounts in the other compartments.

If TIME is not defined, ADVAN9, ADVAN15, or ADVAN17 is called by default with every event record unless the AES routine specifies the calling protocol "call once per individual record". In this case, CALL=10 may be used to force calls to the ADVAN routine for records subsequent to the first.

REFERENCES: Guide VI, section V.J
REFERENCES: Guide IV, section V.C.1

CALLFL

MEANING: Control calls to PK and ERROR subroutine and ADVAN9/15/17
CONTEXT: Abbreviated code

USAGE:
`CALLFL=n`

DISCUSSION:
This pseudo-assignment statement may be present in $PK abbreviated code, in which case it controls when PREDPP calls the PK routine. It may be present in $ERROR abbreviated code, in which case it controls when PREDPP calls the ERROR routine. It may be present in $AESINITIAL abbreviated code when the TIME data item is not used, in which case it controls when PREDPP calls the ADVAN9, ADVAN15, or ADVAN17 routine. The use of CALLFL in each of these three abbreviated codes is independent of its use in the others. The pseudo-assignment statement may be enclosed in parentheses. A calling protocol phrase may be used within parentheses instead of a pseudo-assignment statement CALLFL (See **calling protocol**). The pseudo-assignment statement may take these forms:

`CALLFL=-2`

Call the PK subroutine with every event record, with additional and lagged doses, and at modeled event times. Does not apply to the ERROR or ADVAN9, ADVAN15, or ADVAN17 routines.

The following messages will appear in the NONMEM output report:

```
PK SUBROUTINE CALLED WITH EVERY EVENT RECORD
PK SUBROUTINE CALLED AT NONEVENT (ADDITIONAL AND LAGGED) DOSE TIMES
```

or

```
PK SUBROUTINE CALLED AT NONEVENT (ADDITIONAL AND LAGGED) DOSE TIMES
AND AT MODEL TIMES
```

`CALLFL=-1`

Call the subroutine with every event record. This is the default.

Some of the following messages will appear in the NONMEM output:

```
PK SUBROUTINE CALLED WITH EVERY EVENT RECORD
PK SUBROUTINE NOT CALLED AT NONEVENT (ADDITIONAL OR LAGGED) DOSE TIMES

ERROR SUBROUTINE CALLED WITH EVERY EVENT RECORD

ADVAN9 CALLED WITH EVERY EVENT RECORD
ADVAN15 CALLED WITH EVERY EVENT RECORD
ADVAN17 CALLED WITH EVERY EVENT RECORD
```

`CALLFL=0`

For the PK subroutine: If the data are population data, call the subroutine with the first event record of each individual record; if the data are single-subject data, call the subroutine with the first event record of the data set. In addition, call the subroutine with with every event record where the event time differs from the previous event time.

The following messages will appear in the NONMEM output report:

```
PK SUBROUTINE CALLED ONLY WITH NEW INDIVIDUAL OR NEW TIME
```

```
PK SUBROUTINE NOT CALLED AT NONEVENT (ADDITIONAL DOSE OR LAGGED) DOSE TIMES.
```

For the ERROR subroutine: When the Simulation Step is being implemented, call the subroutine with every event record. Otherwise, call the subroutine only with observation event records.

The following messages will appear in the NONMEM output report:

```
DURING SIMULATION, ERROR SUBROUTINE CALLED WITH EVERY EVENT RECORD
OTHERWISE, ERROR SUBROUTINE CALLED ONLY WITH OBSERVATION EVENTS
```

Does not apply to the ADVAN9, ADVAN15, or ADVAN17 routine.

With `CALLFL=0`, The CALL data item may be used to request calls with additional event records.

`CALLFL=1`

For the PK and ADVAN9, ADVAN15, or ADVAN17 subroutines: If the data are population data, call the subroutine with the first event record of each individual record; if the data are single-subject data, call the subroutine with the first event record of the data set.

The following messages appear in the NONMEM output report:

```
PK SUBROUTINE CALLED ONCE PER INDIVIDUAL RECORD
PK SUBROUTINE NOT CALLED AT NONEVENT (ADDITIONAL DOSE OR LAGGED) DOSE TIMES
```

```
ADVAN9 CALLED ONCE PER INDIVIDUAL RECORD.
ADVAN15 CALLED ONCE PER INDIVIDUAL RECORD.
ADVAN17 CALLED ONCE PER INDIVIDUAL RECORD.
```

For the ERROR subroutine: When the Simulation Step is being implemented, call the subroutine with every event record. Otherwise, if the data are population data, call the subroutine with the first event record of each individual record; if the data are single-subject data, call the subroutine with the first event record of the data set.

The following messages appear in the NONMEM output report:

```
DURING SIMULATION, ERROR SUBROUTINE CALLED WITH EVERY EVENT RECORD
OTHERWISE, ERROR SUBROUTINE CALLED ONCE PER INDIVIDUAL RECORD
```

With `CALLFL=1`, The CALL data item may be used to request calls with additional event records.

In a block of abbreviated code, the CALLFL=n pseudo-assignment statement must precede all the other abbreviated code (except for verbatim code or other pseudo-assignment statements). The pseudo-assignment statement may not be used conditionally. `CALLFL` may not be used as a variable elsewhere in the abbreviated code.

Note: If the $ERROR record consists of exactly one of these four statements:

```
Y=F+ERR(1)
Y=F*(1+ERR(1))
Y=F+F*ERR(1)
Y=F*EXP(ERR(1))
```

with *no* other lines of code (except for comment lines), NM-TRAN will automatically limit calls to ERROR to once-per-problem (unless the Simulation Step is being

implemented, in which case. calls are made with every event record). In effect, this amounts to yet another way to control when it is that calls may occur to the ERROR routine, but one which may not be explicitly specified in $ERROR via the use of `CALLFL`.

With the last three models (proportional and exponential), NM-TRAN will also cause PREDPP to output the message:

```
ERROR IN LOG Y IS MODELED
```

(This does not mean that a model is fit to Log Y data.)

REFERENCES: Guide IV, section V.C.5

CALLING PROTOCOL PHRASE

MEANING: Control calls to PK and ERROR and ADVAN9, ADVAN15, and ADVAN17
CONTEXT: Abbreviated code

USAGE:

```
$PK (ONCE PER INDIVIDUAL RECORD)
$ERROR (ONCE PER INDIVIDUAL RECORD)
$AESINITIAL (ONCE PER INDIVIDUAL RECORD)
```

DISCUSSION:
A calling protocol phrase can be used instead of the `CALLFL` pseudo-statement in $PK, $ERROR, and $AESINITIAL abbreviated codes. The phrase must be enclosed in parentheses. Either upper or lower case may be used. In an abbreviated code, the line of code containing the phrase must precede all the other abbreviated code (except for verbatim code or other pseudo-assignment statements), and it may be the same line that marks the beginning of the code, as in the above usage examples. Pseudo-statements defining COMRES may be coded within the same parentheses, separated by a semicolon ";". No abbreviated code may follow ")" on the same line as the ")".

Phrases equivalent to `CALLFL=-2` for the $PK record:

```
(NON-EVENT)
(ADDITIONAL OR LAGGED)
```

The following messages will appear in the NONMEM output report:

```
PK SUBROUTINE CALLED WITH EVERY EVENT RECORD
PK SUBROUTINE CALLED AT NONEVENT (ADDITIONAL AND LAGGED) DOSE TIMES
```

or

```
PK SUBROUTINE CALLED AT NONEVENT (ADDITIONAL AND LAGGED) DOSE
TIMES AND AT MODEL TIMES
```

Phrases equivalent to `CALLFL=-1` for the $PK, $ERROR and $AESINITIAL records (the default):

```
(EVERY EVENT)
(EVERY)
```

The following messages will appear in the NONMEM output report (as appropriate):

```
PK SUBROUTINE CALLED WITH EVERY EVENT RECORD
PK SUBROUTINE NOT CALLED AT NONEVENT (ADDITIONAL OR LAGGED) DOSE TIMES

ERROR SUBROUTINE CALLED WITH EVERY EVENT RECORD

ADVAN9 CALLED WITH EVERY EVENT RECORD.
ADVAN15 CALLED WITH EVERY EVENT RECORD.
```

Phrases equivalent to `CALLFL=0` for the $PK record:

```
(NEW TIME)
(NEW EVENT TIME)
```

The following messages appear in the NONMEM output report:

```
PK SUBROUTINE CALLED ONLY WITH NEW INDIVIDUAL OR NEW TIME
PK SUBROUTINE NOT CALLED AT NONEVENT (ADDITIONAL OR LAGGED) DOSE TIMES.
```

Phrases equivalent to CALLFL=0 for the $ERROR record:

```
(OBSERVATION EVENT)
(OBS)
(OBSERVATION ONLY)
(OBS ONLY)
```

One or more of the following messages appear in the NONMEM output report when calls are limited in this manner:

```
DURING SIMULATION, ERROR SUBROUTINE CALLED WITH EVERY EVENT RECORD
OTHERWISE, ERROR SUBROUTINE CALLED ONLY WITH OBSERVATION EVENTS
```

Phrases equivalent to CALLFL=1 for the $PK, $ERROR and $AESINITIAL blocks:

```
(ONCE PER INDIVIDUAL RECORD)
(ONCE/IND.REC.)
```

The following messages appear in the NONMEM output (as appropriate):

```
PK SUBROUTINE CALLED ONCE PER INDIVIDUAL RECORD
PK SUBROUTINE NOT CALLED AT NONEVENT (ADDITIONAL OR LAGGED) DOSE TIMES
```

```
DURING SIMULATION, ERROR SUBROUTINE CALLED WITH EVERY EVENT RECORD
OTHERWISE, ERROR SUBROUTINE CALLED ONCE PER INDIVIDUAL RECORD
```

```
ADVAN9 CALLED ONCE PER INDIVIDUAL RECORD.
ADVAN15 CALLED ONCE PER INDIVIDUAL RECORD.
```

REFERENCES: Guide IV, section V.C.5

CCONTR

MEANING: CCONTR subroutine
CONTEXT: User-supplied subroutine; replaces a NONMEM dummy routine

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of NONMEM code and previous examples that may be available from advanced users.

USAGE:

```
SUBROUTINE CCONTR (I,CNT,P1,P2,IER1,IER2)
USE SIZES, ONLY: ISIZE,DPSIZE
INTEGER(KIND=ISIZE), INTENT(IN OUT)     :: I,IER1,IER2
REAL(KIND=DPSIZE),   INTENT(IN OUT)     :: CNT,P1(:),P2(:,:)
```

DISCUSSION:
CCONTR is a user-supplied routine for computing the contribution made to the objective function from an L2 record. It is used to override the NONMEM default objective function.

CCONTR may be used when there are no epsilons or etas and in other situations, e.g., with categorical population data.

NONMEM sometimes produces this error message:

```
USER CCONTR ROUTINE NOT USED, BUT THERE ARE NO EPSILONS,
AND ETAS ARE TO BE ESTIMATED
```

This can happen when POSTHOC etas are requested, but the data are single-subject data. The user may have included the POSTHOC option in error.

The CCONTR routine may contain the following code:

```
CALL CELS (CNT,P1,P2,IER1,IER2)
```

If the data is correlated across L2 records (e.g., auto-regressive data), CCONTR computes the contribution to the objective function for data from an entire individual record.

Input argument:

`I` Similar to Ifor PRED subroutine
Possible values: 0, 1, 2

Output argument:

`CNT`
The conditional contribution to the objective function for this L2 record.

`P1` P1(i) is the partial derivative of CNT with respect to eta(i).

`P2` P2(i,j) is the second partial derivative of CNT with respect to eta(i) and eta(j) (for i <= j).
At ICALL=1, CCONTR sets P2(1,1)=-1 if second derivative values are to be used. Otherwise, if cross-gradient values are to be used.

`IER1`
0 - Normal return.
non-zero - error return.

`IER2`
0 if error-recovery is to be implemented when IER1 is nonzero.
1 if NONMEM is to stop when IER1 is nonzero.

Other inputs

Other inputs are available to CCONTR in NONMEM read-only global variables.
(See **MIX_CONTR:_THETA**)
E.g., USE ROCM_REAL, ONLY: THETA=>THETAC
(See **CCONTR:_Y,DATA,N1,N2,DIM**)
(See **CCONTR:_F,G,H**)

CCONTR is called by NONMEM's NCONTR routine with one level 2 ("L2") record after another. If no CCONTR routine is supplied by the user, NCONTR calls NONMEM utility CELS ("Conditional ELS"; ELS contribution conditional on knowing eta). If no L2 data item is present, each level 2 record is a single observation event record. Otherwise, it is a group of observation records grouped together by a common value of L2.

REFERENCES: Guide IV, section III.B.4

CCONTR: F,G,H

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of NONMEM code and previous examples that may be available from advanced users.

USAGE:

```
USE ROCM_REAL, ONLY: F=>FL2,G=>GL2,G2=>GGL2,H=>HL2
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: NO,LVR,LVR2,DPSIZE
REAL(KIND=DPSIZE) :: FL2(NO),GL2(NO,LVR),GGL2(NO,LVR2*(LVR2+1)/2), &
                     HL2(NO,LVR*LVR/4+LVR/2)
```

DISCUSSION:

These variables change values with each L2 record. They may be used by CCONTR.

`F` F(n) = value returned in F from PRED for the nth observation of the L2 record.

`G` G(n,i) = Partial derivative of F(n) with respect to eta(i)

`G2` G2(n,i*(i-1)/2+j) = second partial derivative of F(n) with respect to eta(i) and eta(j) (j<=i)
G2 is arranged in symmetric storage, e.g.

1
2 3
4 5 6

G2(n,1) = 2nd. partial of F(n) wrt. eta(1) eta(1)
G2(n,2) = 2nd. partial of F(n) wrt. eta(2) eta(1)
G2(n,3) = 2nd. partial of F(n) wrt. eta(2) eta(2)
G2(n,4) = 2nd. partial of F(n) wrt. eta(3) eta(1)
G2(n,5) = 2nd. partial of F(n) wrt. eta(3) eta(2)
G2(n,6) = 2nd. partial of F(n) wrt. eta(3) eta(3)

`H` H(n,i) = Partial derivative of F(n) with respect to eps(i) for i=1,neps, where neps is the number of epsilons in the problem.
H(n,j*neps+i) = second partial derivative of F(n) with respect to eps(i) and eta(j).

E.g. suppose that there are two epsilons in the problem:

1 3 5
2 4 6

H(n,1) = partial derivative of F(n) wrt. eps(1)
H(n,2) = partial derivative of F(n) wrt. eps(2)
H(n,3) = 2nd. partial of F(n) wrt. eps(1) eta(1)
H(n,4) = 2nd. partial of F(n) wrt. eps(2) eta(1)
H(n,5) = 2nd. partial of F(n) wrt. eps(1) eta(2)
H(n,6) = 2nd. partial of F(n) wrt. eps(2) eta(2)

Location prior to NONMEM 7: rocm5

REFERENCES: None.

CCONTR: Y,DATA,N1,N2,DIM

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of NONMEM code and previous examples that may be available from advanced users.

USAGE:

```
USE ROCM_REAL, ONLY:  Y=>DV_ITM2,DATA=>CDATA
USE ROCM_INT, ONLY: N1=>NL2_OBS,N2=>L2NO,DIM=>L2CR_DIM
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: NO,DPSIZE
REAL(KIND=DPSIZE) :: DV_ITM2(NO),CDATA(NO,3)
INTEGER(KIND=ISIZE) :: NL2_OBS,L2NO,L2CR_DIM(NO)
```

DISCUSSION:

These variables change values with each L2 record. They may be used by CCONTR. When the epsilons are correlated between L2 records these variables change values with each L1 record.
(See **Correlation_Across_L2_Records**)

`Y(k)`
: DV data item on the kth observation record of the L2 (L1) record.

`DATA(k,i)`
: The ith type of data item specified in NM-TRAN's $CONTR record, appearing on the kth observation record of the L2 (L1) record.

`N1` Number of observations in the L2 record. When the epsilons are correlated between L2 records, N1 is the number of L2 records within the L1 record.

`N2` Number of the L2 record within the L1 record. When the epsilons are correlated between L2 records, N2 is 0.

`DIM`
: When the epsilons are correlated between L2 records, DIM gives the lengths of the L2 records.

Location prior to NONMEM 7: rocm3

REFERENCES: None.

CELS

MEANING: CELS subroutine
CONTEXT: NONMEM utility routine

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of NONMEM code and previous examples that may be available from advanced users.

DISCUSSION:
CELS is a NONMEM utility routine. It may be called by a user-supplied CCONTR routine. It computes the default contribution to the conditional objective function from a L2 record.

Certain NONMEM error messages refer to CELS, for example:

```
PROGRAM TERMINATED BY OBJ, ERROR IN CELS
WITH INDIVIDUAL    1 (IN INDIVIDUAL RECORD ORDERING)
INTRAINDIVIDUAL VARIANCE OF DATA FROM OBS RECORD  1 ESTIMATED TO BE 0
```

A common situation giving rise to this message is the use of a proportional intra-individual error model while some predicted values for actual observations are zero or close to zero. (For example, if the first dose is an infusion and there is a "base- line" observation at the start of the infusion, the predicted level will be zero.)

CELS must not be called when a user-supplied CRIT routine is used.

REFERENCES: None.

CHOL

MEANING: CHOL subroutine
CONTEXT: NONMEM utility routine

USAGE:

```
USE SIZES, ONLY:: ISIZE,DPSIZE
INTEGER(KIND=ISIZE) :: MODE,N,IER
REAL(KIND=DPSIZE) :: R
DIMENSION R(5)
CALL CHOL (MODE,N,R,IER)
```

DISCUSSION:
CHOL is a NONMEM utility routine that may be called by any user routine. It computes the Cholesky square root of a given positive definite matrix A, i.e., the lower triangular matrix B satisfying BB' = A.

Input argument:

`MODE`
An integer variable or integer constant.
0 - the required initialization call.
1 - a normal call.

`N` An integer variable or integer constant between 1 and 5; the dimension of the matrix.

`R` An array of dimension at least N.
The ith row of the matrix A, at the i+1 st. call to CHOL. Elements R(j) with j>i are ignored.

`IER`
Upon input when MODE is 0:
0 - CHOL should terminate NONMEM execution when the matrix is algorithmically not positive definite.
1 - CHOL should always return control to PRED.

Output argument:

`R` The ith row of B, upon return from the i+1 st. call to CHOL.

`IER`
0 - Normal return.
1 - matrix is algorithmically not positive definite. (This can only happen if IER was 1 at the call with MODE=0.)

If the dimension of A is NxN, then CHOL is called N+1 times. The first call initializes the routine. On the i+1 st. call, the ith row is passed to CHOL, and CHOL returns the ith row of B (i=1,..., N).

REFERENCES: Guide II, section D.4 , D.4.2
REFERENCES: Guide II, Figure 20

CMT PCMT DATA ITEM

MEANING: Compartment (CMT) and Prediction Compartment (PCMT) data items for PREDPP
CONTEXT: $INPUT record and NONMEM data set

USAGE:
```
$INPUT ... CMT PCMT ...
```

DISCUSSION:
`CMT` and `PCMT` label PREDPP's compartment (CMT) and prediction compartment (PCMT) data items. The compartment and prediction compartment data items are optional. They may be used separately or together. Their meanings depend on the particular kind of event record.

Observation Event:

CMT specifies the number of the observation compartment (the compartment whose scaled drug amount appears as `F` in the error routine). When this number is that of the output compartment, it is allowed to have a negative sign, in which case the output compartment is turned off after `F` is obtained.

This is also the case for an output-type compartment, which is a compartment with attributes INITIALOFF and NODOSE. (With a compartment n which is not the output compartment or an output-type compartment, two records must be used, with the same TIME value: the first with CMT=n and EVID=1 to obtain the observation, and the second with CMT=-n and EVID=2 to turn the compartment off.)

By convention, the number of the output compartment may also be given by the CMT data item as the number 1000 (this is true regardless of the type of the event record). With small models (number of compartments <= 99), this number may also be given as 100. With versions of NONMEM prior to 7.3, only small models are possible, and only 100 may be used.

When CMT is 0, the default compartment for observations is used.
PCMT is ignored.

Dose Event:

CMT specifies the number of the compartment into which the dose is introduced. The dose compartment is turned on if it was previously off. When CMT is 0, the default compartment for doses receives the dose.
PCMT specifies the number of the compartment for which an `F` value is computed.
PCMT may be specified as 1000 or 100 to indicate the default compartment for output, similarly to CMT above.
When PCMT is 0, the default compartment for observations is used.

Other-Type Event:

A positive value of CMT specifies a compartment to be turned on.
A negative value of CMT specifies a compartment to be turned off.
When CMT is 0, no change is made in the status of any compartment.
PCMT is the same as for dose events.

Reset Event:

CMT is ignored.
PCMT is the same as for dose events.

Reset-Dose Event:

CMT and PCMT are the same as for dose events.

Guide V (Introductory Guide) has an example of how urine collections and observations of urine concentration can be described in the data file, along with observations of plasma concentrations. See Chapter 6, Section 9, The Output Compartment: Urine Collections and Observations

REFERENCES: Guide VI, section V.H
REFERENCES: Guide V, section 6.9

COM COMACT COMSAV COMRES

MEANING: Variables related to the module NMPRD4
CONTEXT: Abbreviated code, verbatim code, user-supplied routines, NM-TRAN

Reserved labels starting `COM` are related to each other. They all refer in some manner to the module NMPRD4. (This module was originally a global COMMON; hence the use of the letters "COM".)

`COM(n)` (in $TABLE and $SCATTER records)

Certain positions of MODULE NMPRD4 may be reserved, and the variables stored in these positions may be displayed by listing them as `COM(1)`, `COM(2)`, etc. on $TABLE and $SCATTER records, e.g.,

```
$TABLE COM(1) COM(2)
```

`COM(n)` (in abbreviated and verbatim code)

Certain positions of MODULE NMPRD4 may be reserved, and the variables stored in these positions are referenced in abbreviated and verbatim code as `COM(1)`, `COM(2)`, etc.

`COMACT` (in any user code)

Reserved variable `COMACT` is set by NONMEM. It may be tested in PRED (e.g., in abbreviated code, verbatim code, or in user-written routines) to determine when NONMEM is making a copying pass, i.e., when the data records are being passed to PRED for the purpose of computing values of variables which will be obtained (i.e. copied) from NMPRD4 for tables and scatterplots. NONMEM only makes a copying pass when PRED-defined items are listed in $TABLE or $SCATTER records. There may be a few copying passes. With the first copying pass, the value of COMACT is 1. As copying passes proceed, the value of COMACT may remain the same or increase. The values used in tables and scatterplots are those copied from NMPRD4 with the last copying pass.

`COMACT=0`: This is not a copying pass.

`COMACT=1`: This is a copying pass with final thetas and zero-valued etas.

`COMACT=2`: This is a copying pass with final thetas and conditional estimates of etas.

`COMACT=3`: This is a copying pass with conditional (nonparametric) estimates of etas.

Such a pass takes place when the control stream includes this record:

```
$NONPARAMETRIC ETAS
```

`COMRES=`n_1 (option of $ABBREVIATED record)

`COMRES` ("common reserve") is an option of the $ABBREVIATED record. It gives instructions to NM-TRAN about NMPRD4.

`COMRES=-1`: Do not store any variables in module NMPRD4.

`COMRES=0`: Store variables in NMPRD4, but do not reserve any positions (the default).

`COMRES=`n_1: Store variables in NMPRD4, and reserve the first n_1 positions.

`COMRES=-1` (in abbreviated code)

The pseudo-assignment statement `COMRES=-1` may be used in any block of abbreviated code to prevent any variable defined in the block from being stored into NMPRD4.

COMSAV=n_2 (option of $ABBREVIATED record)

Values of variables displayed in tables and scatterplots are obtained from module NMPRD4. There are particular times when data records are passed to PRED for the purpose of obtaining these values; these are called copying passes. The SAVE region of module NMPRD4 is the initial part of NMPRD4. If a variable is stored in the SAVE region, then the value of the variable computed with a given data record during a copying pass will be found in NMPRD4 when the same record is passed during the next copying pass, i.e. it will have been saved from the previous copying pass. This is in contrast to the usual behaviour, where with a given data record, the value in NMPRD4 is the value computed with the previous data record.

n_2 is the initial size of the SAVE region, i.e. the number of positions in this region. $n_2 = 0$ is the default value. n_2 may not exceed n_1.

The SAVE region has size n_2 initially, but NM-TRAN may extend it if SAVE variables are used. However, if $n_2 = -1$, the SAVE region is not to be extended, and there is to be no SAVE region altogether.

(See **copying_block**).

NM-TRAN causes the generated routines to store the value of COMSAV at ICALL=1.

COMSAV=n_2 (in user written code)

In the absence of abbreviated code, COMSAV may be set by a user-written PRED routine at ICALL<=1, or at COMACT=1 with the very first data record. n_2 is as described above.

(See **COMACT,COMSAV**)
(See **PRED-Defined Variables**)
(See **abbreviated**, **abbreviated_code**, **displayed_values**).

REFERENCES: Guide IV, section III.B.7 , IV.E.2
REFERENCES: Guide VI, section III.J

COMACT,COMSAV

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:
```
USE NMPRD_INT, ONLY: COMACT,COMSAV
```

GLOBAL DECLARATION:
```
INTEGER(KIND=ISIZE) :: COMACT,COMSAV
```

DISCUSSION:

COMACT is set by NONMEM; COMSAV is set by PRED.

COMACT

COMACT=0: NONMEM is not making a copying pass, i.e., values from NMPRD4 will not be copied for tables and scatterplots. (NONMEM only makes a copying pass when PRED-defined items are listed in $TABLE or $SCATTER records.)
COMACT=1: This is a copying pass with final thetas; etas are zero.
COMACT=2: This is a copying pass with final thetas and conditional eta estimates.
COMACT=3: This is a copying pass with conditional (nonparametric) estimates of etas. Such a pass takes place when the control stream includes this record:

```
$NONPARAMETRIC ETAS
```

COMSAV

Set by PRED at ICALL<=1 or at COMACT=1 with the very first data record. The Save Region is an initial part of NMPRD4. If with a given data record, the value of a PRED-defined variable is stored in this region, then during a copy pass with COMACT = 2, the value of the variable is initialized to the value obtained during the copy pass with the same data record and COMACT=1. COMSAV is the size of this region, i.e., the number of variables whose values are stored in this region.

When NM-TRAN is used, COMACT may be tested in abbreviated code. COMSAV may not be referenced in abbreviated code. Instead, the COMSAV option of the $ABBREVIATED record is coded, and NM-TRAN causes the specified value to be stored in COMSAV.

Location prior to NONMEM 7: nmprd3

REFERENCES: Guide IV, section III.B.7 , IV.E.2
REFERENCES: Guide VI, section III.J , IV.E

COMPARTMENT INITIALIZATION BLOCK

MEANING: Abbreviated code for compartment initialization
CONTEXT: $PK abbreviated code

SAMPLE:

```
$PK
IF (A_0FLG.EQ.1) THEN
 ... compartment initialization block ...
ENDIF
```

DISCUSSION:
A "compartment initialization block" is a block of abbreviated code that sets the initial state of the kinetic system. It is to be executed only when A_0FLG=1. Such a block may be present only in $PK. PREDPP sets A_0FLG to 1 at a call to PK with the first event record of an individual record (if the data are population data), with the first event record of the data set (if the data are single-subject data), and with a reset record.

Special rules apply to such a block.

1) Values may be assigned to reserved variables A_0(n), but only in a compartment initialization block. The value of the amount in the nth compartment (the nth element of the state vector) is set to the value assigned to A_0(n). If PK is called with a dose record or a dose-reset record where the dose is input into the nth compartment, this amount is then increased by the amount of the (bioavailable) dose. If a value is assigned to A_0(n), then it is not necessary that values be assigned to any of the remaining variables A_0(m). A value to the output compartment cannot be assigned. A_INITIAL(n) is a synonym for A_0(n).
2) The statement "`IF (A_0FLG.EQ.1)`" and the corresponding "`ENDIF`" statement may be included explicitly in abbreviated code, thus defining an explicit compartment initialization block. (See example 1 below.)
3) A_0(n) may be assigned a value with an unconditional statement. This defines an implicit compartment initialization block; NMTRAN inserts "`IF (A_0FLG.EQ.1)` ..." before the statement and "`ENDIF`" after it. (See example 2 below.)
   Indicator variables may be included in the unconditional statement. (See example 3 below.)
4) An IF statement testing ICALL and A_0FLG together is not permitted. Instead, two separate nested IF statements must be used: an IF statement testing ICALL must occur as the outermost statement, and an IF statement testing A_0FLG must occur as the innermost statement. The latter may be supplied by NM-TRAN as a result of using an implicit initialization block. (See example 4 below.)
5) Within an explicit compartment initialization block, A_0(n) may be assigned conditionally. The usual rules apply if A_0(n) is a random variable. E.g., A_0(n) cannot be assigned within a nested IF, and it defaults to 0 if it is assigned conditionally but incompletely. However, in checking for a nested IF, tests of A_0FLG and of ICALL are ignored. (See example 3 below.)
6) User-defined variables may be defined in compartment initialization blocks, but not reserved variables such as basic or additional PK parameters.

EXAMPLES OF USAGE:

The following two fragments of code yield identical results:

(1) Explicit compartment initialization block:

```
IF (A_0FLG.EQ.1) THEN
 A_0(1)=THETA(1)*(1+ETA(1))
ENDIF
```

(2) Implicit compartment initialization block:

```
A_0(1)=THETA(1)*(1+ETA(1))
```

(3) This is an example of conditional assignment of A_0(n) (X is a data item or user-defined variable):

```
IF (A_0FLG.EQ.1) THEN
    IF (X.EQ.1) THEN
    A_0(1)=THETA(1)*(1+ETA(1))
    ELSE
    A_0(1)=THETA(2)*(1+ETA(2))
    ENDIF
ENDIF
```

Note that this can be expressed unconditionally with the use of an indicator variable. E.g., if X is a 0/1 variable, then the above is equivalent to

```
    A_0(1)=X*THETA(1)*(1+ETA(1))+(1-X)*A_0(1)=THETA(2)*(1+ETA(2))
```

(4) Suppose compartment initialization should occur only during the Simulation step. The following is not permitted:

```
IF (ICALL.EQ.4.AND.A_0FLG.EQ.1) A_0(1)=THETA(1)*(1+ETA(1))
```

Instead, use:

```
IF (ICALL.EQ.4) THEN
  IF (A_0FLG.EQ.1) A_0(1)=THETA(1)*(1+ETA(1))
ENDIF
```

or simply

```
IF (ICALL.EQ.4) THEN
   A_0(1)=THETA(1)*(1+ETA(1))
ENDIF
```

or even more simply

```
IF (ICALL.EQ.4) A_0(1)=THETA(1)*(1+ETA(1))
```

(See **Compartment Initialization: A_0**)
(See **Compartment Initialization: A_0FLG**)
(See **pk**).

REFERENCES: None.

COMPARTMENT INITIALIZATION: A 0

MEANING: PREDPP-PK global variables
CONTEXT: PK routine

USAGE:

```
USE PRMOD_REAL, ONLY: A_0,DA_0,D2A_0
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: PC,LVR,DPSIZE
REAL(KIND=DPSIZE) :: A_0(PC),DA_0(PC,LVR),D2A_0(PC,LVR,LVR)
```

DISCUSSION:

PREDPP sets A_0FLG to 1 (See **compartment initialization: a_0flg**) at a call to PK with the first event record of an individual record (if the data are population data), with the first event record of the data set (if the data are single-subject data), and with a reset record. At such times, the amounts in the various compartments can be set by the PK routine. It can do this by storing the initial values for the state vector and its partials in A_0,DA_0,D2A_0. The amount in the output compartment can not be set.

`A_0`
: `A_0(n)` = the amount for compartment n

`DA_0`
: `DA_0(n,i)` = the derivative of `A_0(n)` wrt eta(i)

`D2A_0`
: `D2A_0(n,i,j)` = the second derivative of `A_0(n)` wrt eta(i), eta(j) (lower-triangular; j=1, ..., i)

There is a one-to-one correspondence between A_0,DA_0,D2A_0 and A,DAETA,D2AETA (See **State Vector: A**).

NM-TRAN includes A_0,DA_0,D2A_0 in the PK routine when the $PK block includes references to variables A_0FLG, A_0, or A_INITIAL, or when verbatim code is present.

(See **Compartment Initialization: A_0FLG**, **State Vector: A**).

Location prior to NONMEM 7: prdpk3

REFERENCES: None.

COMPARTMENT INITIALIZATION: A 0FLG

MEANING: PREDPP read-only global variables
CONTEXT: User-supplied PK routine

USAGE:
```
USE PROCM_INT, ONLY: A_0FLG
```

GLOBAL DECLARATION:
```
INTEGER(KIND=ISIZE) :: A_0FLG
```

DISCUSSION:

`A_0FLG`

When ICALL>=2, at a call to PK with the first record of of an individual record or with a reset record, PREDPP sets A_0FLAG=1. At such a call, the state vector A and its first and second eta partials (DAETA and D2AETA) have been set to zero. A_0FLG=0 at all other calls to PK.

When A_0FLG=1, PK may initialize compartments. It can do this by storing the initial values for the state vector and its partials in A_0,DA_0,D2A_0.

NM-TRAN includes A_0FLG in the PK routine when the $PK block includes references to variables A_0FLG, A_0(n), or A_INITIAL(n), or when verbatim code is present.

(See **Compartment Initialization: A_0**, **State Vector: A**).

Location prior to NONMEM 7: procmc

REFERENCES: None.

COMPARTMENT UPDATE BLOCK (NM75)

MEANING: Abbreviated code for compartment update
CONTEXT: $PK abbreviated code

SAMPLE:

```
$PK
IF (A_UFLG.EQ.1) THEN
 ... compartment update block ...
ENDIF
```

DISCUSSION:
With NONMEM 7.5, a compartment update block is a block of abbreviated code that is very similar to a compartment initialization block.

In a compartment initialization block, PREDPP sets A_0FLG to 1 at a call to PK with all the compartments at their initial state so that values may be assigned to reserved variables A_0(n).
(See **Compartment Initialization: A_0**)

In a compartment update block, the user sets A_UFLG to 1 in PK to indicate to PREDPP that PK is going to update the compartments. The desired compartment values may be set in the array A_U(n). The user should use MTIME to designate a variable time position at which an abrupt change in compartment amounts occurs. One could input a dose as follows:

```
MTIME(1)=wtime
MTDIFF=1
AZTEST=A_0FLG
IF(TSTATE==MTIME(1).AND.AZTEST==0) A_UFLG=1
IF(A_UFLG==1) THEN
A_U(1)=A(1)+wdose
A_U(2)=A(2)
A_U(3)=A(3)
ENDIF
```

With the Compartment Update Block, the user sets A_UFLG to 1 when the compartments are to be updated. The A_UFLG event must be triggered with an IF(TSTATE==MTIME()) condition as indicated in the above example. Values may be assigned to reserved variables A_U(n). The value of the amount in the nth compartment (the nth element of the state vector) is set to the value assigned to A_U(n). Any A_U(x) not explicitly defined are set to 0. An un-assigned A_U(k) should retain its value, A_u(k)=A_u(k).

The code "IF(A_UFLG==1)...THEN...ENDIF" is optional, as NMTRAN will insert it if not present. A_0FLG must be 0 whenever A_UFLG is set to 1, as shown in the example above (..\examples_uflg.ctl).

The rules for compartment update blocks are similar to those for compartent initialization blocks.

PREDPP expects to find the A_U values in the A_0 arrays. NMTRAN converts A_U() in abbreviated code to A_0() during FSUBS code construction.

(See Guide Introduction_7, "Updating Amounts in Compartments at any Time: The A_UFLG Flag (NM75)"

REFERENCES: Guide Introduction_7

CONSTRAINT

MEANING: CONSTRAINT subroutine
CONTEXT: NONMEM utility routine

DISCUSSION:
CONSTRAINT is a NONMEM utility routine.

Additional algorithmic constraints may be imposed upon model parameters by use of the subroutine CONSTRAINT. This feature is available only for the EM and Bayesian algorithms.

The default CONSTRAINT.f90 is identical to source\CONSTRAINT.f90.

Option CONSTRAIN of the $ESTIMATION record and the $ANNEAL record may be used to give information to the subroutine.

It may be modified by the user.

See **INTRODUCTION TO NONMEM 7, $ANNEAL to facilitate EM search methods**
See **INTRODUCTION TO NONMEM 7, Imposing Thetas, Omegas, and Sigmas by Algebraic Relationships: Simulated Annealing Example**

REFERENCES: Guide Introduction_7

CONT DATA ITEM

MEANING: Continuation (CONT) data item for PREDPP
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... CONT ...`

DISCUSSION:
`CONT` labels PREDPP's continuation (CONT) data item. The continuation data item is optional. The CONT data item allows a PREDPP event record to span several data records. It is useful when more than NONMEM's maximum number of data items per record are needed to describe a single event. When the CONT data item is not defined on the $INPUT record, every event record consists of a single data record.

Values are:

0 This is the last or only data record of the event record.

1 This and the succeeding record are both members of the same event record. MDV must be 1.

PREDPP ignores records having CONT=1, except to pass them to the PK and ERROR routines as part of the complete event record. Values of PREDPP data items such as EVID, AMT, etc., are significant only on the final record of each event record, i.e., on records having CONT=0.

With NM-TRAN abbreviated code, values on data records having CONT=1 are not available for use as right-hand quantities, i.e., abbreviated code cannot reference the values in data records other than the last of each event record. (It would be possible to reference such values using verbatim code, however). When NM-TRAN performs time translation or ii conversion, it does so only for the event records having CONT=0. (See **date**, **time**, **ii**).

REFERENCES: Guide VI, section II , V.A , V.I
REFERENCES: Guide IV, section V.B , V.C.5

CONTR

MEANING: CONTR subroutine
CONTEXT: User-supplied subroutine; replaces a NONMEM dummy routine

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of NONMEM code and previous examples that may be available from advanced users.

USAGE:

```
SUBROUTINE CONTR (I,CNT,IER1,IER2)
USE SIZES, ONLY: ISIZE,DPSIZE
INTEGER(KIND=ISIZE), INTENT(IN OUT) :: I,IER1,IER2
REAL(KIND=DPSIZE),   INTENT(IN OUT) :: CNT
```

DISCUSSION:

CONTR is a user-supplied routine for computing the contribution made to the objective function from an L1 record. It is used to override the NONMEM default objective function.

A user-supplied CONTR routine may be used when the dimension of OMEGA is zero, i.e., when there are no etas in the problem, and in other situations, e.g., with categorical data.

When NM-TRAN is used, the $CONTR record may be used to request that information from the data records be made available to CONTR.

Input argument:

`I` Similar to ICALL for PRED subroutine.

Possible values: 0, 1, 2

Output argument:

`CNT`
Contribution to -2log likelihood for data from the individual record.

`IER1`
0 - Normal return.

non-zero - error return.

`IER2`
0 - error-recovery is to be implemented when IER1 is nonzero.

1 - NONMEM is to stop when IER1 is nonzero.

Other Inputs:

Other inputs are available to CONTR in NONMEM read-only global variables.
In particular:

THETA (current theta) (See **CONTR_MIX:_THETA**)

DV and data values for this L1 record.
(See **CONTR:_Y_DATA_NOBS**, **CONTR:_III_DIM**)
(See **$CONTR**)

Predictions and derivatives. (See **CONTR:_F_G_H**)

NONMEM Utility Routines:

Other NONMEM subroutines may be called by CONTR, depending on the type of data, as follows.

	POPULATION	SINGLE-SUBJECT
CONTINUOUS	ELS, NCONTR (note: same result)	ELS
CATEGORICAL	NCONTR	none

Scatterplots

If the scatterplot step is implemented, and zero lines are appropriate for values of RES and/or WRES, CONTR should request that NONMEM generate such lines. (NONMEM does this by default when a user supplied CONTR is not supplied.) To request zero lines for RES and WRES, CONTR should set OPSCRS(2) and OPSCRS(3) (respectively) to 1.

e.g.,

```
USE CMNM6_INT, ONLY: OPSCRS=>ICONTRSC
 ...
IF (ICALL.LE.1) THEN
OPSCRS(2)=1
OPSCRS(3)=1
ENDIF
```

REFERENCES: Guide I, section G.3
REFERENCES: Guide IV, section III.B.4
REFERENCES: Guide V, section 12.4.16

CONTR: F,G,H

MEANING: NONMEM read-only global variables

CONTEXT: User-supplied routines

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of NONMEM code and previous examples that may be available from advanced users.

USAGE:

```
USE ROCM_REAL, ONLY: F=>FL1,G=>GL1,G2=>GGL1,H=>HL1
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: NO,LVR,LVR2,DPSIZE
REAL(KIND=DPSIZE) :: FL1(NO),GL1(NO,LVR), &
                     GGL1(NO,LVR2*(LVR2+1)/2), &
                     HL1(NO,LVR*LVR/4+LVR/2)
```

DISCUSSION:

These variables changes values with each individual record. They may be used by CONTR.

`F` F(n) = value returned in F from PRED for the nth observation of the individual record.

`G` G(n,i) = Partial derivative of F(n) with respect to eta(i)

`G2` G2(n,i*(i-1)/2+j) = second partial derivative of F(n) with respect to eta(i) and eta(j) (j<=i)
G2 is arranged in symmetric storage, e.g.

1
2 3
4 5 6

G2(n,1) = 2nd. partial of F(n) wrt. eta(1) eta(1)
G2(n,2) = 2nd. partial of F(n) wrt. eta(2) eta(1)
G2(n,3) = 2nd. partial of F(n) wrt. eta(2) eta(2)
G2(n,4) = 2nd. partial of F(n) wrt. eta(3) eta(1)
G2(n,5) = 2nd. partial of F(n) wrt. eta(3) eta(2)
G2(n,6) = 2nd. partial of F(n) wrt. eta(3) eta(3)

`H` H(n,i) = Partial derivative of F(n) with respect to eps(i) for i=1,neps, where neps is the number of epsilons in the problem.
H(n,j*neps+i) = second partial derivative of F(n) with respect to eps(i) and eta(j).

E.g. suppose that there are two epsilons in the problem:

1 3 5
2 4 6

H(n,1) = partial derivative of F(n) wrt. eps(1)
H(n,2) = partial derivative of F(n) wrt. eps(2)
H(n,3) = 2nd. partial of F(n) wrt. eps(1) eta(1)
H(n,4) = 2nd. partial of F(n) wrt. eps(2) eta(1)
H(n,5) = 2nd. partial of F(n) wrt. eps(1) eta(2)
H(n,6) = 2nd. partial of F(n) wrt. eps(2) eta(2)

Location prior to NONMEM 7: rocm3
REFERENCES: None.

CONTR: III,DIM

MEANING: NONMEM read-only global variables

CONTEXT: User-supplied routines

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of NONMEM code and previous examples that may be available from advanced users.

USAGE:

```
USE ROCM_INT, ONLY:  III=>NL2_RECS, DIM=>L2_DIM
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: NO
INTEGER(KIND=ISIZE) :: NL2_RECS,L2_DIM(NO)
```

DISCUSSION:

These variables change values with each individual record. They may be used by CONTR.

`III`
: Number of L2 records in the individual record.

`DIM`
: Length of each L2 record.

When there is no L2 data item, III = number of observations in the individual record, and all values in DIM = 1.

Location prior to NONMEM 7: rocm2

REFERENCES: None.

CONTR: KCALL

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of NONMEM code and previous examples that may be available from advanced users.

USAGE:

```
USE ROCM_INT, ONLY: KCALL=>K_CONTR
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: K_CONTR
```

DISCUSSION:

KCALL
: KCALL=1 with a unique pass through data calling PRED and CONTR with final parameter estimates.
KCALL=0 otherwise.

Location prior to NONMEM 7: rocm13

REFERENCES: None.

CONTR: Y DATA NOBS

MEANING: NONMEM read-only global variables
CONTEXT: CONTR routine

USAGE:

```
USE ROCM_REAL, ONLY: Y=>DV_ITM,DATA=>RDATA
USE ROCM_INT,  ONLY: NOBS=>NOBSIND2
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: NO,DPSIZE
REAL(KIND=DPSIZE) :: DV_ITM(NO),RDATA(NO,3)
INTEGER(KIND=ISIZE) :: NOBSIND2
```

DISCUSSION:

These variables change values with each individual record. They may be used by a user-written CONTR subroutine.

`Y(k)`
: DV data item on the kth data record of the individual record, ignoring data records with MDV=1.

`DATA(k,i)`
: The value of the ith type of data item specified in NM-TRAN's $CONTR record, appearing on the kth observation record of the individual record.

`NOBS`
: Number of observations in the individual record.

DATA is used also in MIX (See **data_for_mix**).

Location prior to NONMEM 7: rocm1

REFERENCES: Guide IV, section III.B.4

COPYING BLOCK

MEANING: Abbreviated code especially for copying pass
CONTEXT: $PRED, $PK, $ERROR, $AES, $DES abbreviated code

SAMPLE:

```
IF (COMACT.EQ.1) TVCL=CL
```

DISCUSSION:

Values of variables displayed in tables and scatterplots are obtained (i.e. copied) from module NMPRD4. There are particular times when data records are passed to PRED for the purpose of obtaining these values; these are called copying passes. The variable COMACT signals that a copying pass is in progress when its value is positive. There may be a number of copying passes, but if (and only if) values of variables are to be displayed, there is at least one copying pass. With the first copying pass, the value of COMACT is 1 and that of MIXNUM is 1. If a mixture model with k subpopulations is used, then COMACT remains 1 during a total of k copying passes, and with each copying pass the value of MIXNUM increments by 1. If conditional estimation is used (or if the POSTHOC option appears), then there follows a set of copying passes where the value of COMACT is 2 and again, MIXNUM increases from 1 to k. If a mixture model is not used, there are altogether at most two copying passes, one with COMACT=1 and another with COMACT=2.

If values of a variable from earlier copying passes are needed in later passes, the values for the variable should be stored in the SAVE region of module NMPRD4. (See **comsav**). When the values are stored in the SAVE region, that value computed with a given data record during a copying pass will be found in NMPRD4 when the same record is passed during the next copying pass, i.e. it will have been saved from the previous copying pass. This is in contrast to the usual behaviour (with a noncopying pass), where with a given data record, the value in NMPRD4 is the value computed with the previous data record. The value is 0 when a record is passed during the first copying pass, and though PRED may set it or reset it during a subsequent copying pass, this need not be done (see discussion below about COMACT).

The values used in tables and scatterplots (whether or not these values are stored in the SAVE region) are those copied from NMPRD4 with the last copying pass.

A copying block is a block of abbreviated code that is only executed during a copying pass, i.e. when COMACT has a positive value. Special rules apply, which allow the user to be less concerned about using the COMSAV option of the $ABBREVIATED record (see example below):

`COM(i)` variables that are defined in a copying block are referred to as explicit SAVE variables. PRED-defined variables that are defined in a copying block, other than `COM(i)` variables, are referred to as implicit SAVE variables. Collectively, the two types of SAVE variables are referred to as SAVE variables. NM-TRAN stores SAVE variables in the SAVE region of module NMPRD4.

Implicit and explicit SAVE variables cannot both appear in abbreviated code. The COMRES option of the $ABBREV record cannot be used when any implicit SAVE variables are used, but it must be used when explicit SAVE variables appear (as whenever `COM(i)` variables appear).

The size of the SAVE region of NMPRD4 depends on the COMSAV option of the $ABBREV record. This option may be used in three ways:

No COMSAV option. The SAVE region of module NMPRD4 will nonetheless include all the SAVE variables.

`COMSAV=n` (n>=0). NM-TRAN will, if necessary, extend the size of the SAVE region from n to a larger value so that all SAVE variables will be included in the region.

`COMSAV=-1`. There is to be no SAVE region. Variables defined in a copying block will not be SAVE variables.

EXAMPLE OF USAGE:

```
$ERROR
Y=F+F*EPS(1)
IPRED=F
IF (COMACT.EQ.1) FT=F
WR=(DV-IPRED)/FT

$EST ... POSTHOC
$TABLE FT IPRED WR
```

With the first copying pass the value of COMACT is 1, which signals that during this copying pass, all ETA variables are set to 0. Since the option POSTHOC appears, with a subsequent copying pass the value of COMACT is 2, which signals that during this copying pass, all ETA variables are set to their conditional estimates.

In this example, WR is set to the weighted intra-individual residual. When COMACT=1, the prediction for the typical individual (ETA=0) is computed and stored in FT. FT is a SAVE variable, so this value will have been saved, and when COMACT=2, this same value will be found in FT. With COMACT=2, the weight used with the residual is computed to be the reciprocal of this FT value, while IPRED is computed from conditional estimates of the ETA variables and thus its value applies to to the individual with these estimates, rather than to the typical individual. The values of IPRED and WR appearing in the table are those obtained during the last copying pass. The tabled value of IPRED is based on the conditional estimate of ETA. The value of FT is also the value obtained during the last copying pass, but it in turn is also the value obtained from the first copying pass, as the value of FT has not changed during any subsequent copying pass, and this value is based on ETA=0.

Since FT is a SAVE variable, a SAVE region will have been allocated where this variable will be stored, and (unless there is some reason to use the COMSAV option other than to insure this) the user not be concerned about this option.

(See **COMACT,COMSAV**)
(See **PRED-Defined Variables**).
(See **abbreviated**).

REFERENCES: None.

CORRELATION ACROSS L2 RECORDS

MEANING: NONMEM-PRED global variables
CONTEXT: PRED and ERROR routines

USAGE:
`CORRL2(n,m)=...`

DISCUSSION:

An individual record is divided into L2 records. An L2 record may contain one or more observations (on one or more separate data records respectively), in which case it is called an observation-L2 record. The values of epsilons used in the intraindividual model may be correlated across the observations contained in the L2 record, and thus the L2 record may define a multivariate observation - the L2 observation. (When all L2 observations in the data set are univariate, L2 data items need not appear, and when L2 data items do not appear, NONMEM assumes that each data record is a distinct L2 record.)

By default, the values of a given epsilon are statistically independent across L2 observations within an individual record. Using CORRL2, however, these values may be correlated. More precisely, the values of the epsilons associated with the mth diagonal block of SIGMA may be correlated across L2 observations, and it will be understood that for two different epsilons (eps1 and eps2, say) associated with the mth block, the correlation between the values of eps1 for L2 observations A and B will be taken to be the same as the correlation between the values of eps2 for these same two L2 observations.

With NONMEM 7.3, reserved variable CORRL2 is used and code such as the following may be used in $ERROR and $PRED blocks.

Proceed as follows. With the first data record of the nth observation-L2 record, and with respect to the values of the epsilons associated with the mth diagonal block of SIGMA, the PRED routine should set CORRL2(k,m), for k=1,...,n, to the correlation between the values for the kth L2 observation and the nth L2 observation. (CORRL2(n,m) should be set to 1.0; in particular, with the first data record of the 1st observation-L2 record, CORRL2(1,m) should be set to 1.0.)

E.g. Suppose that the L2 observations are bivariate and chronologically ordered by a TIME data item, and suppose that the intraindividual model has two epsilons (one for each element of the bivariate L2 observation), each associated with the same diagonal block of SIGMA, Then the values of these epsilons may be autocorrelated across the L2 observations, as specified by the above code. But if the two epsilons are associated with two different diagonal blocks, then one might use this code, in which each sigma block has its own autocorrelation parameter theta(4) or theta(5):

```
$ABBREVIATED DECLARE T1(NO)
$ABBREVIATED DECLARE INTEGER I1
$ABBREVIATED DECLARE DOWHILE J1

IF (NEWL2==1.AND.EVID==0) THEN
  I1=I1+1
  T1(I1)=TIME
  J1=1
  DO WHILE (J1<=I1)
  CORRL2(J1,1)=EXP(-THETA(4)*(TIME-T1(J1))) ; Sigma block 1
  CORRL2(J1,2)=EXP(-THETA(5)*(TIME-T1(J1))) ; Sigma block 2
```

```
  J1=J1+1
  ENDDO
ENDIF


$THETA 2  ;[CL]
$THETA 30 ;[V]
$THETA 0.05 ;[Rho1]
$THETA 0.075 ;[Rho2]
```

If you wish to have more control as to when the CORRL2 is used to model among the individual data records within the L2 records, consider the following example:

```
$ABBR DECLARE T1(NO),
$ABBR DECLARE INTEGER I1, DOWHILE J1

$ERROR
IF (NEWIND.NE.2) I1=0


IF (NEWL2==1) THEN
  I1=I1+1
  T1(I1)=TIME
  J1=1
  DO WHILE (J1<=I1)
  IF(MV1==0.0) CORRL2(J1,1)=EXP(-THETA(4)*(TIME-T1(J1)))
  IF(MV2==0.0) CORRL2(J1,2)=EXP(-THETA(5)*(TIME-T1(J1)))
  J1=J1+1
  ENDDO
ENDIF
IF(CMT==1) Y=F+F*EPS(1)+EPS(2)
IF(CMT==2) Y=F+F*EPS(3)+EPS(4)
 ...
$SIGMA BLOCK(2)
0.3
0.001 0.04
$SIGMA BLOCK(2)
0.7
0.001 0.08
```

Here MV1 and MV2 data item that is assumed to exist in the data set, and MV signals the desire to use CORRL2 on an L2 observation within an L2 record, *on the first data record of the L2 record*. The following is a section of data to indicate more clearly how this may be set up:

```
ID   TIME L2 MV1 MV2 CMT DV
1    2.0  1  0   0   1   0.8
1    2.0  1  0   0   2   10.0
1    4.0  2  0   0   1   0.6
1    4.0  2  0   0   2   12.0
1    6.0  3  1   1   1   0.3
1    6.0  3  1   1   2   20.0
1   10.0  4  1   0   1   0.1
```

```
1   10.0  4  1   0   2   15.0
1   14.0  5  1   1   1   0.03
```

Only data points with times 2, 4, and 10 hours will incorporate the correlation of the CORRL2 model. Furthermore the record of time 10 hours has MV1=1 and MV2=0, so the observation record in the 10 hour L2 record that uses Sigma block 2 will have a CORRL2 assessment, whereas the observtation in the 10 hour L2 record that uses Sigma block 1 will not be CORRL2 modeled. In the above example, Sigma block 1 consists of EPS(1) and EPS(2), and Sigma block 2 consists of EPS(3) and EPS(4), with the CMT value determining which data records use which Sigma blocks (epsilons sets) based on how Y is evaluated.

RESTRICTIONS:

With versions of NONMEM before 7.3, C should be used rather than CORRL2. Because C is not recognized by NM-TRAN, and because of other restrictions regarding abbreviated code, a specification of C, as above, within a block of abbreviated code, must be done using verbatim code. See help for that version of NONMEM.

SIMULATION WITH POPULATION DATA AND AUTO-CORRELATION

If population data are simulated, the correlations must be stored in CORRL2 before the NONMEM utility routine SIMEPS is called. With NONMEM 7.3, code such as the following may be used in $ERROR and $PRED blocks. Because the value of CORRL2 is set before SIMEPS is called, NM-TRAN omits the usual default call to SIMEPS.

```
$ABBR DECLARE T(NO), INTEGER I, DOWHILE J

IF (ICALL == 4) THEN
IF (NEWIND /= 2) I=0
IF (MDV == 0) THEN
   I=I+1
   T(I)=TIME
   J=1
   DO WHILE (J <= I)
   CORRL2(J,1)=EXP(-THETA(4)*(TIME-T(J)))
   J=J+1
   ENDDO
ENDIF
CALL SIMEPS(EPS)
ENDIF
```

RESTRICTIONS:

With versions of NONMEM before 7.3, C should be used rather than CORRL2. Because C is not recognized by NM-TRAN, and because of other restrictions regarding abbreviated code, a specification of C, as above, within a block of abbreviated code, must be done using verbatim code. Since NM-TRAN generated code has a call to SIMEPS in its second section (see Guide IV), this means that correlations must be computed and stored using verbatim code in the "FIRST block. (Details can be supplied on request.)

SINGLE-SUBJECT DATA

"Single-subject data" with correlated residual error, can be simulated and analyzed. To do this, though, a technique is needed which can always be used with such data: the data are handled as data from a population sample with a single individual, and OMEGA is constrained to be 0.

(See **Simulation:_SIMEPS_Error_Code**)
Location prior to NONMEM 7: nmprd5
REFERENCES: None.

CORRELATION MATRIX OF ESTIMATE

MEANING: Part of NONMEM's estimate of the precision of its parameter estimates
CONTEXT: NONMEM output

DISCUSSION:
Asymptotic statistical theory applied to extended least-squares estimation (as used in NONMEM) says that the distribution of the parameter estimators is multivariate normal, with variance-covariance matrix that can be estimated from the data. NONMEM supplies such an estimate of the variance-covariance matrix of the parameter estimates (See **covariance matrix of estimate**). The correlation matrix is the variance-covariance matrix in correlation form. If the correlation between two parameters is large (e.g., >.95), then one may conclude that a considerable portion of the uncertainty in each parameter is due to the inability of the data to distinguish between the two. The problem can be avoided by getting additional data or by using a simpler (fewer parameters) model.

```
********** CORRELATION MATRIX OF ESTIMATE **********
            TH 1      TH 2      OM11      OM12      OM22      SG11
 TH 1        1.01E+00
 TH 2       -2.85E-01  2.55E-02
 OM11        8.87E-01 -6.77E-01  1.67E+01
 OM12       ......... ......... ......... .........
 OM22        2.81E-01 -9.96E-01  6.72E-01 .........  1.31E-03
 SG11       -4.83E-01  3.10E-01 -5.44E-01 ......... -2.67E-01  1.60E-01
```

The matrix (which is symmetric) is given in lower triangular form. In this example, the 2x2 matrix, OMEGA, was constrained to be diagonal; the omitted entries above (.....) indicate that omega(2,1) is not estimated, and consequently has no corresponding row/column in the correlation matrix of the estimates.

With NONMEM 7.2 and higher, the diagonal elements are equal to the square root of the diagonal elements of the covariance matrix (standard error); with previous versions they were 1.0.

When the size of the array exceeds 75x75, a compressed form is printed in which the omitted entries (.....) are not printed. The compressed form may also be requested for arrays smaller than 75x75 (See **$covariance**).

REFERENCES: Guide I, section C.3.5.2
REFERENCES: Guide V, section 5.4

COVARIANCE MATRIX OF ESTIMATE

MEANING: NONMEM's estimate of the precision of its parameter estimates
CONTEXT: NONMEM output

DISCUSSION:
From asymptotic statistical theory, the distribution of the parameter estimates is multivariate normal, with a variance-covariance matrix that can be estimated from the data. Such an estimate forms the basic output of NONMEM's Covariance Step. The variance-covariance matrix is not to be confused with either SIGMA, the covariance matrix for the second level random effects, or with OMEGA, the covariance matrix for the first level random effects. These two matrices describe the variability of epsilons or etas, respectively, about their means. The variance-covariance matrix of the (distribution of) parameter estimates, on the other hand, describes the variability under the assumed model of the parameter estimates across (imagined) replicated data sets, using the design of the real data set. The following is an example of the NONMEM output giving the estimate of the variance-covariance matrix.

```
********** COVARIANCE MATRIX OF ESTIMATE  **********
            TH 1      TH 2      OM11      OM12      OM22      SG11
 TH 1        1.02E+00
 TH 2       -7.34E-03  6.50E-04
 OM11        1.50E+01 -2.88E-01  2.78E+02
 OM12       ......... ......... ......... .........
 OM22        3.73E-04 -3.33E-05  1.47E-02 .........  1.72E-06
 SG11       -7.79E-02  1.26E-03 -1.45E+00 ......... -5.59E-05  2.55E-02
```

The matrix (which is symmetric) is given in lower triangular form. In this example, the 2x2 matrix, OMEGA, was constrained to be diagonal; the omitted entries above (.....) indicate that OM12 is not estimated, and consequently has no corresponding row/column in the variance-covariance matrix. When the size of the array exceeds 75x75, a compressed form is printed in which the omitted entries (.....) are not printed. The compressed form may also be requested for arrays smaller than 75x75 (See **$covariance**).

The (estimated) variance-covariance matrix is computed from the R and S matrices; it is Rinv*S*Rinv, where Rinv is the inverse of the R matrix. The R matrix is the Hessian matrix of the objective function, evaluated at the parameter estimates. The S matrix is obtained by summing the cross-product gradient vectors of the individual-based objective functions, evaluated at the parameter estimates. The individual-based objective functions are the separate terms contributed by each individual's data to the overall objective function, and the cross-product gradient vectors are summed across the individuals in the data set.

The inverse variance-covariance matrix R*Sinv*R is also output (labeled as the Inverse Covariance Matrix), where Sinv is the inverse of the S matrix. If S is judged to be singular, a pseudo-inverse of S is used, and since a pseudo-inverse is not unique, the inverse variance-covariance matrix is really not unique. In either case, the inverse variance-covariance matrix can be used to develop a joint confidence region for the complete set of population parameters. As we usually develop a confidence region for a very limited set of population parameters, this use of the inverse variance-covariance matrix is somewhat limited.

An error message from the Covariance Step stating that the R matrix is not positive semidefinite suggests that the parameter estimate does not correspond to a true (local) minimum and is not to be trusted. (It may be a saddle point.) An error message stating that the R matrix is positive semidefinite, but singular, indicates that the objective function is flat in a neighborhood of the parameter estimate, and so the minimum is not really unique, and there is probably some overparametrization. With both error messages, neither a variance-covariance matrix nor inverse variance-covariance matrix is output. An error message stating that the S matrix is singular indicates strong overparameterization. However, provided the R matrix is judged to be positive semidefnite and nonsingular (i.e. positive definite), both the variance-covariance and inverse variance-covariance matrices are output.

When the R matrix is judged to be singular, but positive semidefinite, then the T matrix, R*Sinv*R, where Sinv is the inverse (or a pseudo-inverse) of the S matrix, is output. This cannot be called the inverse covariance matrix, as the covariance matrix does not exist. However, as with the inverse variance-covariance matrix, T can be used to develop a joint confidence region for the complete set of population parameters.

There are options that allow the variance-covariance matrix to be computed as either 2*Rinv or 4*Sinv. Asymptotic statistical theory suggests that these matrices are appropriate under the additional assumption that the objective function is indeed additively proportional to minus twice the log likelihood function for the data.

Unless the reported number of significant digits in the final parameter estimate is at least as large as the requested number of significant digits, the Covariance Step will not be implemented. (See **sig digits**). Sometimes, the number of significant digits is not reportable. However, when it is and the user thinks this number to be adequate, and a model specification file was output (See **model specification file**), NONMEM may be run again where the Covariance Step is implemented, while the Estimation step is is not repeated (i.e. the `MAXEVAL` option is set to 0). With the subsequent run, the model specification file should be input, and the requested number of significant digits should be set to a value less than the reported number of significant digits from the first run (presumably, this value would be the reported number rounded down to the highest integral value).

(See **standard error**, **correlation matrix of estimate**).

REFERENCES: Guide I, section C.3.5.2
REFERENCES: Guide II, section D.2.5
REFERENCES: Guide V, section 5.4 , 13.3

CRIT

MEANING: CRIT subroutine
CONTEXT: User-supplied subroutine; replaces a NONMEM routine

USAGE:

```
SUBROUTINE CRIT (ICALL,J,N,WRES,V)
USE SIZES, ONLY: ISIZE,DPSIZE
INTEGER(KIND=ISIZE), INTENT(IN)      :: J,N
INTEGER(KIND=ISIZE), INTENT(IN)      :: ICALL
REAL(KIND=DPSIZE),   INTENT(OUT)     :: V
REAL(KIND=DPSIZE),   INTENT(IN)      :: WRES(*)
END SUBROUTINE CRIT
```

DISCUSSION:
The CRIT subroutine is used to modify the NONMEM objective function used with the First-Order (or the Extended Least Squares) Estimation method. This objective function may be regarded as being the sum of contributions computed from each individual record. The first term in the contribution from an individual record is independent of data, but the second term in the contribution is the sum of the squared weighted residuals for the data in the record. The weights are functions of the model parameters and are obtained so that with a given set of parameter values, assumed to be the true parameter values, each weighted residual has unit variance and all the weighted residuals are mutually uncorrelated. With the CRIT routine, a function of the weighted residuals other than the sum of their squares may be substituted for the second term. The function may vary between individual records. The CRIT routine is called by NONMEM with one individual record after another (individual records without observations are skipped).

Input argument:

`ICALL`
: Similar to ICALL for PRED subroutine.
0 - First call to CRIT in the run
1 - First call to CRIT in the current problem
2 - Computation of function value required

`J` Number of individual record

`N` Number of observations in the individual record

`WRES`
: Vector of weighted residuals (as many residuals as there are observations)

Output argument:

`V` Value of function

Note: When a user CRIT is supplied, NONMEM subroutine CELS may not be called.

REFERENCES: Guide II, section C.6 , Figure 8

REFERENCES: Guide III, section V.5.0
REFERENCES: Guide IV, section III.B.6

CTLO CTUP PROBABILITY: PR CT

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

USAGE:

```
USE ROCM_REAL, ONLY PR_CT
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: DPSIZE
REAL(KIND=DPSIZE) :: PR_CT
```

DISCUSSION:

When with a given data record, either of the limits CTLO or CTUP are set, thus defining an interval of values comprising one of several categories equated with the possible values of a potential observation, then during a copying pass (and during ICALL=5 and 6), PR_CT is the estimated probability that an observation will be of the category in question.
(See **CTLO**, **CTUP**)
(See **copying_block**, **expectation**, **data_average**)

If the mean and variance of the intra-individual model for a potential observation are specified, and if the limits CTLO or CTUP are set, a value of PR_CT will be computed, whether the record has MDV=0 or MDV=1. If neither CTLO nor CTUP are set, the value PR_CT will be 1.

`PR_CT` may be used as a right-hand quantity in abbreviated code for $PRED, $PK, $ERROR, and $INFN blocks.

Location prior to NONMEM 7: rocm45

REFERENCES: None.

CTLO

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPR_REAL, ONLY:CTLO=>CTLW,DCTLO=>DCTLW,DDCTLO=>DDCTLW
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: LVR,DPSIZE
REAL(KIND=DPSIZE) :: CTLW,DCTLW(LVR),DDCTLW(LVR,LVR)
```

DISCUSSION:
An observation may be the event that the value of a normally distributed variable falls in a given interval. The likelihood for this event may be automatically computed. With the data record containing the observation, one specifies the mean and variance of the variable for NONMEM, as usual, and one sets CTLO to the lower endpoint of the interval. If with population data, this endpoint depends on an eta variable, then the first- and second-derivatives of the endpoint with respect to etas should also be set. Derivatives equal to 0 need not be explicitly set, and the only elements of DDCTLO that need be set are those in the lower triangle. CTLO may be set in a $PRED or $ERROR abbreviated code, and then NM-TRAN automatically sets the derivatives. NONMEM can detect when CTLO is not set. When CTLO is set, the Laplacian estimation method must be used.

CTUP may be used in conjunction with CTLO.
(See **ctup**).

PR_CT is the estimated probability that an observation will be of the category in question.
(See **pr_ct**, **CTLO_CTUP_Probability:_PR_CT**).

Limitation:
May not be used with the LIKELIHOOD or -2LOGLIKELIHOOD options.

Location prior to NONMEM 7: nmpr13

REFERENCES: None.

CTUP

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPR_REAL, ONLY CTUP,DCTUP,DDCTUP
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: DPSIZE,LVR
REAL(KIND=DPSIZE) :: CTUP,DCTUP(LVR),DDCTUP(LVR,LVR)
```

DISCUSSION:
An observation may be the event that the value of a normally distributed variable falls in a given interval. The likelihood for this event may be automatically computed. With the data record containing the observation, one specifies the mean and variance of the variable for NONMEM, as usual, and one sets CTUP to the upper endpoint of the interval. If with population data, this endpoint depends on an eta variable, then the first- and second-derivatives of the endpoint with respect to etas should also be set. Derivatives equal to 0 need not be explicitly set, and the only elements of DDCTUP that need be set are those in the lower triangle. CTUP may be set in a $PRED or $ERROR abbreviated code, and then NM-TRAN automatically sets the derivatives. NONMEM can detect when CTUP is not set. When CTUP is set, the Laplacian estimation method must be used.

CTLO may be used in conjunction with CTUP.
(See **ctlo**).

PR_CT is the estimated probability that an observation will be of the category in question.
(See **pr_ct**, **CTLO_TUP_Probability:_PR_CT**).

Limitation:
May not be used with the LIKELIHOOD or -2LOGLIKELIHOOD options.

Location prior to NONMEM 7: nmpr14

REFERENCES: None.

DATA AVERAGE BLOCK

MEANING: Abbreviated code for computation of raw-data-averages
CONTEXT: Abbreviated code

SAMPLE:

```
 $ERROR
IF (ICALL.EQ.6) THEN
 ... data average block ...
ENDIF
```

DISCUSSION:
A data average block is a block of abbreviated code that is only executed when ICALL=6. This value of ICALL occurs when the raw-data-average (RAW_) data item is defined in the data set and has a non-zero value for some records. Data average blocks are not required when the raw-data-average data item is present, but they allow the user additional functionality. Such blocks may be present in $PRED and $ERROR. If for a given observation record matching a template record, a data average block sets the DV variable to a value different from the one in the record, this value is the one included in the average.

In this example, the displayed DV value for each template record is the proportion of DV items greater than 10 on those records matching the template record.

```
 Y=...
 TRDV=DV
 IF (ICALL.EQ.6) THEN
    DV=0
    IF (TRDV.GT.10) DV=1
 ENDIF
```

PRED may return a value of 1 in F, indicating that the DV item in the record is not to be included in the average. Continuing the above example, suppose that the DV variable is set to a value only when it exceeds 2, so that the average is the proportion of observations exceeding 10 among those that exceed 2. The code is as follows. See note 2 below on the returned value of F.

```
Y=...
TRDV=DV
IF (ICALL.EQ.6) THEN
   IF (TRDV.LE.2) Y=1
   DV=0
   IF (TRDV.GT.10) DV=1
ENDIF
```

When ICALL=6, PRED and ERROR are called with successive records as usual. However, with each observation record, all output from these routines other than a value set for the DV data item and values set for PRED-defined (ERROR-defined) items V used in a table or scatterplot and located in the SAVE region (See **save_region** (and the F value, in so far as it is or is not 1) is ignored. In the same way that an average is formed for the DV, averages are formed for the elements of V. Upon entry into PRED or ERROR with a given record, the value of the DV item is the one on the record, and the values for V are the values that will be used in a table and/or scatterplot. If PRED or ERROR changes one of these values, the new value is the one used in the average. If a value is not changed,

the unchanged value is used in the average. (For a nonobservation record, the output from that record is completely ignored.)

The following series of examples concern taking averages of PRED-defined items. They involve a mixture model, where, under each of the subpopulations, there is a parameter PA whose value depends on an eta. There is a template record.

Example A.

Suppose first that PA does not depend on (interindividually-varying) covariate values. Suppose moreover, that during copying passes, with each individual record a value of the quantity Q=P1*PA1+P2*PA2+P3*PA3 has been computed and stored in the SAVE region. Here, the P's are the mixture probabilities, and the PA's are conditional estimates of PA under the different models for the different subpopulations. Due to the presence of the template record, at ICALL=6, the average of Q across the eta estimates (from individuals with the same covariate values as are contained in the template record) will be computed - without any need for a data average block. More precisely, at ICALL=6 a pass through the data set occurs, during which a value of Q is obtained with each of the observation records that match the template record. However, the average of the Q values is an average of within-individual averages, and since Q does not vary within an individual record, the within-individual average computed for that record is the same value of Q as is obtained with each of the observation records of the individual record (matching the template record). The resulting average of the Q values is an estimate of the expected value of PA over the subpopulations and the randomly-varying PA's.

Example B.

Suppose that PA depends on a covariate X
E.g. PA=THETA(1)*X**THETA(2)*EXP(ETA(1)),
and that one is interested in an estimate of the expected value of PA for X=x. Suppose also that the value for X in the template record is x. Once again, suppose that during copying passes, with each individual record a value of Q has been computed and stored in the SAVE region - using whatever value of X appears in the individual record. At ICALL=6, if *and only if* the observation records within an individual record have the value X=x and match the template record with respect to the other relevant data items, will the within-individual average be included in the average Q. Moreover, these observation records are the very ones whose value for Q is of interest.

Example C.

However, under a well-specifed model, ETA(1) should be independent of X, and so one might want to use an average of Q across the eta estimates from *all* individuals (with observation records). Then one might (i) during copying passes, for each individual record compute Q using the specific value X=x - regardless of what value of X appears in the individual record, and (ii) include the record

```
$OMIT X
```

to prevent the values of X from affecting the match with the template record.

Example D.

The strategy in example C fails if it is necessary to match on X for the purpose of forming averages other than the average Q, and it is at least awkward if one is interested in the expected value of PA for a variety of values of X. Here is an alternative strategy.

```
$ABB COMRES=4 COMSAV=4
$ERROR
 ...
Y=...
IF (COMACT.EQ.2) THEN
   IF (MIXNUM.EQ.1) COM(1)=ETA(1)
   IF (MIXNUM.EQ.2) COM(2)=ETA(2)
   IF (MIXNUM.EQ.3) COM(3)=ETA(3)
ENDIF
IF (ICALL.EQ.6) THEN
   PA1=THETA(1)*TEMPLT(X)**THETA(2)*EXP(COM(1))
   PA2=THETA(1)*TEMPLT(X)**THETA(2)*EXP(COM(2))
   PA3=THETA(1)*TEMPLT(X)**THETA(2)*EXP(COM(3))
   COM(4)=MIXP(1)*PA1+MIXP(2)*PA2+MIXP(3)*PA3
ENDIF

$TABLE COM(1) COM(2) COM(3) NOPRINT FILE=junk
$TABLE ID ... COM(4) ...
```

During the copying passes, the values of the eta estimates are stored (in items COM(1), COM(2) and COM(3) of the SAVE region) rather than the value of Q. The first table record appears because unless the items COM(1), COM(2), and COM(3) are displayed, at ICALL=6 their values are 0. TEMPLT(X) refers generically to the value of X on the template record (see Special Rule 4 below); so there can be numerous template records with different values of X.

Example E.

Examples A-C are not explicit about how the mixture probabilities are computed. They might be obtained via MIXP, in which case during the copying passes (similar to what happens at ICALL=6 with example D), with a given individual record, they are the probabilities pertaining to that record. As long as the mixture probabilities do not depend on interindividual-varying covariates, the mixture probabilities are the same no matter what individual record it is to which they pertain. But if the probabilities depend on interindividual-varying covariates, and especially if one wants to estimate the expected value of Q for numerous different sets of values for these covariates, then one might use:

```
$ERROR
include nonmem_reserved_general
Y=...
IF (COMACT.EQ.2) THEN
   IF (MIXNUM.EQ.1) COM(1)=ETA(1)
   IF (MIXNUM.EQ.2) COM(2)=ETA(2)
   IF (MIXNUM.EQ.3) COM(3)=ETA(3)
ENDIF
IF (ICALL.EQ.6) THEN
   PA1=THETA(1)*TEMPLT(X)**THETA(2)*EXP(COM(1))
   PA2=THETA(1)*TEMPLT(X)**THETA(2)*EXP(COM(2))
   PA3=THETA(1)*TEMPLT(X)**THETA(2)*EXP(COM(3))
   COM(4)=MIXPT(1)*PA1+MIXPT(2)*PA2+MIXPT(3)*PA3
ENDIF
```

The mixture probabilities found in MIXPT pertain to the individual record containing the template record (see Special Rule 7 below). Because they have been computed using the covariate values in that record, they are commensurate with the way PA has been computed for the different subpopulations. With versions prior to NONMEM 7.3, the probabilities in MIXPT must be referenced via verbatim code (note the double quotes).

```
$ERROR
" USE ROCM_REAL MIXPT=>MIXP_RAW
 ...
"  COM(4)=MIXPT(1)*PA1+MIXPT(2)*PA2+MIXPT(3)*PA3
```

Special rules apply to data average blocks and passes through the data with ICALL=6.

1) No eta derivatives are computed in a data average block.
2) A RETURN statement may be used in a data average block. If Y= appears (i.e. Y is assigned a value) in a data average block before the RETURN (not necessarily in the same block containing the RETURN), then F is set to Y (F=Y); otherwise F is set to to 0 (F=0). If there is no RETURN statement in a data average block, then as usual, F is set to the value of Y assigned by the time PRED (ERROR) exits.
3) Loops are permitted in a data average block. The syntax is as follows.

   ```
   DO WHILE (condition)
    .. statements ..
   END DO
   ```

4) During calls with ICALL=6, the template data record is found in NONMEM global variable TEMPLT. Items in the template data record may be referred to in abbreviated code of the data average block by position or by label, e.g., TEMPLT(1) or TEMPLT(ID).
5) During calls with ICALL=6, the repetition feature may not be used.
6) If a mixture model is used, then during calls with ICALL=6, both MIXNUM and MIXEST are the index of the subpopulation into which the individual (whose data record is being passed) has been classified.
7) If a mixture model is being used, then during calls with ICALL=6, (the final estimates of) the mixture probabilities associated with the individual record containing the template record are found in NONMEM global variable MIXPT. Code that uses these probabilities must be verbatim code.

(See **raw_**).
(See **MIX CONTR: TEMPLT**)
(See **Mixture model: MIXPT**).

REFERENCES: None.

DATA ITEMS

MEANING: The elements of a NONMEM Data Record
CONTEXT: NONMEM output

DISCUSSION:
Each data record consists of a series of data items, the labels for which are defined on the $INPUT record. Data items are of three types: NONMEM items, PREDPP items, and others. The first two types are signified by certain reserved labels, such as (for NONMEM items): `ID` (for IDentification), `MDV` (for Missing Dependent Variable), or `DV` (for Dependent Variable). NONMEM prints the labels of the data items in the order it understands them to appear on the data records as a check for the user.

```
LABELS TO BE USED FOR ITEMS APPEARING IN TABLES AND SCATTERPLOTS ARE:
   ID     TIME     DV     AGE       WT     PRED     RES      WRES
```

The above states that the first data item is ID (NONMEM item), the second is TIME (a PREDPP item), the third is DV (a NONMEM item), then AGE and WT, which are other items, and finally PRED, RES and WRES, which are NONMEM-generated items — the predicted value of DV, the residual (DV-PRED), and the weighted residual (a transformation of RES to units of the population standard deviation).

(See **data record**, **data set**).

REFERENCES: Guide IV, section III.B.2

DATA RECORDS

MEANING: Organized elements of a NONMEM Data Set
CONTEXT: NONMEM output

DISCUSSION:
A NONMEM data set consists of a series of data records; each data record, in turn, consists of a series of data items, the names of which are defined on the $INPUT record. NONMEM counts the data records and displays this count in its output as a check for the user. This count is of logical records as read by a FORTRAN format specification, which is also displayed in the output. If the format specification contains no slash (/) characters (which denote the start of a new FORTRAN record), then each logical record corresponds to a FORTRAN record.

```
NO. OF DATA RECS IN DATA SET:  394
```

(See **data item**, **data set**).

REFERENCES: Guide IV, section II.C

DATA SET

MEANING: The data to be analyzed by NONMEM
CONTEXT: NONMEM output

DISCUSSION:
"Data set" refers to all of the data (from all of the individuals) that is to be used in the NONMEM analysis. NONMEM is aware of certain features of the data. NONMEM lists what it knows about the data set as a check for the user.

```
DATA SET LOCATED ON UNIT NO.:    2
THIS UNIT TO BE REWOUND:        NO
NO. OF DATA RECS IN DATA SET:  394
NO. OF DATA ITEMS IN DATA SET:   8
ID DATA ITEM IS DATA ITEM NO.:   1
DEP VARIABLE IS DATA ITEM NO.:   4
```

The above states that the data set is associated with a file on FORTRAN logical unit 2. The rewind feature ("UNIT TO BE REWOUND ...") is of interest only if this is not the first problem of the NONMEM run. The other lines report how many records were read, the number of data items on each record, and the index order of the two NONMEM data items, ID (identification) and DV (dependent variable).

(See **data record**, **data item**).

REFERENCES: Guide IV, section III.B.2

DATE DATA ITEM

MEANING: Date (DATE) data item for NM-TRAN
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... DATE[=DROP] ...`

DISCUSSION:
`DATE`, `DAT1`, `DAT2`, and `DAT3` are labels for the date data item. The date data item is optional. When it is present, NM-TRAN performs time translation for the time data item (to relative times starting at 0), whether or not any time values contain the character ":".

The label given to the date data item describes its format:

`DATE`	month day year
`DAT1`	day month year
`DAT2`	year month day
`DAT3`	year day month

When the date represents a calendar date, components are separated by any nonnumeric character, such as - or / (e.g., 12-3-90 or 12/3/90). `DATE=DROP` should be specified in this case because the NONMEM data set must not contain non-numeric characters.

If only one component is present, it is assumed to be the day (and it may be 0 or negative). If two components are present, they are assumed to be the day and month, and they must not exceed 31 and 12, respectively.

EXAMPLE:

NM-TRAN data set

ID	DATE	TIME
1	10-1-86	9:15
1	10-1-86	14:40
1	10-2-86	8:30
2	10-12	8
2	10-14	10

NONMEM data set

ID	TIME
1	0.00
1	5.42
1	23.25
2	0.00
2	50.00

If present, the year component may consist of one to four digits. The value of NM-TRAN's LAST20 constant determines the assumed century of one and two digit years: 1900's or 2000's (See **last20**).

REFERENCES: Guide IV, section II.C.2 , III.B.2

DDEXPAND

MEANING: DDEXPAND program
CONTEXT: NONMEM run

USAGE:

```
..\util\ddexpand  < original.ctl > new.ctl
```

DISCUSSION:
With NONMEM 7.4, The utility program ddexpand in the ..\util directory will expand an NM-TRAN template control stream to propagate and incorporate delay differential equations (dde).

Example:

```
ddexpand dde3.dde dde3.ctl
```

The destination file dde3.ctl will be produced, and a finedata control stream file (fine.ftl) will be produced and submitted to finedata utility to add extra doses to the data file identified in the dde3.dde control stream.

For details and examples,
See **INTRODUCTION TO NONMEM 7, ddexpand Utility Program**

REFERENCES: Guide Introduction_7

DEFAULT COMPARTMENT

MEANING: Default compartments for kinetic models with PREDPP
CONTEXT: NONMEM data set

DISCUSSION:

The prediction computed by PREDPP is a scaled drug amount in some compartment called the observation compartment. Every dose is input into some dose compartment.

With every kinetic model there are designated a default observation compartment and a default dose compartment. For some kinetic models, these designations are built into the model definition; for others, these designations are given by the user in the user subroutine MODEL (or in $MODEL abbreviated code).

Unless otherwise specified on an observation event record by means of the compartment data item (CMT), the observation compartment is taken to be the default observation compartment. Unless otherwise specified on a dose event record by means of the compartment data item, the dose compartment is taken to be the default dose compartment.

The models for which default designations are built-in are listed here. The numbers are the compartment numbers.

Model	Default dose compartment	Default observation compartment
ADVAN1	1	1
ADVAN2	1	2
ADVAN3	1	1
ADVAN4	1	2
ADVAN10	1	1
ADVAN11	1	1
ADVAN12	1	2

(See **cmt**, **$model**, **model**).

REFERENCES: Guide VI, section V.H , VI.B , VII.C
REFERENCES: Guide IV, section V.C.4

DES AES: ICALL,IDEFD,IDEFA

MEANING: PREDPP read-only global variables
CONTEXT: DES and AES routines

USAGE:
```
USE PRMOD_INT, ONLY ICALL=>ICALLD,IDEFD,IDEFA
```

GLOBAL DECLARATION:
```
INTEGER(KIND=ISIZE) :: ICALLD,IDEFD(2),IDEFA(2)
```

DISCUSSION:

`ICALL`
Identical to the argument ICALL passed by NONMEM to PREDPP.

`IDEFD`
DES may set IDEFD when ICALL is 1, as follows.

IDEFD(1) may optionally be set by DES to indicate how many thetas it uses. Set to 0 if none. Otherwise, set to the index of the highest numbered theta used. IDEFD(1)=-9 means "unknown". PREDPP determines from the value stored in IDEFD(1) how many elements of theta to copy from its input argument THETA to THETAS.
(See **DES_AES:_THETA**).

When IDEFD(1) is set by DES to -9, PREDPP copies all thetas in the problem to THETAS (See **DES AES: THETA**). With NM-TRAN, IDEFD(1) is set to -9 when the $DES block contains verbatim code. If a user-written DES leaves IDEFD(1) unchanged, it defaults to -9, which does no harm, but may cause the run to be slower than necessary.

`IDEFD(2)` The full/compact flag for DES.

DES sets this as follows:
`IDEFD(2)=0` DES will return compact arrays.
`IDEFD(2)=1` DES will return full arrays (the default).

`IDEFA`
AES may set IDEFA when ICALL is 1, as follows.

IDEFA(1) is set by AES to indicate how many thetas it uses. See above remarks for IDEFD and DES.

`IDEFA(2)` Calling protocol for ADVAN9 and ADVAN15 and ADVAN17

AES sets this as follows:

`IDEFA(2)=-1` "call with every event record" (the default)
`IDEFA(2)=1` "call once per individual record"

IDEFA(2) applies only when no TIME data item
is defined. It is ignored when the TIME data item is defined.

Location prior to NONMEM 7: prdde1

REFERENCES: None.

DES AES: ISFINL

MEANING: PREDPP read-only global variables
CONTEXT: User-supplied DES and AES routines

USAGE:
```
USE PROCM_INT, ONLY: ISFINL
```

GLOBAL DECLARATION:
```
INTEGER(KIND=ISIZE) :: ISFINL
```

DISCUSSION:

`ISFINL`

When NONMEM is performing simulation or a copying pass (`COMACT`>0), DES and AES are called immediately after the advance to an event time or non-event time, with a value of `T` equal to this time.

`ISFINL` = 1 at such a call; otherwise =0.

NM-TRAN includes this global variable in the DES or AES routine when the $DES or $AES block includes references to variable ISFINL, or when verbatim code is present.

Location prior to NONMEM 7: procmb

REFERENCES: None.

DES AES: THETA

MEANING: PREDPP read-only global variables
CONTEXT: User-supplied DES, AES routines

USAGE:

```
USE PROCM_REAL, ONLY: THETA=>THETAS
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: LTH,DPSIZE
REAL(KIND=DPSIZE) :: THETAS(LTH)
```

DISCUSSION:

`THETA`

The THETA vector passed as an argument by NONMEM to PRED, PK, ERROR. IDEFD(1) and IDEFA(1) affect how many elements are copied.
(See **DES_AES:_ICALL,IDEFD,IDEFA**)

Location prior to NONMEM 7: procm6

REFERENCES: None.

DES

MEANING: DES subroutine
CONTEXT: User-supplied subroutine; for use with PREDPP's ADVAN 6,8,9,13,14,15,16,17,18

USAGE:

```
SUBROUTINE DES (A,P,T,DADT,IR,DA,DP,DT)
USE SIZES, ONLY: DPSIZE,ISIZE
INTEGER(KIND=ISIZE) :: IR
REAL(KIND=DPSIZE) :: A,P,DADT,DA,DP,DT
DIMENSION :: A(*),P(*),DADT(*),DA(IR,*),DP(IR,*),DT(*)
```

DISCUSSION:
The DES subroutine is called by PREDPP to evaluate right-sides of differential equations.

Input argument:

`A(n)`
: The amount in the nth compartment at time T.

`P(n)`
: The value of the nth PK parameter.

`T` Time. `T` takes values continuously over an integration interval.

Output argument:

`DADT(n)`
: The derivative with respect to T of the nth compartment's amount. It is important to note that PREDPP itself adds in the rates for any infusions that may be active.

  It is possible to introduce drug into a compartment by explicit terms in a differential equation, rather than by PREDPP dose event records. Drug introduced in this manner is not included by PREDPP in the computation of the output compartment. Specifically, the amount in the output compartment may be thought of as being calculated by summing all relevant doses from the INPUT file (i.e., those that precede the time of the present record, accounting for bioavailability), subtracting all amounts present in compartments other than the output compartment, and then multiplying the result by the output fraction parameter.

  For example, suppose differential equations were used for ADVAN2, rather than the analytic solution. They would be:

```
DADT(1)=-P(3)*A(1)
DADT(2)= P(3)*A(1)-P(1)*A(2)
```

`DA(n,j)`
: The derivative of DADT(n) with respect to A(j). Continuing the above example,

```
DA(1,1)=-P(3)
DA(2,1)= P(3)
DA(2,2)=-P(1)
```

`DP(n,j)`
: The derivative of DADT(n) with respect to P(j). Continuing the above example,

```
DP(1,3)=-A(1)
DP(2,3)= A(1)
```

```
DP(2,1)=-A(2)
```

`DT(n)`

The derivative of DADT(n) with respect to T.

The format of arrays DA, DP, DT described above is called "full format". Alternately, compact format may be used (See **prdde1**). It is the default format for these arrays. A description of compact format is beyond the scope of this document, but is described in the NONMEM 7.4 version of Guide VI, Appendix IV. Compact Arrays in DES.

Also see variables in NONMEM modules, NONMEM-PRED modules, and PREDPP modules.
(See **variables in modules**)
In particular,
(See **DES AES: ICALL,IDEFD,IDEFA**)

REFERENCES: Guide IV, section V.C.7
REFERENCES: Guide VI, section VI.C

DIFF EQ SOLVER SETTINGS

MEANING: PREDPP global variables
CONTEXT: For use with PREDPP

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of PREDPP code.

USAGE:

```
$PK
"FIRST
" USE PRCOM_INT, ONLY: METH,MITER,IMAX,ISTFLG,INTFLG
"MAIN
" IMAX=200000
```

DISCUSSION:
These variables allow the user to over-ride certain default settings in ADVAN6, ADVAN8, ADVAN9, ADVAN13, ADVAN14, ADVAN15, ADVAN16, ADVAN17, ADVAN18, SS6, and SS9.

DES_DER, MITER, and METH

PREDPP sets MITER and METH to default values with every new individual or reset record. METH is the solver method type, and MITER determines whether analytical Jacobian
is to be used (MITER=1, based on DA() array calculated in DES subroutine), or Jacobian is to be numerically determined by the solver (MITER=2). As of NM75, MITER may be accessed by the reserved variable DES_DER, without requiring the "FIRST" and "USE" header lines, or verbatim code. For example:

```
$PK
DES_DER=1
```

ADVAN8

METH=2 is always used.
If user set MITER, use his value. Otherwise, use MITER=2.

ADVAN9

If METH=1, then Adams implicit method, and METH=2 (default, BDF method).

If user did set MITER, use his value.

ADVAN13

METH set in $PK is not used. LSODA determines whether ADAMS NON-stiff (METH=1) or BDF stiff (METH=2) is to be used as it works its way through the problem.
MITER (or DES_DER) should be set only to 1 (analytical Jacobian) or 2 (numerical Jacobian determined by LSODA). Usually should be left alone.

ADVAN14

METH=2 by default. Could be set in cvodeu.f90 or $PK.
Setting MITER=1 (or DES_DER=1) is required if it is desired that the Jacobian be analytically evaluated for ODE models (ADVAN8, 9, 13, 14, 15, 16, 17), for IMP, SAEM, BAYES problems that normally do not have first derivatives turned on. In addition to setting DES_DER=1 in $PK, various options of Jacobians (analytical/numerical, full/band, etc) should be set in CVODEU.f90,

where there are many other things the user can set.

ADVAN15
: METH is not used. It is always METH=2 (BDF stiff).
Setting MITER=1 is required if it is desired that the Jacobian be analytically evaluated for ODE models (ADVAN8, 9, 13, 14, 15), for IMP, SAEM, BAYES problems that normally do not have first derivatives turned on. In addition to setting DES_DER=1 in $PK, various options of Jacobians (analytical/numerical, full/band, etc) should be set in IDAU.f90, where there are many other things the user can set.

ADVAN16
: METH and MITER are not used. Instead, IJAC would set in RADAR5u.f90, but presently this is not functional

ADVAN17
: METH and MITER are not used. Instead, IJAC would set in RADAR5u.f90, but presently this is not functional

ADVAN18
: METH and MITER are not used.

IMAX

ADVAN6, ADVAN8, ADVAN9, ADVAN13, ADVAN14, ADVAN15, ADVAN16, ADVAN17, ADVAN18, SS6
: The variable MAXCAL gives the maximum number of calls to FCN1 (ADVAN6, ADVAN8, ADVAN13, ADVAN14, ADVAN16, ADVAN18, SS6) or RES (ADVAN9, ADVAN15, ADVAN17) during an integration interval.
Each of the above routines sets MAXCAL to the value given by MAXFCN (a parameter in the MODULE SIZES) at the start of each integration interval unless the user supplies a value in IMAX, in which case the user's value is used.

ISTFLG

ADVAN9 ADVAN15, ADVAN17
: ISTFLG controls how ADVAN9 calls LSODI1 and what it does if LSODI1 returns and indicates that an integration failed. ISTFLG is set to 0 (default) at ICALL=0. If changed by the user, it retains the changed value until the user changes it again. ISTATE is a variable that is passed from the ADVAN to LSODI1. ISTATE=1 indicates that the integration is starting. ISTATE=2 indicates that this call is a continuation from a prior successful integration. Default: Use ISTATE=1 for the first integration, and ISTATE=2 for a continuation when nothing external to the ADVAN has changed. In case of failure with ISTATE=2, restore original inputs and try again with ISTATE=1.

ISTFLG=1
: Never try ISTATE=2, always use ISTATE=1.

ISTFLG=2
: Never retry (only try ISTATE=2).

ADVAN15 also uses ISTFLG for calls to IDA, but uncertainty exists as to what its usefulness is.

INTFLG

ADVAN6, SS6, ADVAN8, ADVAN13, ADVAN14, ADVAN16, ADVAN18

INTFLG stands for "Integration Flag" and affects the number of calls to the integrating subroutine during each advance. It is only of interest when second derivatives of the state vector are calculated.

It is present because there may be some trade-off between run time and accuracy of computation. More calls to the integrator, with a smaller number of derivatives obtained with each call, result in longer run times, but might also produce more accurate derivatives. Also, it might provide more consistent computations when the number of compartments and/or etas is to be changed.

Default is -1. User may set to any other value in user-written code (e.g., with verbatim code in $PK).

PREDPP examines this value after the first call to PK for an individual record, so that it can be set on an individual-by-individual basis. PREDPP examines INTFLG when NEWIND=0 or 1 (i.e., at the start of an individual's data). Presumably, this is the only time that NONMEM might change the LVOUT array (See **non-active eta list for pred**).

When INTFLG is set to any other value than -1, ADVAN 6,8,13,14,16,18 and SS6 calculate second derivatives "one group at a time".

E.g., with ADVAN6, suppose etas 1, 2, 3 are active.

1) call DVERK to obtain 2nd derivatives 1,1
2) call DVERK to obtain 2nd derivatives 1,2 and 2,2
3) call DVERK to obtain 2nd derivatives 1,3 and 2,3 and 3,3

(Each calculation involves the integration of the state vector, augmented by the relevant first derivative(s) and the second derivatives for one eta.)

Thus, the maximum number of differential equations that would ever be integrated at one time is

PW=2*PE*PM+PM

Where: (PE is the maximum number of etas; currently 10)
(PM is the maximum number of compartments - 1; currently 9)
PW is the size defined for various work arrays in the source code.
It is currently defined as above (189).

When INTFLG is -1, the ADVAN routines and SS6 makes the most efficient use of the work arrays when computing second derivatives, to reduce the number of calls to DVERK.

At NEWIND=0 or 1, PREDPP looks at the number of active etas for this individual and the number of user-defined compartments defined by the MODEL subroutine (NCM), and creates a scheme to calculate as many groups of 2nd. derivatives at once as it can. (Although compartments may be turned on and off within the data set, PREDPP does not revise the scheme of integration each time.)

E.g., with the current values of PE, PM, and PW, here are some schemes of integration. Under "nth. call" are the etas whose second derivatives are computed with that call to DVERK.

# compts.	1st. call	2nd. call	3d. call	4th. call
9	etas 1 - 5	etas 6 - 7	eta 8	eta 9
8	etas 1 - 5	etas 6 - 7	eta 8	eta 9
7	etas 1 - 5	etas 6 - 7	eta 8 - 9	

6	etas 1 - 6	etas 7 - 8	eta 9
5	etas 1 - 7	etas 8 - 9	
4	etas 1 - 8	etas 9	
3	etas 1 - 9		

Changing the size of the work arrays:

If the system is large enough that integration involves more than one call to DVERK, and the user would like all second derivatives to be computed with a single call to DVERK, the source code must be changed to define a larger value for PW.

To integrate the maximum number of etas and compartments in one call, set:

PW=PM*(1+PE+PE*(PE+1)/2)

With the current values of PE and PM, PW=594

Suppose PW is changed, but by accident is made smaller than the default (2*PE*PM+PM=189). Then for problems with large numbers of compartments and/or etas, the work arrays will not be large enough, with either INTFLG=-1 or INTFLG!=-1.

A new error message exists in PREDPP, for which PRED exit code is 2 (always abort).

WORK ARRAYS ARE TOO SMALL FOR 2ND. DERIVS. INCREASE PW, OR DECREASE NO. OF. COMPTS AND/OR ETAS, OR USE DERIV2=NO

Again, this message cannot occur unless the source code of PREDPP is changed incorrectly.

Location prior to NONMEM 7: prcomg

REFERENCES: None.

DISPLAYED PRED-DEFINED ITEMS

MEANING: Displayed values of PRED-Defined Items
CONTEXT: NONMEM output

DISCUSSION:
PRED-defined items can be displayed in tables and scatterplots. (This includes PK-defined and ERROR-defined items.) These items are computed and recomputed many times during a NONMEM run, with various values of theta and eta. It is important to know which values are used in the computation of the items as displayed.

Consider this fragment of an NM-TRAN control stream:

```
$PK
CL=THETA(1)*EXP(ETA(1))
$TABLE CL ETA(1)
```

The values of ETA(1) and the PRED-defined item CL that are displayed depend on which task specification records are present.

Task specification records	THETA	Displayed values ETA(1)	CL
none	initial	0	typical
$EST (or $MSFI) METH=0	final	0	typical
$EST (or $MSFI) METH=1 or POSTHOC	final	conditional (CPE)	conditional (CPE)
$SIMULATION	initial	0	typical
$SIMULATION ... ONLYSIM	initial	simulated†	simulated
$NONPARAMETRIC ETAS	final	nonparametric (CNPE)	nonparametric (CNPE)

† Displayed etas are the simulated etas if the simulated etas are stored in common NMPRD7. This is the case when NM-TRAN is used. If they are not stored in NMPRD7, displayed etas are all 0.

Displayed values of PRED, RES, and WRES are always computed with typical values of parameters, except as noted below.

Displayed values of DV are simulated values when $SIMULATION is present.

With NONMEM 7, Additional types of PRED, RES, and WRES values may be requested than the usual set available in NONMEM VI. They may be specified at any $TABLE command or $SCATTER command, as one would request PRED, RES, or WRES items. Such items are supplied internally by NONMEM, and in the case of L2 data, more accurately. An example of their use is in Example #4a, below.

Every NONMEM output report contains the line:
`THE FOLLOWING LABELS ARE EQUIVALENT`
Subsequent lines show the relationship of such items for the current Estimation method.
(See **$table**, **$scatter**).
(See **PRED,RES,WRES**).
See **INTRODUCTION TO NONMEM 7, $TABLE: Additional Statistical Diagnostics, Associated Parameters, and Output Format**

Suppose that marginal items appear. If this item is 1 or 2 on a given record, the values of PRED and any PRED-defined displayed item D (assuming D *is not* in the SAVE region

(See **comsav**) associated with this record) are determined as follows:

If the Simulation Step is implemented, and ONLYSIMULATION is specified, the PRED item is the simulation expectation of F, over the simulated values of eta for all the individuals in the data set (including those individuals with no observation records), and the D item is the simulation expectation of the variable D.

If conditional estimates are obtained (e.g. with the use of the POSTHOC option), the PRED item is the posterior expectation of F, over the conditional estimates of eta for all individuals in the data set (that have some observation records), and the D item is the posterior expectation of the variable D.

Otherwise, the PRED and D items are the same as with a marginal item of 0 (see description above).

Suppose that the raw-data items appear. If this is 1 on a given record, the values of the DV, RES, and any PRED-defined displayed item D (assuming D *is* in the SAVE region (See **comsav**) associated with this record) are determined as follows:

The data record serves as a template record. The DV item appearing in a table or scatterplot is the raw-data-average of the values of DV in observation records matching the template record. The RES item is the difference between the PRED item and this average. A PRED-defined item D is the average of the values of D obtained with observation records matching the template.

(See **mrg**, **raw**, **template**).

Several examples will illustrate techniques for displaying (subject-specific) conditional values, (non-subject-specific) typical values, and simulated values.

#1 Display typical and conditional values using different variables:

```
$PK
 ...
TVCL=THETA(1)
CL=TVCL*EXP(ETA(1))
$ESTIM  ... POSTHOC ...
$TABLE TVCL CL
```

TVCL is a typical value by definition. CL is a conditional value.

#2 Display typical and conditional values of the same variable:

```
$PK
 ...
CL=THETA(1)*EXP(ETA(1))
IF (COMACT.EQ.1) TVCL=CL
$ESTIM  ... POSTHOC ...
$TABLE TVCL CL
```

TVCL is a SAVE variable because it is defined in a copying block. During the copying pass with COMACT=1, it is set to the typical value of CL, because when COMACT=1, etas are 0. Were the values of TVCL not stored in the SAVE region, then during the copying pass to obtain values for tables and with COMACT=2, TVCL would retain its value set with the previous data record.
The first column of the table is labelled TVCL and contains the typical value of CL, and the second column contains the conditional value.

#3 Display typical and conditional values of the prediction, and the individual residual and weighted residual values:

```
$ERROR
IPRED=F
IRES=DV-IPRED
IWRES=IRES/IPRED
Y=F+EPS(1)
$ESTIM  ... POSTHOC ...
$TABLE IPRED IRES IWRES
```

The IPRED column contains the conditional value of F ("individual prediction"). (F is not a left-hand quantity and cannot be specified in a $TABLE or $SCATTERPLOT record.) The PRED column contains the typical value. The IRES column contains the "individual residual". The IWRES column contains a value which is proportional to (but not equal to) the "individual weighted residual".

If predicted values of F may be zero, then division by zero can be avoided by code such as the following.

```
$ERROR
IPRED=F
IRES=DV-IPRED
DEL=0
IF (IPRED.EQ.0) DEL=1
IWRES=(1-DEL)*IRES/(IPRED+DEL)
Y=F+EPS(1)
```

#4 Display intra-individual weighted residual:

This uses the constant CV intraindividual error model, and allows one to display the weighted intra-individual residual that is used during Estimation when the FOCE method without interaction is used.

```
$ERROR
Y=F+F*EPS(1)
IPRED=F
IF (COMACT.EQ.1) FT=F
IF (FT.NE.0) WR=(DV-IPRED)/FT/SQRT(SIGMA(1,1))
IF (IPRED.NE.0) WR2=(DV-IPRED)/IPRED/SQRT(SIGMA(1,1))
$ESTIM METHOD=COND
$TABLE FT IPRED WR WR2
```

IPRED is the conditional estimate of F.
FT is a SAVE variable because it is defined in a copying block. During the copying pass with COMACT=1, it is set to the typical value of F, because when COMACT=1, etas are 0. Were the values of FT not stored in the SAVE region, then during the copying pass to obtain values for tables and with COMACT=2, FT would retain its value set with the previous data record.
WR is the weighted intraindividual residual with no eta-eps interaction. It is defined in a conditional assignment since it is possible that the typical value of F is zero with some non-observation record. (With previous versions of the help guide the SIGMA term was omitted and the WR was proportional to the weighted intraindividual residual.) It is a no-interaction residual because the denominator term, the residual standard deviation FT*SQRT(SIGMA(1,1)), represents the epsilon error, and uses the predicted value (FT) evaluated with eta=0, that is, eta does not interact with the epsilon error. WR2 is the conditional intra-individual weighted residual with eta-eps interaction. It is an eta-eps

interaction residual because the denominator term, the residual standard deviation IPRED*SQRT(SIGMA(1,1)), represents the epsilon error, and uses the predicted value (IPRED) evaluated with a non-zero eta, that is, eta interacts with the epsilon error.

#4a Display intra-individual weighted residual using reserved variables

With NONMEM 7, It is possible to obtain the same table more easily, as follows, using the reserved residual variables CIWRES (=WR) and CIWRESI(=WR2) :

```
$ERROR
Y=F+F*EPS(1)
$ESTIM METHOD=COND
$TABLE IPRD CIPRED CIWRES CIWRESI
```

#5 Display simulated values:

```
$PK
 ...
CL=THETA(1)*EXP(ETA(1))
 ...
$SIMULATION (seed) ONLYSIM
$TABLE ETA(1) CL
```

ETA(1) and CL are simulated values.

#6 Display simulated and estimated values:

```
$INPUT  .... CLSM E1SM ...
$PK
 ...
CL=THETA(1)*EXP(ETA(1))
IF (ICALL.EQ.4) THEN
    CLSM=CL
    E1SM=ETA(1)
ENDIF
$SIMULATION (seed)
$ESTIM
$TABLE CLSM E1SM CL ETA(1)
```

Modification of the data record during the simulation pass is the only way to save the simulated values in the absence of ONLYSIM. Column 1 and 2 of the table are the simulated values. Column 3 and 4 are the typical values.

#7 Display conditional values of a variable in each of two mixture subpopulations:

```
$PK
   ...
IF (MIXNUM.EQ.1) CL=  ...
IF (MIXNUM.EQ.2) CL=  ...
IF (COMACT.EQ.2) THEN
  IF (MIXNUM.EQ.1) CL1=CL
  IF (MIXNUM.EQ.2) CL2=CL
ENDIF
$ESTIM  ... POSTHOC ...
$TABLE CL1 CL2
```

CL1 (and CL2) is a SAVE variable because it is defined in a copying block. Were the values for CL1 not stored in the SAVE region, during the copying pass to obtain values for tables and with MIXNUM=2, CL1 would retain its value set with the previous data record.

#8 Display conditional values of a variable in the subpopulation into which the individual is classified:

```
$PK
   ...
IF (MIXNUM.EQ.1) CL=  ...
IF (MIXNUM.EQ.2) CL=  ...
IF (COMACT.EQ.2.AND.MIXNUM.EQ.MIXEST) CLE=CL
$ESTIM  ... POSTHOC ...
$TABLE CLE
```

CLE is a SAVE variable because it is defined in a copying block. Were the values for CLE not stored in the SAVE region, during the copying pass to obtain values for tables and with MIXNUM>MIXEST, CLE would retain its value set with the previous data record.

REFERENCES: Guide IV, section III.B.7 , IV.E.2
REFERENCES: Guide VI, section III.J , IV.E
REFERENCES: Guide Introduction_7

DOEXPAND

MEANING: DOEXPAND program
CONTEXT: NONMEM run

USAGE:

```
..\util\doexpand  < original.ctl > new.ctl
```

DISCUSSION:
With NONMEM 7.4, The utility program doexpand augments abbreviated code in an NM-TRAN control stream file that has been annotated with DOE (which stands for DO expand) and ENDDOE (which stands for ENDDO expand) directives. This is useful for repetitive code statements that differ only by increments of indices. For example, it is useful for replicating derivative equations that differ only by their indices to DADT() and A(). These may occur with time delay and transit compartment differential equations.

For details and examples,
See **INTRODUCTION TO NONMEM 7: doexpand Utility Program**

See also The DOPDE Method of Modeling PDE's.

REFERENCES: Guide Introduction_7

DOSE EVENT RECORD

MEANING: Specification of dose for PREDPP
CONTEXT: $INPUT record and NONMEM data set

DISCUSSION:
PREDPP recognizes two varieties of doses, transient and steady-state. Steady-state doses are described separately (See **SS_dose_event**). Transient doses are described here.

For a transient dose, the SS data item must be not defined or must be zero. The ADDL data item may be used to specify additional transient doses at regular intervals given by II. The dose is introduced into a dose compartment n, which is defined either explicitly in the event record (by the value n>0 of the CMT data item), or as the default compartment for doses for the particular ADVAN (when the CMT data item is not defined or contains zero).

Absorption lag applies to the initiating dose and to all subsequent additional doses. If an absorption lag parameter is defined by PK for the dose compartment (this parameter is coded `ALAGn` in abbreviated code) and has a positive value, then the dose is referred to as a "lagged dose" and it is actually introduced into the system at a "lagged time" equal to the time at which the dose would ordinarily be introduced plus the value of the absorption lag parameter. The value of ALAGn in effect at time T applies to the dose and to all subsequent additional doses, even if ALAGn changes value with a subsequent event record.

Bioavailability applies to the initiating dose and to all subsequent additional doses. A bioavailability parameter may be defined by PK for the dose compartment (it is coded `Fn` in abbreviated code) and must have a positive value. If no such parameter is defined, the bioavailability parameter is assumed to have the value 1. The value of amount in the following discussion is the value of the AMT data item multiplied by the value of the bioavailability parameter in effect at the time the dose is actually introduced. Changes in the value of Fn at later times have no affect on past doses, e.g., infusions and zero-order bolus doses that are already in progress are unaffected.

The type of dose is described by the values of the AMT and RATE data items.

AMT	RATE	
>0	0	Bolus dose. The dose is introduced into the dose compartment at the event time (before ERROR is called with the event record).
>0	>0	Infusion dose ("regular infusion"). The infusion is started at the event time. Its duration is computed as amount/RATE.
>0	-1,-2	Zero-order bolus dose. Similar to a regular infusion, except that the duration of a regular infusion is specified by information in the dose event record and computed by PREDPP itself, whereas the duration of a zero-order bolus dose is regarded as a parameter which may be modeled and computed by the user's PK routine or $PK abbreviated code. The infusion is started at the dose event time.

If RATE=-2, the duration is computed by PK. The parameter is coded `Dn` in abbreviated code, e.g., D1 models the duration of zero-order bolus doses to compartment 1. The duration is given by the value of Dn as computed by PK at the dose event time (or lagged time, if the dose is lagged). Rate is computed as amount/duration and is thus fixed for this infusion, even if the value of Dn changes value with a later event record.

If RATE=-1, the infusion rate is computed by PK. The parameter is coded `Rn` in abbreviated code, e.g., R1 models the rate of zero-order bolus doses to compartment 1. The rate which applies during the advance from a particular state time to a later state time is given by the value of Rn in effect for the the later time, and may change during the course of the infusion. The duration at any state time is computed as amount/rate, where amount represents the remaining dose amount and rate represents the value of Rn at the state time. The infusion continues until the entire dose amount has been introduced into the system.

CAUTION: If PREDPP is used without using NM-TRAN as a preprocessor then any user-supplied negative values for these data items will be passed to PREDPP. For a discussion of the circumstances such that the data file used by NONMEM and PREDPP is FDATA (in which case NM-TRAN has preprocessed it) or is the file named in the $DATA record, see NONMEM Users Guide, Part IV.

(See **multiple dose example**, **exogenous supplementation example**).

REFERENCES: Guide VI, section III.B.2 , V.F , V.G , V.H

REFERENCES: Guide V, section 6.8

REFERENCES: Guide IV, section III.B.5

DOSE RECORD: DOSREC

MEANING: PREDPP read-only global variables
CONTEXT: User-supplied PK, DES, AES routines

USAGE:

```
USE PROCM_REAL, ONLY: DOSREC
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: VSIZE,DPSIZE
REAL(KIND=DPSIZE) :: DOSREC(VSIZE)
```

DISCUSSION:

`DOSREC`

At a call to PK which occurs at a non-event dose time (i.e., at the time of an additional or lagged dose), `DOSREC` contains the (last data record of the) event record describing the dose.

When PK is called at an event time, `DOSREC` contains 0's.

(See **Dose Time Non-Event: DOSTIM**).

Location prior to NONMEM 7: procm3

REFERENCES: Guide VI, section III.I

DOSE TIME NON-EVENT: DOSTIM

MEANING: PREDPP read-only global variables
CONTEXT: User-supplied PK, DES, AES routines

USAGE:

```
USE PROCM_REAL, ONLY: DOSTIM,DDOST,D2DOST
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: LVR,DPSIZE
REAL(KIND=DPSIZE) :: DOSTIM,DDOST(LVR),D2DOST(LVR,LVR)
```

DISCUSSION:

`DOSTIM`

`DOSTIM`=0: this call to PK occurs at an event time. `DDOST`, `D2DOST`, and `DOSREC` contain zeros.

`DOSTIM`>0: this call to PK occurs at a non-event dose time `DOSTIM`, i.e., at the time of an additional or lagged dose. `DDOST`, `D2DOST`, and `DOSREC` contain values of interest. (See **Dose Record: DOSREC**).

`DDOST`

`DDOST(i)` = Partial derivative of `DOSTIM` with respect to eta(i).

`D2DOST`

`D2DOST(i,j)` = Second partial derivative of `DOSTIM` with respect to eta(i), eta(j) (lower-triangular; j=1, ..., i).

When `DOSTIM`>0, TIME and all user (concomitant) data items in EVTREC are from the next event record. All other NONMEM/PREDPP reserved data items are from the initiating dose event record. (The $BIND record may be used to override this; (See **bind**)

Location prior to NONMEM 7: procm2

REFERENCES: Guide VI, section III.I

DOWHILE BLOCK

MEANING: Coding technique
CONTEXT: Abbreviated code

DISCUSSION:

SAMPLE:

```
$ABBR DECLARE DOWHILE ILOOP $PK
ILOOP=1
DOWHILE (condition)
 .. statements ..
ILOOP=ILOOP+1
ENDDO
```

DISCUSSION:

DOWHILE blocks allow loops in abbreviated code. Previously, they were permitted only in initialization-finalization, simulation, data average, and expectation blocks. See the separate descriptions of these blocks. With NONMEM 7.3, DOWHILE may also be used in data-analytic blocks (i.e., in code that is executed at ICALL=2).

When it is used in data-anlytic code, DOWHILE must be used with a looping variable. Such a variable meets three criteria:
(1) It is declared as such, e.g.

```
$ABBR DECLARE DOWHILE ILOOP
```

(2) It is given a value in the statement immediately before DOWHILE
(3) It is given a value in the statement immediately before ENDDO.

Usually, the looping variable will be tested in the condition, but this is not necessary.

DOWHILE blocks may be nested.

DOWHILE blocks may not include IF/THEN/ENDIF blocks. Only single-statement IF statements are permitted. E.g.,

```
IF (condition) statement
```

Subscripted variables may be used within the block. These are variables that have been declared arrays using the $ABBR statement, or subscripted reserved variables such as THETA. Subscripts may be integer expressions using declared integer variables and integer values. E.g.

```
$ABBR DECLARE X(10)
$ABBR DECLARE DOWHILE ILOOP
 ...
$PK
ILOOP=1
DOWHILE (condition)
X(ILOOP)=THETA(ILOOP+1)
ILOOP=ILOOP+1
ENDDO
```

(See **abbreviated code**).

A DOWHILE loop may compute recursive random variables. These are variables that are modified recursively in a dowhile block and which have eta derivatives. For example,

```
TERM=THETA(1)*EXP(ETA(1))
SUM=0
```

```
ILOOP=1
DO WHILE(ILOOP<=IMAX)
SUM=SUM+TERM
ILOOP=ILOOP+1
ENDDO
```

The dowhile recursive variable must be initialized to a non-random variable outside the loop and must appear on both sides of the equal sign only once within the DOWHILE loop: V= ... V ...

The syntax is very limited. The following are not permitted:

```
ILOOP=1
DO WHILE(ILOOP<=IMAX)
IF (ILOOP == 1) SUM=0  ; initialization within the loop
SUM=SUM+TERM
ILOOP=ILOOP+1
ENDDO

ILOOP=1
DO WHILE(ILOOP<=IMAX)
IF (ILOOP == 1) THEN
SUM=0                  ; initialization within the loop
ELSE                   ; else statement
SUM=SUM+TERM
ENDIF
ILOOP=ILOOP+1
ENDDO
```

Although the example given is a summing loop, a product loop such as

```
PROD=PROD*TERM
```

is also possible, as are other ways the dowhile recursive variable can be used. The looping variable ILOOP may also be set and tested and incremented with different integers than shown.

EXAMPLES:

Several examples are present in the examples directory:

Auto-correlation (See ar1mod.ctl)
Auto-correlation with Simulation (See ar1newsim.ctl).
Dose Superposition (See sumdosetn.ctl)
Summing elements of THETA (See superid3_6.ctl)

REFERENCES: None.

DURATION PARAMETER

MEANING: Duration (D) parameter for PREDPP
CONTEXT: Additional PK Parameters

USAGE:

```
$PK
 D1= ....
```

DISCUSSION:

Duration parameters are used with PREDPP. They are optional additional PK parameters. With NM-TRAN, they are symbolized in the $PK block by reserved variables `Dn`, where n is the compartment number to which the parameter applies.

There is one duration parameter associated with every possible dose compartment of the kinetic model (the output compartment is not a possible dose compartment) and the duration parameter used for a given dose is that one associated with the compartment into which the dose is given (the dose compartment).

A duration-modeled zero-order bolus dose is actually an infusion, but one whose duration is given by the rate parameter computed by the PK routine.

Duration parameters are optional in the sense that duration parameters associated with compartments never receiving duration-modeled zero-order bolus doses may be ignored. However, if the RATE data item on some dose event record contains the value -2, then a duration parameter for the dose compartment must be computed in the PK routine.

A duration parameter is computed by the PK routine using, if needed, information in the dose record. Therefore, its value for the given dose is constant over time.

When additional doses are specified on a dose event record the duration parameter applies to the dose and to all the additional doses.

(See **cmt**, **rate**, **pk**, **$pk**, **default_compartment**). (See **multiple dose example**).

REFERENCES: Guide IV, section V.C.5
REFERENCES: Guide V, section 12.2.3
REFERENCES: Guide VI, section III.F.3

DV AND MDV DATA ITEMS

MEANING: Dependent Variable (DV) and Missing Dependent Variable (DV) data items for NONMEM
CONTEXT: $INPUT record and NONMEM data set

USAGE:
```
$INPUT ... DV MDV ...
```

DISCUSSION:

`DV` and `MDV` label NONMEM's dependent variable (DV) and missing dependent variable (MDV) data items, respectively.

The dependent variable (DV) data item is required by NONMEM. It is the observed value associated with the data record.

The missing dependent variable (MDV) data item is optional. If it is present, then the ID data item must also be present.

Values of MDV are:

`0` The DV data item is an observed value, i.e., DV is not missing.

`1` The DV data item is not regarded an observed value, i.e., DV is missing. The DV data item is ignored.

`100`
Same as MDV=0, but this record is ignored during Estimation and Covariance Steps. During other steps, MDV will changed to 0.

`101`
Same as MDV=1, but this record is ignored during Estimation and Covariance Steps. During other steps, MDV will changed to 1.

Reserved variables MDVI1, MDVI2, MDVI3 can be used to override values of MDV>100. These variables are defined in include file nonmem_reserved_general, which needs to be copied from util to the present run directory so NMTRAN has access to it.

With PREDPP, event records whose types are dose, other-type, reset, or reset-dose (i.e., which have EVID values 1, 2, 3, 4) must all have MDV=1. NM-TRAN will append the MDV data item if PREDPP is used and MDV is not present on the $INPUT record.

(See **observation records**).

REFERENCES: Guide I, section B.1 , C.4.3
REFERENCES: Guide VI, section II , V.A
REFERENCES: Guide IV, section III.B.2 , V.B

ELS

MEANING: ELS subroutine
CONTEXT: NONMEM utility routine

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of NONMEM code and previous examples that may be available from advanced users.

USAGE:

```
USE SIZES,       ONLY:: ISIZE,DPSIZE,NO
 ...
INTEGER(KIND=ISIZE) :: IER1,IER2
REAL(KIND=DPSIZE)   :: CNT,D
DIMENSION :: D(NO,2)
 ...
CALL ELS (CNT,D,IER1,IER2)
```

DISCUSSION:
ELS is a NONMEM utility routine. It computes the Extended Least Squares contribution to the objective function for the data from a single individual ("L1") record. It may be called by CONTR or SPTWO.

Output argument:

`CNT`
: Contribution to -2log likelihood for data from the L1 record as computed by extended least squares.

`D(J,1)`
: value of RES for Jth observation record of the L1 record when called by SPTWO; otherwise 0.

`D(J,2)`
: value of WRES for Jth observation record of the L1 record when called by SPTWO; otherwise 0.

`IER1`
: =0 - Normal return.
>1 - Abnormal return.

`IER2`
: undocumented error return code.

ELS obtains all its inputs from NONMEM read-only variables.

Certain NONMEM error messages refer to ELS, for example:

```
PROGRAM TERMINATED BY OBJ, ERROR IN ELS
 WITH INDIVIDUAL 1 (IN INDIVIDUAL RECORD ORDERING)
 VAR-COV OF DATA FROM INDIVIDUAL RECORD ESTIMATED TO BE SINGULAR
```

A common situation giving rise to this message is the use of a proportional intra-individual error model while some predicted values for actual observations are zero or close to zero. (For example, if the first dose is an infusion and there is a "base- line" observation at the start of the infusion, the predicted level will be zero.)

REFERENCES: Guide I, section C.1
REFERENCES: Guide V, section 5.2.2

EMPIRICAL SS DOSE EVENT RECORD

MEANING: Specification of empirical steady-state dose for PREDPP
CONTEXT: $INPUT record and NONMEM data set

DISCUSSION:

With some systems, steady state cannot be computed by the usual methods requested by the SS data item. The Empirical steady state method computes steady state by giving a large number of doses until the state variables no longer change according to the SSTOL/SSATOL tolerance specified in $SUBROUTINES. The SS data item is not used. A negative value of ADDL requests the computation, and also specifies the maximum number of doses (ABS(ADDL)+1). Interdose interval II is used as usual.

Reserved variables ADDL_ACTUAL and ADDL_TIME and ADDL_TIMEDIFF are given useful values.

(See Guide Introduction_7 "An Empirical Method of Achieving Steady State")

REFERENCES: Guide Introduction_7

ERROR

MEANING: ERROR subroutine
CONTEXT: User-supplied subroutine; for use with PREDPP

USAGE: Versions before NONMEM 7.2:

```
SUBROUTINE ERROR (ICALL,IDEF,THETA,IREV,EVTREC,NVNT,INDXS,F,G,HH)
USE SIZES, ONLY: DPSIZE,ISIZE,LVR
INTEGER (KIND=ISIZE) :: ICALL,IDEF,IREV,NVNT,INDXS
REAL(KIND=DPSIZE) :: THETA,F,G,HH
DIMENSION IDEF(*),THETA(*),EVTREC(IREV,*),INDXS(*),G(LVR,*)
DIMENSION HH(LVR,*)
```

With NONMEM 7.2 and higher:

```
SUBROUTINE ERROR (ICALL,IDEF,THETA,IREV,EVTREC,NVNT,INDXS,F,G,HH)
USE SIZES,     ONLY: DPSIZE,ISIZE
USE PRDIMS,    ONLY: GERD,HERD
IMPLICIT REAL(KIND=DPSIZE) (A-Z)
REAL(KIND=DPSIZE) :: EVTREC
INTEGER(KIND=ISIZE) :: ICALL,IDEF,IREV,NVNT,INDXS
DIMENSION :: IDEF(*),THETA(*),EVTREC(IREV,*),INDXS(*)
REAL(KIND=DPSIZE) :: G(GERD,*),HH(HERD,*)
```

DISCUSSION:
The ERROR subroutine is called by PREDPP to model intra-individual error in observed values. It can also be used to to convert predictions from PREDPP, i.e., scaled drug amounts, to other types of predictions (for example, to obtain the prediction of a drug effect as a function of concentration, in a pharmacodynamic study).

Input argument:

`ICALL`

`ICALL=1`: ERROR has been called for initialization at the beginning of a NONMEM problem; one such call per problem. EVTREC contains the first event record. THETA contains the initial estimates. ERROR must return values in IDEF which inform PREDPP what tasks it will perform at later calls. It may also set elments of HH to the appropriate derivatives, when it has requested that ERROR be called only once per problem.

`ICALL=2`: ERROR has been called to obtain the modeled value; multiple calls occur. ERROR should compute derivatives of F for HH. If ERROR changes F, it should also change the derivates of F in G.

`ICALL=4`: ERROR has been called during the Simulation Step; multiple calls occur. ERROR should compute simulated observations and store them in F.

`ICALL=5`: ERROR has been called during the computation of expectations; multiple calls occur. Such calls occur when the marginal (MRG_) data item is defined in the data set and has non-zero values for some records. If the MRG_ data item has the value 1 or 2, expectations are computed for that record, and the value returned by ERROR in F contributes to the expectation that is being computed for the PRED data item. The value of an ERROR-defined item contributes to the expectation computed for the item.

`ICALL=6`: ERROR has been called during the computation of raw data averages; multiple calls occur. Such calls occur when the raw-data (RAW_) data item is

defined in the data set and has the value 1 for some records (See **template**). ERROR may re-compute DV when the value of DV in the data record is not the quantity to be included in the average. ERROR may return a value of 1 in F if no DV item is to be included in the average with the particular record.

`THETA`
The NONMEM THETA vector.

`EVTREC`
The PREDPP event record.

`INDXS`
The values specified in the $INDEX record of the NM-TRAN control stream. (This is the NONMEM INDXS array starting at position 12, the first position beyond those positions used by PREDPP itself.)

Output argument:

`IDEF`
ERROR should store values in IDEF only when ICALL=1. A value of 1 in IDEF(1) indicates that ERROR will store the derivatives of log y in HH (which causes PREDPP to multiply each element of HH by F). That is, PREDPP understands that the "error in log y" is modeled.

The value in IDEF(2) describes when ERROR should be called:
`-1` call with every event record.
`0` call once per observation record.
`1` call once per individual record.
`2` call once per problem.

The value in IDEF(3) describes whether ERROR uses derivatives of compartment amounts (i.e. whether compartment amounts themselves are used as random variables in arithmetic statements in ERROR).
`-1` ERROR may use derivatives of A.
`0` ERROR does not use derivatives A.
`1` ERROR does use derivatives of A.

The default used by PREDPP is IDEF(3)=-1. However, when ERROR does not use A, then if IDEF(3) is set to 0, PREDPP can avoid some time-consuming processing. Indeed, when $ERROR abbreviated code is supplied, and there is no reference to a compartment amount A(n) (as a random variable in an arithmetic statement) in the abbreviated code (or to its derivatives in verbatim code), then NM-TRAN sets IDEF(3)=0.

Input/Output argument:

`F` On input, the prediction based on the pharmacokinetic model, i.e., the value of the scaled drug amount in the observation compartment. On output, F may be unchanged, or it may be modified, e.g., when a PD prediction is needed. If ERROR modifies F and uses population eta variables to do so, then at ICALL=2 ERROR must call GETETA to obtain eta values prior to modifying F. When ICALL=4, ERROR should calculate the simulated observation (after calling SIMETA and/or SIMEPS to obtain simulated etas and epsilons, as necessary) and place its value in F. With the Simulation Step, ERROR may return the simulated observation as the DV data item, rather than in the argument F. With odd-type data the simulated observation must be returned as the DV data item. (See **$estimation**).

`G` On input, at ICALL=2 when the data are population, an array of partial derivatives of F with respect to etas
G (i,1) is the partial derivative of F with respect to eta(i).
G (i,j+1) is the second derivative of F with respect to eta(i), eta(j) (lower-triangular; j=1, ..., i).
(Second derivatives are only needed with estimation by the Laplacian method.)
On input, `G` contains zeros when the data are single-subject data.
At ICALL=2, ERROR must modify G when it has changed F and has thus changed the derivatives of F with respect to the population etas.

`HH` An array of partial derivatives of F with respect to etas (when the data are single-subject data) or epsilons (when the data are population). Values should be stored when ICALL=2 and also when ICALL=1 if ERROR sets IDEF(2)=2.
HH(i,1) is the derivative of F with respect to eta(i) or eps(i).
HH(i,j+1) is the derivative of H(i,1) with respect to eta(j) (but are only needed with conditional estimation when the dependence on etas of the variance of intra-individual random error should be preserved in the computation of the objective function; see the INTERACTION option (See **$ESTIMATION**).

Also see NONMEM read-only modules (of the form ROCMn), NONMEM-PRED modules (of the form NMPRDn), and PREDPP read-only modules (of the form PROCMn).

REFERENCES: Guide IV, section V.C.6
REFERENCES: Guide V, section 8
REFERENCES: Guide VI, section IV

ESTIM COVAR ERROR CODES

MEANING: NONMEM read-only global variables

CONTEXT: PRED routine

USAGE:

```
USE ROCM_INT, ONLY: IERE=>IEST_ERR,IERC=>ICOV_ERR
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: IEST_ERR,ICOV_ERR
```

DISCUSSION:

`IERE`
: The return code from the Estimation Step.

`IERC`
: The return code from the Covariance Step.

Values of 0 indicate normal termination.

These variables may be used as right-hand quantities in abbreviated code for initialization/finalization blocks.

Location prior to NONMEM 7: rocm9

REFERENCES: None.

ETABAR

MEANING: NONMEM's estimate of the bias in the underlying assumption about eta.
CONTEXT: NONMEM output

DISCUSSION:

ETABAR is printed when a conditional population estimation method is used. The following is an example.

```
 ETABAR IS THE ARITHMETIC MEAN OF THE ETA-ESTIMATES,
 AND THE P-VALUE IS GIVEN FOR THE NULL HYPOTHESIS THAT THE TRUE MEAN IS 0.

 ETABAR:         -3.5224E-02 -7.1437E-05  2.5095E-03
 SE:              3.4060E-01  3.1223E-03  1.9001E-01
 N:                       12          12          12

 P VAL.:          9.1763E-01  9.8175E-01  9.8946E-01
```

The ith number listed after "ETABAR" is the sample average (across individuals) of the conditional estimates of the ith eta, and the ith number listed after "SE" is the standard error for this average. Under the assumed model, the population average of the the conditional estimates is approximately zero. If the model is well-specified, the sample average should be near 0. (but see below for a mixture model). The P-value helps one assess whether the sample average is "far" from 0. A value under 0.05, for example, indicates such an average (notice the value 0.32E-02).

With a mixture model, the ith eta is understood to have a different distribution for each subpopulation of the mixture. Accordingly, different instances of the above output will appear, one for each of the different subpopulations. Using a standard Bayesian-type computation, each individual is classified into one of the subpopulations, and the conditional estimate of the ith eta under the model for this subpopulation is used in the sample average for that subpopulation. If under the mth submodel, the ith eta does not influence the data from any individual, but it does influence the data from some individual under some other submodel, then the sample average for the ith eta for the mth submodel will be 0. If the ith eta does not influence the data from any individual under any model, then the sample average for the ith eta for the mth submodel will usually be 0, but it will not be if (i) the ith eta is correlated with an eta that influences some individual's data under the mth submodel, and (ii) that individual is classified to be in the mth subpopulation.

The population average of the conditional estimates is only approximately zero because a conditional estimate is a (Bayesian) posterior mode, and not a posterior expectation. However with a mixture model, with the estimate for a given individual, the posterior distribution is that for the subpopulation into which the individual is classified, and due to possible missclassification the expectation of the estimate may be even "further from" zero than with a nonmixture model. For this reason too, the centered FOCE method may not work well with a mixture model.

With a mixture model, or with a nonmixture model, one may implement a second Estimation Step (in a subsequent problem), and then a second ETABAR estimate (EB2) can be obtained, with which the first ETABAR estimate (EB1) can be compared. If the data-analytic model is well-specified, the two estimates should represent nearly the same quantity. Using an option on the $ESTIMATION record, the second P-value assesses the magnitude of the difference between EB1 and EB2, and a P-value under 0.05 would suggest

that the data-analytic model is not well-specifed. To obtain EB2, a data set is simulated under the fitted model, and EB2 is obtained using this data set. Both EB1 and EB2 are (univariate) measures of central tendency of the distribution of interindividual "residuals", i.e. the distribution of the conditional estimates of the etas. In both cases the residuals are defined in terms of the data-analytic model. But for EB1, the distribution is governed by the true (unknown) model, and for EB2, the distribution is governed by the fitted model. If the two models are "close", EB1 and EB2 will be close. The conditional estimates of the etas from the simulated data should be based on the population parameter estimates from these data. It may cost considerable CPU time to obtain this second set of parameter estimates, and so it may not always be feasible to compute EB2.

One proceeds by constructing a problem specification that
(a) includes the same $INPUT record as was used with a previous problem wherein EB1 was obtained
(b) includes an $MSFI record specifying a model specification file from that previous problem, so that in particular, EB1 is available
(c) includes a $SIMULATION record with the option TRUE=FINAL, so that a data set will be simulated using the final parameter estimate from that previous problem.
(d) includes a $ESTIMATION record with the option ETABARCHECK (and either the option METHOD=COND or METHOD=HYBRID).
A data set will be simulated, and EB2 will be obtained. With the ETABARCHECK option, the P-value for the difference EB2-EB1 will be computed. Otherwise, if the model is a nonmixture model, EB1 is ignored, and the P-value will be simply that for EB2, and if the model is a mixture model, no P-value will be output (only the standard error for EB2 will be output). The numbers of data and/or individual records in the simulated data set may differ from those for the previous problem; so if desired, this data set can be much larger than the real data set.

REFERENCES: Guide VII, section II.A , III.D

ETASXI

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPRD_INT, ONLY: ETASXI
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE), ALLOCATABLE, DIMENSION(:)    :: ETASXI
```

DISCUSSION:

With NONMEM 7, an alternative eta shrinkage evaluation using empirical Bayes variances (EBVs, or conditional mean variances) are now also reported. The $ESTIMATION record option ETASTYPE=0 requests that eta shrinkage be averaged for all subjects. This is the default. With ETASTYPE=1, eta shrinkage is averaged only among subjects that provided a non-zero derivative of their data likelihood with respect to that eta.

Reserved variable ETASXI(i) may be used to specify certain etas of particular subjects to be included, or to specify certain etas of certain subjects to be excluded, from the average eta shrinkage assessment. ETASXI stands for eta shrinkage exclude/include. The subscript i refers to ETA(i).

If ETASXI(i) is set to 2, ETA(i) is included.
If ETASXI(i) is set to 1, ETA(i) is excluded.

This overrides whatever would happen based on ETASTYPE.

EXAMPLE:

```
ETASXI(3)=1
ETASXI(4)=2
```

For all subjects, ETA(3) is excluded and ETA(4) is included.

ETASXI can be set differently for certain subjects.

EXAMPLE:

```
IF(ID==3)  ETASXI(1)=1
IF(ID==23) ETASXI(3)=2
```

This excludes ETA(1) for subject 1, and includes ETA(3) for subject 23.

ETASXI may be set only in $PK and $PRED blocks.

The additional output file root.shm (which stands for shrinkage map) will contain information which etas were excluded in the eta shrinkage assessment.

REFERENCES: Guide Introduction_7

EVENT RECORD: EVTREC

MEANING: PREDPP read-only global variables
CONTEXT: User-supplied DES and AES routines

USAGE:

```
USE PROCM_REAL, ONLY: EVTREC
USE PROCM_INT, ONLY: NVNT=>NEVENT
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: PD,DPSIZE
REAL(KIND=DPSIZE) :: EVTREC(5,PD+1)
INTEGER(KIND=ISIZE) :: NEVENT
```

DISCUSSION:

`EVTREC`
: Identical to the argument EVTREC passed by PREDPP to PK and ERROR routine

`NVNT`
: Identical to the argument NVNT passed by PREDPP to PK and ERROR routines.

Location prior to NONMEM 7: procm7

REFERENCES: Guide VI, section III.C

EVID DATA ITEM

MEANING: Event Identification (EVID) data item for PREDPP
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... EVID ...`

DISCUSSION:
`EVID` labels PREDPP's event identification (EVID) data item. The event identification data item is required. NM-TRAN can supply this data item if dose and observation events are the only types of events present in the data. EVID can have one of four values in any event record.

0 Observation event. The DV data item is an observation. The CMT data item specifies which compartment is being observed. Dose-related data items (AMT, RATE, II, ADDL, SS) must be zero.

1 Dose event. The CMT data item specifies which compartment is being dosed. The DV data item is ignored. One or more of AMT, RATE, II, ADDL, SS data items must be non-zero to define the dose.

2 Other-type event. The DV data item is ignored. Dose-related data items must be zero. Examples of other-type events are: A compartment is turned on or off (CMT specifies which compartment is to be turned on or off); a prediction is obtained at a specified time so that it may be displayed in a table or scatterplot (PCMT specifies the compartment from which the prediction is obtained); some event occurs at a different time than any observation or dose event, e.g. a covariate such as weight changes, an intervention such as hemodialysis is started or stopped.

3 Reset event. The kinetic system is re-initialized. Time is reset to the time of the event record, the amounts in each compartment are reset to zero, the on/off status of each compartment is reset to its initial status. The DV data item is ignored. Dose-related data items must be zero.

4 Reset-and-dose event. The system is first reset, and then a dose is given. The DV data item is ignored.

With NONMEM 7, there is a change in the value of EVID under the following circumstances:
1) NMTRAN is appending EVID
2) The record is not a dose event (AMT, RATE, SS = 0)
3) MDV is present in the data set (i.e., was listed on $INPUT)
4) MDV is 1 or MDV is 101

Previous versions of NMTRAN always set EVID=0 (observation) when the record was not a dose event. NMTRAN will now set EVID=2 (Other-type event) for this case. Note that if the user supplies both EVID and MDV, there is no change. Similarly, if the user supplies neither EVID nor MDV, so that NMTRAN is appending both of them, there is no change. There is only a change when the user supplies MDV but not EVID, and has set MDV=1 or MDV=101 for non-dose records.

With NONMEM 7.2 and higher, the "Repeated Observation Records" feature exists to assist in specialized methodologies such as stochastic differential equations. A record in a data file may be set up for repeated calls to PK and ERROR. Each time, the same record is passed to PK and/or ERROR, but with a different EVID. The user's control stream

model in $PK or $ERROR may then take advantage of executing certain code conditional on the EVID value. For this to occur, the user must introduce one or more of the following data items in the data file, with these names:

```
XVID1 XVID2 XVID3 XVID4 XVID5
```

These stand for "extra" EVIDs. On the first call to PK/ERROR, the EVID is set to the value given in XVID1. On the second call, the EVID is set to that in column XVID2, etc. up to XVID5. Only as many XVIDs as are required are needed to be defined. All the other items in the record do not change, except that if the present EVID used is not 0, then the MDV value is set to 1 for that call. If an XVID is -1, then the call to PK/ERROR for that XVID is not made, nor for the remaining XVIDs. If there is an EVID column, the value in this column is not passed to PK/ERROR unless XVID1=-1 in which case a "normal" call with that record occurs.
(See **stochastic differential equation example**)

REFERENCES: Guide IV, section V.B
REFERENCES: Guide VI, section V.B

EXPECTATION BLOCK

MEANING: Abbreviated code for computation of expectations
CONTEXT: Abbreviated code

SAMPLE:

```
$ERROR
IF (ICALL.EQ.5) THEN
 ... expectation block ...
ENDIF
```

DISCUSSION:

An expectation block is a block of abbreviated code that is only executed when ICALL=5. This value of ICALL occurs when the marginal (MRG_) data item is defined in the data set and has a non-zero value for some records. Expectation blocks are not required when the marginal data item is present, but they allow the user additional functionality. Such blocks may be present in $PRED, $PK, $ERROR.

If the MRG_ data item has the value 1 in a data record, PRED- (PK- and ERROR-) defined items displayed for the record (e.g. in the row of a table corresponding to the record) are expectations. When ICALL=5, the expectations are being computed. With each call to PRED with the data record, the value being set in Y is contributing to the expectation being computed for the PRED item, and the value being set for a PRED-defined item that will be displayed in a table or scatterplot (except for an item stored in the SAVE region) is contributing to the expectation being computed for that item.

In this example, the displayed value of PRED with a record having MRG_=1 is the expectation of the 0-1 variable taking the value 1 if and only if F > 10.

```
IF (ICALL.EQ.5) THEN
 Y=0
 IF (F.GT.10) Y=1
 RETURN
ENDIF
```

Special rules apply to expectation blocks and passes through the data with ICALL=5.

1) No eta derivatives are computed in an expectation block.
2) Calls to certain NONMEM utility routines are permitted in an expectation block:

   ```
   CALL RANDOM(n,R)
   ```

   where n is an integer 1-10. If CALL RANDOM is present, R becomes a reserved variable, and may not be used outside the expectation block. Multiple calls to RANDOM may be present.
3) A RETURN statement may be used in an expectation block. If Y= appears (i.e. Y is assigned a value) in an expectation block before the RETURN (not necessarily the in same block containing the RETURN), then F is set to Y (F=Y); otherwise F is set to to 0 (F=0). If there is no RETURN statement in the expectation block, then as usual, F is set to the value of Y assigned by the time PRED (PK, ERROR) exits.
4) Loops are permitted in an expectation block. The syntax is as follows.

   ```
   DO WHILE (condition)
    .. statements ..
   END DO
   ```

5) If a mixture model is used, then during calls with ICALL=5, both MIXNUM and MIXEST are the index of the subpopulation into which the individual (whose data record is being passed) has been classified.

(See **mrg**).

REFERENCES: None.

FILES

MEANING: FILES Subroutine
CONTEXT: NONMEM utility routine

USAGE:

```
USE SIZES, ONLY: ISIZE
INTEGER(KIND=ISIZE) :: IUNIT
CALL FILES (IUNIT)
```

DISCUSSION:
Whenever a FORTRAN OPEN statement or a CLOSE statement is executed in a user-supplied routine, NONMEM should be informed. This should be done as follows: Immediately after the OPEN or CLOSE statement is executed, a call should be issued to FILES.

Input argument:

`IUNIT`
: An integer variable or integer constant; the number of the logical unit involved in the OPEN or CLOSE statement.

If reads or writes are issued to a given unit, but no OPEN statement for this unit is executed, then when ICALL is 0, a call to FILES should be issued (before any I/O statement involving this file is executed), where IUNIT is the number of the logical unit. However, reads and writes to a unit which has not been opened (connected) may not be allowed by some operating systems.

REFERENCES: Guide III, section V.10

FINAL PARAMETER ESTIMATE

MEANING: Final estimates of parameters THETA, OMEGA, SIGMA
CONTEXT: NONMEM output

DISCUSSION:
A page with the title `FINAL PARAMETER ESTIMATE` is printed as part of every NONMEM output.

If the Estimation Step was NOT run, then the values printed are the initial parameter estimates specified for the problem.

If the Estimation Step WAS run, then the values printed are the final parameter estimates.

These estimates of parameters are used by NONMEM to compute the objective function value printed on the page with title "MINIMUM VALUE OF OBJECTIVE FUNCTION" and to compute predicted values for Tables and Scatterplots.

```
********************  FINAL PARAMETER ESTIMATE   ***************
THETA - VECTOR OF FIXED EFFECTS PARAMETERS   *********

         TH 1      TH 2      TH 3
         1.94E+00  1.02E-01  3.20E+01

OMEGA - COV MATRIX FOR RANDOM EFFECTS - ETAS  ********

            ETA1
ETA1      8.99E-01

OMEGA - CORR MATRIX FOR RANDOM EFFECTS - ETAS  *******

            ETA1
ETA1      9.48E-01
```

This example is from a run involving single-subject data, so there is no SIGMA matrix.

With NONMEM 7.2 and higher, the correlation matrix is included. It is computed from the covariance matrix, whether or not the Estimation step is not run.

(See **initial parameter estimate**).

With NONMEM 7.3 and higher, the `O$THETAR` record may be used to transform final estimates of THETA before they are reported in the NONMEM output.
(See **$thetar**).

REFERENCES: Guide V, section 10.5

FINEDATA

MEANING: FINEDATA program
CONTEXT: NONMEM run

USAGE:

```
..\util\finedata fineplot.ftl
..\util\finedata < fineplot.ftl
```

DISCUSSION:
The utility program finedata augments an NM-TRAN data file to incorporate additional, non-observation, time values spaced at regular increments so that when a table is generated, NONMEM can fill these records with predicted values, from which smooth prediction curves may be plotted. If data items DV, EVID and/or MDV are listed in $INPUT and are present in the input data records, inserted records will be given the following values by default

DV=.
EVID=0
MDV=1

This may be over-ridden by a data item specification, such as $FINEDATA EVID=2. See also the EXAMPLE below.

Alternately, finedata can be used to fill in missing covariate values, but not to add any additional records.

fineplot.ctl is the name of a control stream file containing special commands for the finedata program.

The only records that finedata pays attention to is $INPUT, from which it obtains the column names, $DATA, from which it obtains the input data file, $FINEDATA, which contains instructions, and $PROB by which problems are separated.

With NONMEM 7.4, the $EXTRADOSE record may be used to add additional non-observation dose records to those already existing, but it allows items such as compartment number, or EVID, etc., to vary. The EXTRADOSE has particular value when adding time-delay compartments to a model, to deal with time-delay problems. For details:
See **INTRODUCTION TO NONMEM 7, finedata Utility Program**

All other control stream records are ignored. All options of $DATA are ignored, including RECORD and IGNORE options. (Input records whose first non-blank character is alphabetic are copied to the output file but not used otherwise.) Thus, a way to create a control stream is to copy the first records describing the data layout from an existing NONMEM control stream file, and then adding the $FINEDATA record.

Multiple data sets may be processed by one finedata control stream file, by using $PROB records to separate the problems.

The options to $FINEDATA are as follows:

```
$FINEDATA
            FILE= filename
            AXIS= label[ (LIN) | (LOG) ]
            NEVAL= [n | label ] | [TDELTA= [x | label ]
            [TSTART= [x | label | FIRST ]]
```

[TSTOP= [*x* | label | LAST]]
[OCC= label]
[item= list] ...
[item= [PREV | NEXT] [LIN | LINLIN | LOG | LOGLIN]] ...
[DELIM=s | DELIM=t | DELIM=,]
[MISSING=list]

$EXTRADOSE ...

SAMPLE:

```
$FINEDATA TSTART=0 TSTOP=50 NEVAL=100 AXIS=TIME(LIN) CMT=1,3
          FILE=example6b.csv
```

OPTIONS:

FILE=filename
: Required. The name of the output file.

AXIS= label[(LIN) | (LOG)]
: Required. Name of column containing times, usually TIME. Optionally, designate (LIN) or (LOG) in parenthesis, to indicate linear or geometric time incrementing when new records are incorporated.
If LIN: additive time increment=(tstop-tstart)/(neval+1)
If LOG: multiplicative time increment=(tstop/tstart)**(1/(neval+1))

Times may be entered as numerical values, or in hh:mm:ss format. Data sets with DATE/TIME records may also be processed (but then TSTART and TSTOP must be in numerical hours or hh:mm:ss format).

NEVAL= [*n* | label]
: The number of additional incremental time records to be created per subject. Either an integer value *n*, or the label of a data item in the original data set containing the NEVAL value. When NEVAL=-1, then missing values will be filled in for specified items, but no additional records will be created. If NEVAL/=-1, only the inserted records will have filled in interpolated values, and the original records will remain untouched.
Optional. One of NEVAL or TDELTA is required.

TDELTA= [*x* | label]
: The increment in time. Either a number, or the label of a data item in the original data set containing the TDELTA value. Optional. One of NEVAL or TDELTA is required.

TSTART= [*x* | label | FIRST]
: Optional. Default: FIRST
The start time for creating incremental time records. Either a real or integer numeric value *x*, or the label of a data item in the original data set containing the start time. If TSTART is not a number and is not FIRST, then it is interpreted as the label of a data item in the original data set containing the start time. If omitted or coded as FIRST, then the time of the first record of the subject. When TSTART is given by a data item, it can differ according to occasion. The same holds true for TSTOP and NEVAL (see below) if they are obtained from the data file.

TSTOP= [*x* | label | LAST]
: Optional. Default: LAST
The stop time for creating incremental time records. Either a numeric value real or integer *x*, or the label of a data item in the original data set containing the stop time,

or LAST, in which case the last record of the subject or occasion or time section is used.

Finedata stops inserting records when occasion changes (if OCC= was given), or if EVID=3 or 4, or after a re-initialization of time (indicated by the time in the data record being less than that of the previous record). TSTART will be the next TIME value.

`OCC`= label

Optional. Restart the time incrementing when the occasion changes, in addition to the other conditions listed above.

`item`= list

List of values for data item ITEM for which there is to be a record at each time increment. May be a list of values such as 1,3 etc. This can be done for a series of data items. For example, if you enter $FINEDATA CMT=1,3 EVID=2,2 then two records per time point are inserted, one with CMT=1, EVID=2, and the other with CMT=3, EVID=2.

`item`= [`LAST`|`NEXT`] [`LIN`|`LINLIN`|`LOG`|`LOGLIN`] ...

A scheme to determine how to supply values to various data items for these inserted records may also be given. The following may be interspersed with numeric values.

`NEXT`: When inserting records between two consecutive original records of time t1 (PREV) and t2 (NEXT), the PREDPP's default of using the covariate value of the t2 (NEXT) record is used for the inserted records. NEXT is the default.

`PREV`: When inserting records between two consecutive original records of time t1 (PREV) and t2 (NEXT), the covariate value of the t1 (PREV) record is used for the inserted records. (LAST may be coded instead of PREV, to be consistent with the options of the $BIND record. Note that the $BIND record is not used by finedata.)

`LIN`, or `LINLIN`: A covariate-linear, time-linear interpolation is used for the covariate value for the inserted records. LINT or LINLINT (T for truncate) produces truncated integer values, `LINR or LINLINR (R for round) produces values rounded to the nearest integer.`

`LOG`, or `LOGLIN`: A covariate-logarithmic, time-linear interpolation is used for the covariate value for the inserted records. A T or R suffix results in truncated or rounded integer values, respectively.

`LINLOG`: A covariate-linear, time-logarithmic interpolation is used for the covariate value for the inserted records. A T or R suffix results in truncated or rounded integer values, respectively.

`LOGLOG`: A covariate-logarithmic, time-logarithmic interpolation is used for the covariate value for the inserted records. A T or R suffix results in truncated or rounded integer values, respectively.

`DELIM=x`

Optional. DELIM=delimiter of output data file, if it is to be different from the input data file. DELIM=S is space, DELIM=t is tab, DELIM=, is comma.

`MISSING`=list

Optional. List is a comma delimted list of symbols that denote missing values. By default, a period (.) and space (s) are considered missing values. Values such as 0 or -99 may be present in the data as symbols for missing values. They may be described with MISSING=0 or MISSING=-99. During interpolation, missing values will be skipped, and only records with non-missing values will be used for

interpolation.

EXAMPLE:

With NONMEM 7.3 and higher, if an MDV is set to a value greater than or equal to 100, it is converted to that value minus 100 upon input, but will also not be used at all during estimation, only for table outputting. This option allows you to use the same enhanced data file for estimation and Table outputs, without significantly slowing down the estimation. So, the finedata control stream file might include

```
$FINEDATA TSTART=0 TSTOP=50 NEVAL=100 AXIS=TIME(LIN)
          CMT=1,3 MDV=101,101 FILE=example6b.csv
```

See Guide Introduction_7 for details and examples.

REFERENCES: Guide Introduction_7

FINISH

MEANING: FINISH record
CONTEXT: NM-TRAN and NONMEM data sets

USAGE:

```
FIN
```

DISCUSSION:
A FINISH record signals the end of the data for a given NONMEM problem. It is always optional.

FINISH record in NM-TRAN data sets:

When the RECORDS option of the $DATA record is omitted, NM-TRAN reads the data set to a FINISH record or to end-of-file, whichever comes first.

When a format is omitted on the $DATA record, a FINISH record consists of the characters FIN appearing anywhere in the record (the other characters are all blank).

When a format is provided on the $DATA record, a FINISH record must have the same form as in NONMEM data sets.

FINISH record in NONMEM data sets:

When Fields 3 and 6 of the NONMEM DATA Control Record are blank or zero, NONMEM reads the data set to a FINISH record or to end-of-file, whichever comes first. (Field 3 and 6 either contains the number of data records or are blank or zero. They must be blank or zero when the number of records is greater than 99999999.)

The format of the FINISH record is:

Blank characters in positions 1-76 and 80 and the characters F, I, and N in positions 77, 78, and 79, respectively.

If m is the number of FORTRAN records spanned per data record, there must be m-1 blank FORTRAN records inserted before the FINISH record.

In addition, the following three rules must be followed:

All FORTRAN records of the input data file must be 80 characters long.

All FORTRAN records of the input data file except the FINISH record must have blank characters in positions 77-80.

The total number of A, E, and F codes on the FORMAT record, including their multiplicities, must equal, not exceed, the number of data items specified on the DATA record.

REFERENCES: Guide II, section D.2.3
REFERENCES: Guide IV, section III.B.5

FLU

MEANING: FLU subroutine
CONTEXT: NONMEM user-installation routine

USAGE:

```
SUBROUTINE FLU (I)
RETURN
END
```

DISCUSSION:

Routine FLU is a user-installation routine. It allows the user to cause NONMEM to execute statements needed with certain operating systems.

FLU is called by NONMEM whenever the buffer for an I/O unit should be emptied. Buffer emptying is controlled in an operating system specific way, and often a compiler that is supported under an operating system will support a FORTRAN language extension that allows a buffer to be emptied. Whenever possible, such a FORTRAN instruction should be coded into FLU. FLU has one argument, I, which is the I/O logical unit number. The statement that should be inserted prior to `RETURN` follows, for two different operating systems, is as follows:

(Sun UNIX):

```
CALL FLUSH (I)
```

(MS WINDOWS 95):

```
CALL COMMITQQ (I)
```

REFERENCES: None.

FNLETA

MEANING: FNLETA subroutine
CONTEXT: NONMEM routine

DISCUSSION:
FNLETA is a NONMEM routine.

An error message from FNLETA indicates that NONMEM has experienced difficulties obtaining posthoc etas.

REFERENCES: None.

FORMAT

MEANING: Specifies the format for table files and other files
CONTEXT: Options of NM-TRAN control file records

USAGE:

FORMAT and related options may be used with the following control records:

$TABLE [FORMAT=s1PE11.4(default)][LFORMAT=s][RFORMAT=s][IDFORMAT=s]
Specifies the format for writing to the output table file.

$ESTIMATION [FORMAT=s1PE12.5 (default)] [DELIM=s]
Specifies the format for writing to the raw output files.

$COVAR [FORMAT=s1PE12.5(default)] [DELIM=s]
Specifies the format for writing to the raw output files

$CHAIN [FORMAT=s1PE12.5 (default)] [DELIM=s]
Specifies the format for writing to the raw output files

$RCOV [FORMAT=s] [DELIM=s]
Specifies the format for reading from the input file root.cov

$RCOVI [FORMAT=s] [DELIM=s]
Specifies the format for reading from the input file root.coi

$ETAS [FORMAT=s] [DELIM=s]
Specifies the format for reading from the input file root.eta

$PHIS [FORMAT=s] [DELIM=s]
Specifies the format for reading from the input file root.phi

DISCUSSION:

FORMAT=s1PE12.5

s defines the delimiter [,|s(pace)|t(ab)] followed by a Fortran format specification. The FORMAT option is case-insensitive.

Default for $TABLE: FORMAT=s1PE11.4
Default for $ESTIMATION, $COVAR, $CHAIN:
With versions of NONMEM 7.5 and higher: FORMAT=s1PE12.5
With versions of NONMEM before 7.5: FORMAT=s1PE11.4

As with previous versions:

The first character defines the delimiter. Delimiters are not a part of the Fortran language. The delimiter is stripped from the Fortran format specification, and then used by NONMEM to modify the file being generated. The delimiter may be s for space, t for tab, or the comma. A comma produces a comma delimited file with aligned fields (so, padded with spaces). The delimiters may be upper or lower case.

New to NONMEM 7.5:

There are two new delimiters. c for comma delimited file with no spaces, or q for comma delimited file with no spaces and double quotes around column names that have commas (such as "OMEGA(2,1)").
The list of delimiters is now as follows.

s means space
t means tab

, means comma
c means comma, and spaces removed
q means comma. Spaces removed, and quotes around entries that need them.

Two new formats are permitted, which provide both delimiter and number format:
FORMAT=QCSV is equivalent to FORMAT=q1PG23.16
FORMAT=CSV is equivalent to FORMAT=c1PG23.16

May be terminated with
line length (e.g., FORMAT=s1PE15.8:160),
line length with continuation marker for the end of each line (e.g., FORMAT=s1PE15.8:160c),
and continuation marker for start of each continued line (e.g. FORMAT=s1PE15.8:160cx).
If "c" is the letter s, it stands for "space".
If "c" is the character &, (e.g., FORMAT=s1PE15.8:160&) and & is the last character on the line, it must be followed by ";" (e.g., FORMAT=s1PE15.8:160&;)

If the user-defined format is inappropriate for a particular number, then NONMEM uses the default format for that number.

Most of the numeric formats are similar to those of Fortran, but there are exceptions.

The syntax for the number format is Fortran based, as follows:

For E field: xPEw.d indicates

w total characters to be occupied by the number (including decimal point, sign, digits, E specifier, and 2 digit magnitude),

d digits to the right of the decimal point, and x digits to the left of the decimal point.

Examples:

E12.5: -0.12345E+02
2PE13.6: -12.12345E+02

If you are outputting numbers that are less than 1.0E-99, such as 1.2236E-102, NONMEM changes how the number is displayed. With format 1PE12.4, Fortran would delete the "E" and display the number as 1.2236-102. NONMEM retains the "E" and displays one fewer significant digit to make room for the extra digit in the exponent. The final digit is truncated (no rounding), as follows:
1PE12.4: 1.223E-102
To make room for a three digit exponent, you may set the format as follows:

xPEw.dEe

where e is the number of digits to be provided for the exponent. For example

1PE12.4E3: 1.2236E-102
Another example of this format:
1PE12.4E3: -2.3456E+002

For F field: Fw.d

indicates

w total characters to be occupied by the number (including decimal point, sign and digits),

d digits to the right of the decimal point.

Examples:

F10.3: -0.012
F10.3: 234567.123

For G field: xPGw.d

For numbers >=0.1, will print an F field number if the value fits into w places showing d digits, otherwise will resort to xPEw.d format.

For numbers <0.1, will always use xPEw.d format.

IDFORMAT=s1PE11.4 (NM75)

This specifies the format for the ID column. By default the ID column has the same format as specified by FORMAT. However, sometimes you wish the ID to appear as an integer, in which case, you may set IDFORMAT as I. Do not include the delimiter.
Some examples:

IDFORMAT=I Integer value, left adjusted in the field.

IDFORMAT=I6 Integer value, right adjusted in the first 6 characters of the field

IDFORMAT=F6.1 Floating value, with single digit to the right of the decimal.

If an improper format is given, it defaults to that of FORMAT.

LFORMAT=*s* RFORMAT=*s*

An alternative format description to FORMAT is RFORMAT and LFORMAT.

(where R=real numbers) describes the full numeric record of a table, so that formats for specific columns may be specified.

LFORMAT (where L=label) specifies the format of the full label record of a table.

The formats must be enclosed in double quotes, and (), and have valid Fortran format specifiers. The RFORMAT and LFORMAT options can be repeated if the format specification is longer than 80 characters. Multiple RFORMAT and LFORMAT entries will be concatenated to form a single format record specification.
For example,

```
LFORMAT="(4X,A4,4(',',4X,A8))"
RFORMAT="(F8.0,"
RFORMAT="4(',',1PE12.5))"
```

Will result in the following formats submitted to a Fortran write statement:
`LFORMAT=(4X,A4,4(',',4X,A8))`
for the table's label record, and `RFORMAT=(F8.0,4(',',1PE12.5))`
For the table's numeric records.

If RFORMAT and LFORMAT are given, then the FORMAT option will be ignored. By default, FORMAT, RFORMAT, LFORMAT specifications will be passed on to the next $TABLE record in a given problem unless new ones are given. To turn off an RFORMAT/LFORMAT specification in a subsequent table (and therefore use FORMAT instead), set

```
LFORMAT="NONE"
RFORMAT="NONE"
```

Here is an example of $TABLE statements designated in a control stream file:

```
$TABLE ID TIME PRED RES WRES CPRED CWRES EPRED ERES EWRES NOAPPEND ONEHEA
FILE=tabstuff.TAB NOPRINT,FORMAT=,1PE15.8
$TABLE ID CL V1 Q V2 FIRSTONLY NOAPPEND NOPRINT FILE=tabstuff.PAR
LFORMAT="(4X,A4,4(',',4X,A8))"
RFORMAT="(F8.0,"
RFORMAT="4(',',1PE12.5))"
```

```
$TABLE ID ETA1 ETA2 ETA3 ETA4 FIRSTONLY NOAPPEND NOPRINT
FILE=tabstuff.ETA,FORMAT=";F12.4"
LFORMAT="NONE"
RFORMAT="NONE"
```

There is no NMTRAN error checking on the RFORMAT and LFORMAT records, so the user must engage in trial and error to obtain a satisfactory table output (you should set MAXEVAL=0 or MAXEVAL=1 for the $EST step to do a quick check, so you don't spend hours on estimation only to find the RFORMAT/LFORMAT were not appropriate). The $MSFO option of the $ESTIMATION record should be used to produce a Model Specification File. Such a file can be used via $MSFI record.

A word of caution. The FORMAT descriptor 1P, which means move the decimal point to the left by 1, will be in effect for all remaining FORMAT components. For example, in

```
RFORMAT="(F8.0,37(',',1PE13.6),24(',',F7.2))"
```

the F field format that follows an E field format, in which 1P was used, will also have the decimal placed to the left, and a 1.00 would appear as a 10.00. To prevent this from occurring, revert to no decimal shift with 0P:

```
RFORMAT="(F8.0,37(',',1PE13.6),24(',',0PF7.2))"
```

See **INTRODUCTION TO NONMEM 7, $EST: Format of Raw Output File**
See **INTRODUCTION TO NONMEM 7, IDFORMAT= I**
See **INTRODUCTION TO NONMEM 7, LFORMAT, RFORMAT**

REFERENCES: Guide Introduction_7

F FLAG

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPR_INT, ONLY: F_FLAG=>IPRDFLG1
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: IPRDFLG1
```

DISCUSSION:

When either of the options `LIKELIHOOD` or `-2LOGLIKIHOOD` appear in the $ESTIMATION record, this indicates that for all observations, the variable Y (with NM-TRAN abbreviated code) or F (with a user-supplied PRED or ERROR routine) is being set to a likelihood (or -2 log likelihood) value for the observation. Suppose, though, that neither option appears. Then for a given observation, the default indication is that Y or F is being set to a "prediction" of the observation. This is equivalent to setting the variable F_FLAG to its default value of 0. When, however, the variable F_FLAG is set to the value 1 (or 2) in the PRED or ERROR routine, this signals that Y or F is being set to a likelihood (or -2 log likelihood) value for this particular observation.

F_FLAG may be set in NM-TRAN abbreviated code.

```
IF (TYPE.EQ.1) THEN
   Y=THETA(1)+ETA(1)+ERR(1) ; a prediction
ELSE
   F_FLAG=1
   A=EXP(THETA(2)+ETA(2))
   B=1+A
   Y=DV*A/B+(1-DV)*1/B      ; a likelihood
ENDIF
```

A nonzero value of F_FLAG has no effect during a Simulation Step. But note: When single-subject data are simulated for which, when the data are later analyzed, Y (F) would always be set to a likelihood, and when the ONLYSIMULATION option is used in the $SIMULATION record, then usually the NOPREDICTION option can also be used. When the NOPREDICTION option is used, if any eta's are used in the model, the data are regarded as population data, but the model for the data in question usually does not involve eta variables. When, however, single-subject data are simulated for which, when the data are analyzed, Y (F) would sometimes be set to a likelihood and sometimes to a "prediction", and when the ONLYSIMULATION option is used in the $SIMULATION record, then the PREDICTION option will need to be used (this is the default option!). The reason is that the expression of residual variability associated with the predictions will involve eta variables.

RESTRICTIONS:

When F_FLAG is set to a non-zero-value for some observation, neither the `LIKELIHOOD` or `-2LOGLIKIHOOD` option may be used.

When F_FLAG is set to a nonzero-value for an observation, the Laplacian estimation method must be used.

Unless with all observations within an individual record, F_FLAG is 0, the RES and WRES items will be 0 for all data records in the individual record.

When F_FLAG is set to a nonzero-value for an observation within an individual record, inter-L2 correlated epsilons are not allowed with the observations within the individual record.
(See **Correlation Across L2 Records**)

If the data are population data, a nonzero value of F_FLAG cannot be used within a multiple-observation L2 record. If the data are single-subject data, a nonzero value of F_FLAG cannot be used within a multiple-observation L1 record.

Location prior to NONMEM 7: nmpr17

REFERENCES: None.

GAMLN

MEANING: GAMLN function
CONTEXT: NONMEM utility routine

USAGE:

```
USE SIZES, ONLY: DPSIZE
REAL(KIND=DPSIZE) :: GAMLN,X
REAL(KIND=DPSIZE) :: LN_OF_GAMMA
 .  . .
LN_OF_GAMMA= GAMLN(X)
```

DISCUSSION:
Gives the natural logarithm of the gamma function. It is more accurate than the Stirling's approximation.

May be used in abbreviated code. May be used in the computation of the objective function because it is continuous. The first derivative (GAMLND1) and second derivative (GAMLND2) are also evaluated, and are continuous for X>0.0.

Input argument:

X

EXAMPLE OF USAGE:

FAC=EXP(GAMLN(X+1.))

If X takes only integer values, then

FAC=X!=X*(X-1)*(X-2)... *1

The GAMMA function in effect "fills in" the values of the factorial (but with a shift of 1 in the argument) for non-integer values of x.

If the argument is a random variable, it should always be positive. Eta partial derivatives are computed.

REFERENCES: None.

GETETA

MEANING: GETETA subroutine
CONTEXT: NONMEM utility routine

USAGE:

```
DIMENSION ETA(LVR)
DOUBLE PRECISION ETA
CALL GETETA (ETA)
```

DISCUSSION:
The NONMEM utility routine GETETA is used to obtain subject-specific values of eta. It may be called only when the data are population data.

Output argument:

`ETA`
: An array into which GETETA stores subject-specific eta values ETA(1), ETA(2), etc. If a conditional estimation method is not used, GETETA stores zeros in the ETA array.

GETETA must always be initialized at ICALL=1. This simply involves calling GETETA at ICALL=1. No values are stored in ETA at this call.

When the data are population and abbreviated code is used, values of etas are always obtained by a call to GETETA occurring in generated or Library PK or PRED subroutines. (In the case of PK, the array of etas is stored in a common and is available to the ERROR subroutine as well.) This call occurs even if a conditional estimation method is not implemented, in which case GETETA stores zeros in ETA. As a result, when the first-order method is used, apparent subject-specific values computed in NM-TRAN abbreviated code are the required typical values. Thus, an executable resulting from using an abbreviated code for PRED (for PK and ERROR if PREDPP is used) can be used with any estimation method.

EXAMPLE OF USAGE:

```
IF (ICALL.EQ.1) THEN
 ...
CALL GETETA (ETA)
 ...
RETURN
ENDIF
 ...
IF (ICALL.EQ.2) CALL GETETA (ETA)
```

It is also possible to obtain simulated subject-specific values of eta. (See **simeta**).

REFERENCES: Guide IV, section V.C.5
REFERENCES: Guide VI, section III.E.2 , IV.B.2

HESSIAN

MEANING: A matrix computed by NONMEM
CONTEXT: NONMEM output

DISCUSSION:

The Hessian matrix is the second derivative matrix of the objective function with respect to the parameters of the model. NONMEM computes Hessian matrices for several purposes.

Estimation Step

A rough numerical approximation to the Hessian is maintained during the minimization search. The following message in the intermediate output refers to this matrix:

```
RESET HESSIAN
```

Explanation: The search starts with the Hessian set to the identity matrix. The Hessian is updated after each iteration, using a rank 1 update procedure. When there is no longer a sensible direction to take, but convergence has not been achieved, this may be due to inadequacy of the updated Hessian. Then the Hessian is reset to a certain positive semi-definite diagonal matrix, and a new direction computed from this matrix (and the gradient vector) is tried.

The appearance of the message indicates why an unusually large number of function evaluations were used for the iteration (extra ones were needed to compute the new Hessian and perform a line search along the new direction) and why an unusually long CPU time was needed for the iteration (if intermediate output is being monitored). Its appearance suggests that the search is not going easily, and perhaps (not necessarily) something is wrong.

Covariance Step

A good numerical approximation to the Hessian is computed at the final parameter estimates. It is referred to as the R matrix.

Obtaining Conditional Estimates

In this case, the Hessian matrix is the second derivative matrix of the conditional objective function. The following error messages refer to this matrix:

```
HESSIAN OF OBJ. FUNCT. FOR COMPUTING CONDITIONAL ESTIMATE IS
NON POSITIVE DEFINITE
```

```
NUMERICAL HESSIAN OF OBJ. FUNC. ...
```

REFERENCES: Guide IV, section III.B.15
REFERENCES: Guide V, section 12.4.6
REFERENCES: Guide VII, section II.A

ICALL

MEANING: Argument of user-supplied subroutines
CONTEXT: User-supplied routines

DISCUSSION:
ICALL is an argument passed by NONMEM to user-supplied subroutines CCONTR, CONTR, CRIT, PRED, PRIOR and MIX.

PREDPP selectively passes the same argument to PK, ERROR, and INFN. ICALL can be used in $PK, $ERROR, and $PRED abbreviated code. The discussion below describes the values of ICALL as seen by PRED.

`ICALL=-1`: the routine has been called for the PRED_IGNORE_DATA feature of NONMEM 7.5. One call per data record, at the start of the run. These calls occur only if abbreviated code uses variables PRED_IGNORE_DATA or PRED_IGNORE_DATA _TEST, or if the PRED_IGNORE_DATA option of $DATA is used. Otherwise, there are no calls to PRED with ICALL=-1.

ICALL=0: the routine has been called for initialization at the beginning of the NONMEM run; one such call per run.

ICALL=1: the routine has been called for initialization at the beginning of a NONMEM problem; one such call per problem.

ICALL=2: the routine has been called for a prediction. Multiple calls occur.

ICALL=3: the routine has been called for finalization at the end of a NONMEM problem; one such call per problem.

ICALL=4: the routine has been called during the Simulation Step; multiple calls occur, as with ICALL=2.

ICALL=5: the routine has been called when expectations are being computed; multiple calls occur.

ICALL=6: the routine has been called when raw data averages are being computed; multiple calls occur.

Note: Some subroutines are called with only a subset of the possible values of ICALL.

REFERENCES: Guide I, section C,4.2
REFERENCES: Guide IV, section IV.D , IV.E.1
REFERENCES: Guide VI, section III.C , IV.B.1

ID .ID. AND L1 DATA ITEMS

MEANING: Identification (ID) data item for NONMEM
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... ID ...`

DISCUSSION:
`ID`, `.ID.` and `L1` are labels for NONMEM's identification data item.

`ID` The observations in the data set are divided into groups called level-one observations. These groups may have different sizes. Level-one observations are treated as being statistically independent multivariate observations.

Because there can be at most one scalar observation on any data record, the identification (ID) data item is used to group together the data records containing the scalar observations comprising a level-one observation. Using NM-TRAN, the identification data item is given the label ID in the $INPUT record. A contiguous set of data records with the same value of ID are collectively called an individual record (also called an L1 record), and this record "contains" one level-one observation. A change in ID value indicates the start of a new individual record.

When every level-one observation consists of a single scalar observation, so that the entire level-one observation can be contained in a single data record, ID data items are not generally required. However, they are always required when the data are population data and PREDPP is used.

`.ID.`
When the data are single-subject data and PREDPP is used, NM-TRAN automatically generates the ID data items and provides the label `.ID.` for the NONMEM control stream. (Or if L2 data items appear, these are taken to be the ID data items.) If the ID label is given in the $INPUT record, NM-TRAN ignores it, and it also ignores the data items associated with this label.

`L1` Using NM-TRAN, the identification data item may be given the label L1 in the $INPUT record. Use of L1 as a label or synonym (e.g. ID=L1) in the $INPUT record specifies that the associated data items are to be used as the identification data items and, if the data are single-subject data, the automatic generation of the identification data items is suppressed.

REFERENCES: Guide I, section B.1
REFERENCES: Guide IV, section II.C.4 , III.B.2

II DATA ITEM

MEANING: Interdose Interval (II) data item for PREDPP
CONTEXT: $INPUT record and NONMEM data set

USAGE:
```
$INPUT ... II ...
```

DISCUSSION:
`II` labels PREDPP's interdose interval (II) data item. The interdose interval data item is optional. In a dose or reset-dose event record it gives the time between implied or additional doses (i.e., for repeated doses with a given period, the II data item is this period).

For a steady-state infusion (AMT=0; RATE>0), II should be 0.

For other steady-state doses, II should be a positive number. II gives the time between implied doses.

For non-steady-state doses, II should be a positive number if and only if the ADDL data item is a positive number. II gives the time between additional doses.

For observation, other-type, and reset event records, II should be zero.

The units of II should be consistent with those of TIME.

Any II value in the NM-TRAN data set that contains a colon (:) is assumed to be a clock time (hh:min) and NM-TRAN converts it (to hh.fr) as in the first example:

NM-TRAN data set	NONMEM data set
II	II
----	----
`1:30`	`1.50`
`1.30`	`1.30`

The $DATA record may include the option `TRANSLATE=(II/24)`, in which case all interdose interval data items (obtained after clock time conversion) are divided by 24. (See **$data**).

(See **exogenous supplementation example**).

REFERENCES: Guide VI, section V.G
REFERENCES: Guide IV, section V.B

INCLUDE

MEANING: Causes NM-TRAN to read control stream records from a different file.
CONTEXT: NM-TRAN Control Record

USAGE:
`INCLUDE` filename [*n*]

SAMPLE:
```
INCLUDE datadef
```

DISCUSSION:

This record causes NM-TRAN to read control stream records from a different file. The record must start with the characters `INCLUDE` or `include`. `$INCLUDE` and `$include` are also recognized.

OPTIONS:

`filename`
: Name of the file to be read. Required. Filename may not contain embedded spaces. If it contains commas, semicolons, or parentheses, then it must be surrounded by single quotes ' or double quotes ". Filename may also contain equal signs if it is enclosed in quotes.

n
: Optional. The number of copies of the file. Default is 1.

NMTRAN opens the named file and reads it to end-of-file. The contents of the file may be any portion of an NM-TRAN control stream, e.g., control records and/or abbreviated code. After reaching end-of-file, if the number of copies is greater than 1, NM-TRAN rewinds the file and re-reads it the specified number of times. After reaching end-of-file on the final (or only) copy, NMTRAN resumes reading the original control stream after the include record.

There may be more than one include record anywhere in the control stream, but they may not be nested. That is, an included file may not contain include records.

If an error is detected in the NMTRAN input, the line number displayed is cumulative. The number of the line in the included file is added to the number of lines already read from the control stream and any previous include files, as if the included file(s) are physically part of the control stream.

EXAMPLE:

```
$PROBLEM model A with data set 3
include data3def
include modela
$THETA 1.3 4
$OMEGA .04
$SIGMA 1
$ESTIMATION
```

The file data3def contains the $INPUT and $DATA statements. The file modela contains the $SUBROUTINE statement, $PK block, and $ERROR block. one

Verbatim comment lines may be present in an included file. If the character that immediately follows the initial " is !, this conforms to the FORTRAN 90 syntax for comment lines. The line is copied unchanged. If the character that immediately follows the initial " is C or c or " or *, this conforms to the FORTRAN 77 syntax for comment lines. The

line is copied unchanged, but C or c or " or * is replaced by !.
REFERENCES: None.

INDICATOR VARIABLES

MEANING: Coding technique
CONTEXT: Abbreviated code

DISCUSSION:
An indicator variable is a variable whose value is 0 or 1. It may be identified with an input data item, or it may be a variable defined in the abbreviated code. It is used to make a choice in a computation.

Suppose, for example, ICU is a variable which is either 0 or 1. The following code can be used:

```
IF (ICU.EQ.0) THEN
CLM=TVCLM+ETA(1)
ELSE
CLM=TVCLM+ETA(2)
ENDIF
```

This can be coded unconditionally using an indicator variable:

```
CLM=TVCLM+(1-ICU)*ETA(1)+ICU*ETA(2)
```

Unconditional code is preferred when MU variables are computed.

In Guide V, Chapter 12, appears this example involving observations of two types, CP and effect. The latter is modeled by the Emax model. Observations of both types are recorded in the DV data item. A data item called TYPE with values 1 and 2 is used to distinguish between them. This data item may be used to compute the appropriate prediction in the $ERROR block, as follows:

```
$ERROR
 EMAX=THETA(5)+ETA(3)
 C50=THETA(6)+ETA(4)
 EFF=EMAX*F/(C50+F)
 Y1=EFF+ERR(1)
 Y2=F+ERR(2)
 IF (TYPE.EQ.2) THEN
 Y=Y2
 ELSE
 Y=Y1
 ENDIF
```

The last five lines of this example can also be coded as follows:

```
 Q=1
 IF (TYPE.EQ.2) Q=0
 Y=Q*Y1+(1-Q)*Y2
```

A more general technique is to use two indicator variables. This technique can easily be extended to the case of three or more choices.

```
$ERROR
 EMAX=THETA(5)+ETA(3)
 C50=THETA(6)+ETA(4)
 EFF=EMAX*F/(C50+F)
 Y1=EFF+ERR(1)
 Y2=F+ERR(2)
```

```
Q1=0
Q2=0
IF (TYPE.EQ.1) Q1=1
IF (TYPE.EQ.2) Q2=1
Y=Q1*Y1+Q2*Y2
```

REFERENCES: Guide V, section 7.5.3 , 7.5.4 , 12.5

INDIVIDUALS

MEANING: A data analysis unit
CONTEXT: NONMEM input/output

DISCUSSION:
Data to be analyzed with NONMEM are often population data, by which is meant multiple data arising from each of a number of individual units. Individuals are typically persons, but they may be any other appropriate units, such as families, geographic localities, etc. Data are regarded as being statistically independent from unit to unit.

With NONMEM, there are two nested levels of random effects, The first level applies to individuals; different individuals are regarded as having different realizations of level-one random effects. A second level of random effects applies to the observations from each individual; different (univariate) observations are regarded as having different realizations of level-two randoms effects, but the same realization of level-one random effects.

The data from an individual is given in the data set by a contiguous group of data records, with one observation on each data record, and all data records having the same identification (ID) data item. This group of data records is called the individual record, or level-one (L1) record. (Do not confuse a *data record*, which is one record in a data file, with an *individual record*, which is a group of data records.)

Data to be analyzed may be single-subject data. These are data that appear to require at most one level of random effects. (In fact, there are population data which require only one level of the two NONMEM levels of random effects, along with a second level of random effects which may be expressed in a way that is transparent to NONMEM. This type of situation is communicated in such a way that NONMEM does not mistake these data for single-subject data.) Such data may actually arise physically from different individual units, or individuals. Indeed, when they do, they may even be comprised of multiple data from different units, e.g. pairs of plasma and saliva concentrations obtained at the same time point, each from a number of different subjects. However, if as with this example, only one level of random effects is needed, these data are nonetheless considered to be single-subject data. The data are regarded as being statistically independent from unit to unit. When single-subject data indeed arise physically from the same subject, the data can also be grouped into individual units such that the data are regarded as being statistically independent from unit to unit. These units are also called "individuals". As an example, there may be pairs of plasma and saliva concentrations from the same subject. More precisely, NM-TRAN recognizes population data to be data that do not qualify as single-subject data.

NONMEM counts the number of distinct individuals in the data set, and reports this count as a check.

E.g.,
```
TOT. NO. OF INDIVIDUALS:  166
```

NONMEM 7 also reports how the data is to be analyzed:

```
ANALYSIS TYPE: SINGLE-SUBJECT
ANALYSIS TYPE: POPULATION
ANALYSIS TYPE: POPULATION WITH UNCONSTRAINTED ETAS
```

Population analysis with unconstrained etas is new with NONMEM 7, and can be used to analyze a population data set as separate individuals. OMEGA diagonal values are fixed

to a special value 1.0E+06.

See Guide Introduction_7 "Analyzing Single-subject data as Population with Unconstrained etas".
(See **recid2.exa**).

REFERENCES: Guide I, section E
REFERENCES: Guide Introduction_7

INFERENTIAL STATISTICS

MEANING: Inferential statistics
CONTEXT: NONMEM output

DISCUSSION:
NONMEM provides estimates of the precision of its parameter estimates.

Confidence intervals for parameters, hypothesis tests on parameter values, and model-building all require determination of plausible bounds on parameters. These can be assigned if the distribution of the estimates (upon imagined replication of the data set) can be estimated. Asymptotic statistical theory applied to extended least-squares estimation (as used in NONMEM) says that the distribution of the parameter estimators is multivariate normal, with variance-covariance matrix that can be estimated from the data. NONMEM supplies such an estimate of the variance-covariance matrix of the parameter estimates (See **covariance matrix of estimate**). The square roots of the diagonal elements of that matrix are the estimated standard errors of the corresponding parameter estimators (See **standard error of estimate**), while the off-diagonal elements can be transformed into correlation coefficients, and are presented in the correlation matrix of the parameter estimates (See **correlation matrix of estimate**). The inverse of the covariance matrix and the eigenvalues of the correlation matrix are also available (See **$covariance**).

REFERENCES: Guide V, section 5.4

INFN

MEANING: INFN subroutine
CONTEXT: User-supplied subroutine; replaces a PREDPP dummy routine.

USAGE:

```
SUBROUTINE INFN (ICALL,THETA,DATREC,INDXS,NEWIND)
USE SIZES, ONLY: ISIZE,DPSIZE
USE NMPRD_INT, ONLY: NWIND
INTEGER(KIND=ISIZE) :: ICALL,INDXS(*),NEWIND
REAL(KIND=DPSIZE)   :: THETA(*),DATREC(*)
```

DISCUSSION:
A run consists of one or more problems, and each problem consists of one or more subproblems. With NONMEM 75, there is an opportunity to drop data records from the NONMEM data set at the start of the run, using the PRED_IGNORE_DATA feature. There are opportunities to make some rudimentary computations at the beginning of a run (run initialization), at the beginning of a problem (problem initialization), at the end of a subproblem (subproblem finalization), at the end of a problem (problem finalization), and at the end of a run (run finalization). For example, at problem initialization, data transgeneration may take place, or a variable that will be modified at subproblem finalization, may be initialized. At problem finalization, the value of this variable may be written to a user file. There is no opportunity to do subproblem initialization.

The INFN subroutine is called by PREDPP to make initialization and finalization computations. The version distributed with PREDPP is a "stub" or "dummy" routine that does nothing. It may be replaced by a user-written code.

Input/Output argument:

ICALL

ICALL=-1: INFN may set PRED_IGNORE_DATA=1 for records to be dropped. The following is required:

```
USE NMPRD_INT, ONLY: PRED_IGNORE_DATA,PRED_IGNORE_DATA_TEST
```

The option $DATA ... PRED_IGNORE_DATA is required to cause NONMEM to make a PRED_IGNORE_DATA pass.

ICALL=0: INFN may now make computations for run initialization. ICALL may be reset by INFN to a number in the range 1-8999. This number will appear on NONMEM output, allowing the user to identify the INFN routine being used.

ICALL=1: INFN may now make computations for problem initialization.

ICALL=3: INFN may now make computations for subproblem finalization.

ICALL=3: INFN may now make computations for problem finalization. If there are subproblems, first do subproblem finalization, if required. Then test for IREP=NREP (the number of the current subproblem equals the total number of subproblems), and if true, do problem finalization. Values of IREP and NREP may be found in NONMEM modules
(See **Simulation: NREP,IREP**).

ICALL=3: INFN may now make computations for run finalization. First do problem finalization, if required. Then test for IPROB=NPROB (the number of the current problem equals the total number of problems), and if true, do run finalization.

Values of IPROB and NPROB may be found in NONMEM modules
(See **Problem_Iteration_Counters**).

Input arguments:

DATREC

DATREC contains the first data record of the current problem. (At ICALL=0, the current problem is the first problem.) Using PASS (see below), the contents of DATREC are replaced by other data records for the current problem, allowing all these other records to be read and even modified.

THETA

The NONMEM THETA vector.
At ICALL=0, THETA contains the initial estimates for the first problem.
At ICALL=1, THETA contains the initial estimates for the current problem.
At ICALL=3, THETA contains the final estimates for the current problem.

INDXS

The values specified in the $INDEX record of the NM-TRAN control stream. (This is the NONMEM INDXS array starting at position 12, the first position beyond those positions used by PREDPP itself.)

NWIND

NWIND has value 0 when INFN is called. It changes value during a pass through the data using PASS (see below).
NWIND=0: First record of the data set.
NWIND=1: First record of a subsequent individual record.
NWIND=2: Subsequent data record of an individual record.

EXAMPLES OF USAGE:

Initialization

Constants in the PK routine can be set if they are stored in a module or common block declared both in INFN and PK.

Transgeneration

The data can be accessed and even modified via use of the NONMEM utility routine PASS in routine INFN (See **pass**). As the data records are passed one-by-one to INFN, each record is stored in turn in DATREC.

Data can be transgenerated and additional data items can be produced at both initialization and finalization. Data items that appear in a table or scatterplot with a given subproblem or problem are produced or left unchanged at the subproblem or problem finalization.

At finalization estimates of the eta's can be obtained via the NONMEM utility routine GETETA. When used in conjunction with PASS, the values returned for the eta's with each call to GETETA are appropriate for the individual whose data record is currently in DATREC.

Interpolation

As an example of transgeneration, interpolated values of a covariate can be computed for event records in which values are missing, e.g. for other-type event records that have been included so that predictions can be obtained at the times in these records. This could also be done in PK or ERROR, but then this would be done with every call to these routines; if done in INFN, the computation is done once only.

(See **Infn interpolation example**).

REFERENCES: Guide VI, section VI.A , Figure 37

INFN-DEFINED VARIABLES

MEANING: INFN-defined global variables
CONTEXT: User-supplied routines

USAGE:

```
MODULE INFNP
USE PRINFN, ONLY: TLCOM=>ITV
END MODULE INFNP
```

GLOBAL DECLARATION:

```
MODULE PRINFN
USE SIZES, ONLY : DPSIZE,DIMTMP
IMPLICIT NONE
SAVE
REAL(KIND=DPSIZE), TARGET, DIMENSION (DIMTMP):: ITV
END MODULE PRINFN
```

DISCUSSION:

PRINFN is a module for INFN-defined variables. It contains an array ITV. (TLCOM is suggested above as an alternate to the name ITV to be consistent with generated code, but this is optional.) Variables are stored in PRINFN for communication with other other blocks of abbreviated code or with user-written codes. These are variables defined first in $INFN or in $PK-INFN or $ERROR-INFN blocks. Module PRINFN is declared in all generated routines PK, ERROR, etc., and the variables can be used as right-hand quantities in these routines. INFN-defined variables are not initialized or modified by NONMEM or PREDPP. Hence the variables in the module may be initialized or modified at ICALL values 0 or 1 or 3, and will retain whatever values they are given. These variables cannot be displayed in tables or scatterplots. If they are to be displayed, `WRITE` statements can be used, or their values can be assigned to variables that are listed in module NMPRD4. (See **PRED-defined_variables**).

Variables from PRINFN can be used in a user-written code. MODULE INFNP must be defined as above prior to the SUBROUTINE statement. The variables have names TLCOM(1), TLCOM(2), etc.
E.g., `X=TLCOM(1)/TLCOM(2).` It is possible to give the variables the same (non-subscripted) names that they had in $INFN. Suppose variables X1 and X2 are used in $INFN, and the following is present in the generated INFN routine:

X1=>TLCOM(00001); X2=>TLCOM(00002)

In the user-written code, after the last declaraction, include:

POINTER :: X1,X2

Prior to the first executable statement, include:

X1=>TLCOM(1); X2=>TLCOM(2)

Now the code is `X=X1/X2`.

Code generated by NM-TRAN is somewhat more complex, but achieves the same result.

REFERENCES: None.

INITIAL ESTIMATE (INITIAL EST)

MEANING: Initial estimates of parameters THETA, OMEGA, SIGMA
CONTEXT: NONMEM output

DISCUSSION:
Initial estimates of parameters are required by NONMEM. They are used, for example, to compute the initial value of the objective function (before NONMEM begins its search for final estimates). The user may provide initial estimates. Alternatively, he may request that NONMEM provide initial estimates.

(NONMEM may also use as initial estimates the final estimates from a previous run, by means of a Model Specification File; (See **model specification file**). When an MSFI is used, the outputs described here do not appear.)

NONMEM output reflects initial estimates provided by the user on its problem summary page as in this example:

```
INITIAL ESTIMATE OF THETA:
LOWER BOUND    INITIAL EST    UPPER BOUND
0.0000E+00     0.1500E+03     0.1000E+07
0.0000E+00     0.0000E+00     0.0000E+00
0.1000E+00     0.3000E+00     0.1000E+02
-0.1000E+07    -0.1000E-01    0.1000E+07
INITIAL ESTIMATE OF OMEGA:
0.2500E+00
0.0000E+00   0.2500E+00
INITIAL ESTIMATE OF SIGMA:
0.2500E+00
```

The above states that:

1) The initial estimate of theta(1) is 150, and the parameter is constrained to be greater than zero (no upper bound).
2) Theta(1) is fixed at zero.
3) The initial estimate of theta(3) is .3, and the parameter is constrained to lie between .1 and 10.
4) The initial estimate of theta(4) is -.01, and the parameter is unconstrained.
5) OMEGA is 2x2 and constrained to be diagonal (signified by the 0 value of the initial estimate of omega(2,1); the initial estimates of omega(1,1) and omega(2,2) are .25.
6) SIGMA is 1x1 (a scalar), with initial estimate .25.

If the initial estimate of an element of THETA is left blank, this indicates that the user has requested that NONMEM provide the initial estimate, as in this example:

```
INITIAL ESTIMATE OF THETA:
LOWER BOUND    INITIAL EST    UPPER BOUND
0.1000E+00                    0.1000E+02
```

Similarly, if the initial estimate of all or part of OMEGA and/or SIGMA is not printed, this indicates that the user has requested that NONMEM provide the missing initial

estimate(s).

NONMEM performs a search for missing initial estimates as its first task. With NONMEM 7.4, when initial thetas are to be estimated, evaluations can now be done for FOCE and Laplace, not just for FO. The complete set of initial estimates (including those provided by the user and those provided by NONMEM) are printed on a separate page under the heading "INITIAL PARAMETER ESTIMATE", as in the following example. The format is identical to that of the "FINAL PARAMETER ESTIMATE" page. If all initial parameter estimates are provided by the user, this page does not appear.

```
********************  INITIAL PARAMETER ESTIMATE   ***
THETA - VECTOR OF FIXED EFFECTS PARAMETERS  **********
           TH 1      TH 2      TH 3
         1.50E+00  1.02E-01  2.90E+01
OMEGA - COV MATRIX FOR RANDOM EFFECTS - ETAS  ********
           ETA1
ETA1     1.50E+00
```

This example is from a run involving single-subject data, so there is no SIGMA matrix.

(See **final parameter estimate**).

REFERENCES: Guide IV, section III.B.9 , III.B.10 , III.B.11

INITIAL STEADY STATE: I SS,ISSMOD

MEANING: PREDPP-PK global variables
CONTEXT: PK routine

USAGE:
```
USE PRMOD_INT, ONLY: I_SS,ISSMOD
```

GLOBAL DECLARATION:
```
INTEGER(KIND=ISIZE) :: I_SS,ISSMOD
```

DISCUSSION:

Used for the Initial Steady State feature of PREDPP, with the general non-linear models (ADVAN6, ADVAN8, ADVAN9, ADVAN13, ADVAN14, ADVAN15, ADVAN16, ADVAN17, ADVAN18).

By default, the initial conditions (i.e., compartment amounts) are zero at the start of each individual record. Different initial conditions may be computed using the I_SS (Initial Steady State) feature of MODEL and/or PK.

ISSMOD may be set by MODEL.
(Note that the option of the $MODEL record is `I_SS`, not `ISSMOD`.)
Default: -1 (MODEL does not request the I_SS feature)
Values 0, 1, 2 and 3 are permitted. Value 0 requests that no steady state be computed. ISSMOD values 1, 2, or 3 requests that PREDPP compute an initial steady state for the model before the first event record of an individual record, or after a reset event. The results are identical to those that would be computed by a steady-state dose event record with SS=ISSMOD and AMT=0 and RATE=0. If endogenous drug is specified in the differential equations, non-zero initial conditions will be computed.

I_SS may be set by PK.
Default: -1 (PK does not request the I_SS feature)
Values are the same as ISSMOD, with the same effect. This allows I_SS to be set conditionally, e.g., if some subjects are at steady-state and others are not. If both ISSMOD and I_SS are set, then the value of I_SS overrides that of ISSMOD.

There is no difference between values 1, 2 and 3 of I_SS unless the PK routine also uses the compartment initialization feature `A_0`. The I_SS feature behaves exactly like a steady state dose record in this regard. Specifically,

> With I_SS=1 ("reset"), values of A_0 are ignored.
> With I_SS=2 ("sum"), values of A_0 are added to the SS values.
> With I_SS=3 ("initial ests"), values of A_0 are used as initial estimates when computing the SS values.

(and similarly for ISSMOD).

If ISSMOD is set, or I_SS is set at ICALL=1, PREDPP will acknowlege the fact in the NONMEM output.

When I_SS or ISSMOD is set, even if only to 0, then SSS and the appropriate SS routine are included in the NONMEM executable.

When I_SS or ISSMOD is set, even if only to 0, or verbatim code is present in the $PK or $INFN block, then NM-TRAN includes I_SS,ISSMOD in the PK and/or INFN routine.

Example: $PK may include code such as
```
  IF (TYPE.EQ.1) THEN
```

```
I_SS=1
ELSE
I_SS=0
ENDIF
```

(See **$model**, **$pk**, **model**, **pk**).

Location prior to NONMEM 7: prdpk4

REFERENCES: None.

INITIALIZATION-FINALIZATION BLOCK

MEANING: Abbreviated code for initialization and finalization
CONTEXT: $PRED, $PK, $ERROR, $INFN abbreviated code

SAMPLE:

```
$PRED
     IF (ICALL.EQ.1) CALL SUPP(0,1)
```

DISCUSSION:

A run consists of one or more problems, and each problem consists of one or more subproblems, which are iterations of various tasks specified in the problem. The "end" of a subproblem refers to the end of such an iteration. Problems themselves may be organized into superproblems, which may be iterated. The "beginning" and "end" of a superproblem refers to the beginning and end of the first and last iterations of the superproblem. There are opportunities to make some rudimentary computations at:

the beginning of a run (run initialization)
the beginning of a superproblem (superproblem initialization)
the beginning of a problem (problem initialization)
the end of a subproblem (subproblem finalization)
the end of a problem (problem finalization)
the end of a superproblem (superproblem finalization)
the end of a run (run finalization)

For example, data transgeneration may take place at problem initialization. Or, a variable may be initialized at problem initialization and modified at each subproblem finalization, and its final value written to a user file at problem finalization. There is no opportunity to do subproblem initialization. When using abbreviated code, initialization and finalization opportunities are signalled by values of the variable ICALL:

ICALL=0
Run initialization.

ICALL=1
Superproblem and Problem initialization.
ICALL is set to 1 to signal problem initialization. When this happens:

If a superproblem requires initialization, test
(i) S1IT (number of current superproblem iteration) equals 1 (or S2IT=1),
(ii) S1NUM (number of current superproblem) equals appropriate value, and
(iii) IPROB (number of current problem) equals number of first problem in superproblem,
and if (i)-(iii) are true, do superproblem initialization
(See **Problem_Iteration_Counters**).

ICALL=3
Subproblem, Problem, Superproblem and Run finalization.
ICALL is set to 3 to signal problem finalization. When this happens:

If a problem with subproblems requires finalization, test

IREP=NREP (number of the current subproblem equals total number of subproblems) and if true, do problem finalization
(See Simulation: NREP,IREP).

If a superproblem requires finalization, test
(i) S1IT=S1NIT (number of the current superproblem iteration equals total number of superproblem iterations) (or S2IT=S2NIT),
(ii) S1NUM (number of current superproblem) equals appropriate value,
(iii) IPROB (number of current problem) equals number of last problem in the superproblem, and
(iv) IREP=NREP, if there are subproblems with the last problem in the superproblem
and if (i)-(iv) are true, do superproblem finalization.

If a run having multiple problems requires finalization, test
(i) S1IT=S1NIT (or S2IT=S2NIT), if there are superproblems
(ii) S1NUM equals number of last superproblem, if there are superproblems,
(iii) IPROB=NPROB (number of the current problem equals total number of problems), and
(iv) IREP=NREP, if there are subproblems with the last problem in the superproblem
and if (i)-(iv) are true, do run finalization.

An initialization block is a block of abbreviated code that is only executed at ICALL=0 or ICALL=1. A finalization block is a block of abbreviated code that is only executed at ICALL=3. E.g.,

```
IF (ICALL.EQ.1) THEN
 ... initialization block ...
ENDIF
```

Such blocks may be present in $PRED, $PK, $ERROR, and $INFN blocks of abbreviated code. If such blocks are present in $PK or $ERROR, an INFN routine is used to implement the logic in these blocks. Initialization and finalization blocks will be implemented by means of a generated FORTRAN subroutine.

Variables may be used as right-hand quantities even before they are defined; if an expression which uses such a variable is computed before any value of the variable is computed, the computation of the expression will be uncertain.

Assignment, conditional, WRITE and PRINT statements may be used.

In addition, these rules apply:

Defined quantities are not regarded as random variables; eta derivatives are not computed in an initialization or finalization block

Transgeneration of the data is permitted. NONMEM data items ID and MDV may not be changed. If a data item label may appear on the left of an assignment statement, then NM-TRAN generates assignment statements changing first the data item in the event or data record, and then the value of the local variable having that label. Note, however, that at ICALL=0,1,or 3, by default, references to data items are references to those data items

in the first data or event record. To transgenerate an item in any data or event record (including the first), use of the NONMEM utility routine PASS is required. See below.

Calls to certain NONMEM routines are permitted:

```
CALL PASS(MODE)
CALL RANDOM(n,R)
CALL SUPP(ie,ic)
```

`CALL PASS(MODE)` must be coded exactly in this way. If `CALL PASS(MODE)` is present, MODE becomes a reserved variable and may be used only with other instances of `CALL PASS(MODE)`. Multiple calls to PASS may be present.

If `CALL RANDOM(n,R)` is present, R becomes a reserved variable and may be used only with other instances of `CALL RANDOM(n,R)`. n may only be an integer value 1-10.

SUPP is used to suppress portions of the NONMEM output report. (See **supp**). The arguments ie and ic may only be 0 or 1.

The following variables may be used on the right (their values change with calls to PASS):

Data record items (See **PASS: PASSRC**).
`NEWIND` (See **PASS NEWIND: NWIND**).
`NIREC`, `NDREC` (See **Record Counters: NIREC,NDREC**).
`NEWL2` (See **PASS New L2 record: NEWL2**).
`ETA` when ICALL=3 (See **Simulation: ETA,EPS**)
`LIREC` (See **Size of Individual Record**)
`PRED_`, `RES_`, `WRES_` when ICALL=3 (See **PRED,RES,WRES**).
`IERE`, `IERC` when ICALL=3 (See **Estim Covar Error Codes**).
`NINDR`, `INDR1`, `INDR2` (See **NINDR INDR1 INDR2**).
`NREP`, `IREP` (See **Simulation: NREP,IREP**).
`NPROB`, `IPROB`, `S1NUM`, `S2NUM`, `S1NIT`, `S2NIT`, `S1IT`, `S2IT` (See **Problem_Iteration_Counters**).

RETURN and EXIT statements may be used.

DOWHILE loops are permitted. The syntax is as follows.

```
DO WHILE (condition)
 .. statements ..
END DO
```

Here is an example of a transgeneration loop.

```
$PRED
IF (ICALL.EQ.0) THEN
     MODE=0
     CALL PASS (MODE)
     MODE=2
     CALL PASS (MODE)
     DO WHILE (MODE.EQ.2)
     ... transgeneration statements ...
     CALL PASS (MODE)
     ENDDO
    RETURN
ENDIF
```

This type of usage of the PASS routine can be coded more simply, as follows:

```
$PRED
IF (ICALL.EQ.0) THEN
     DOWHILE (DATA)
     ... transgeneration statements ...
     ENDDO
    RETURN
ENDIF
```

The `DOWHILE (DATA)` causes the transgeneration statements to be executed with each data record. In effect, NM-TRAN supplies the statements `MODE=...` and `CALL PASS(MODE)` that are shown in the above.

Variables that are first defined in an initialization block or finalization block are not stored globally in NMPRD4, but rather, are stored in module PRINFN. (This makes them available to subroutine MIX.)

Variables defined in an initialization or finalization block may be used freely outside of such blocks, and vice versa: PRED-defined variables defined outside such blocks may be used within them.

THETA variables may be used in initialization and finalization blocks. They are obtained from the subroutine argument.
At ICALL=0, THETA contains the initial estimates for the first problem.
At ICALL=1, THETA contains the initial estimates for the current problem.
At ICALL=3, THETA contains the final estimates for the current problem.

Here is an example of code that could be used during a simulation with multiple subproblems.

```
$PRED
IF (ICALL.EQ.1) THEN
   SUM=0
   N=0
   RETURN
ENDIF
IF (ICALL.EQ.3) THEN
  N=N+1
  SUM=SUM+THETA(1)
  RETURN
ENDIF
IF (ICALL.EQ.3.AND.N.EQ.NREP) THEN
  MEAN=SUM/N
  PRINT *,MEAN
ENDIF
```

The following variables may be referenced as right-hand quantities in initialization and finalization blocks. (Note also that unsubscripted arrays may appear in a WRITE statement.)

`OMEGA(n,m)`
`SIGMA(n,m)`
(See **Parameter Values: Initial and Final**).
(See **write print**).

The following variables may be referenced as right-hand quantities in finalization blocks. (Note also that unsubscripted arrays may appear in a WRITE statement.)

```
SETHET(n)
SETHETR(n)
SEOMEG(n,m)
SESIGM(n,m)
OBJECT
```

(See **Standard Errors**).
(See **Objective Function Value**).
(See **write print**).

The following statements are forbidden:

```
CALL SIMETA(ETA)
CALL SIMEPS(EPS)
```

When the PRED repetition feature is used, the variables `RPTO` and `PRDFL` may appear as left-hand quantities in initialization blocks.
(See **Repetition Variables**).

(See **abbreviated**).

REFERENCES: Guide II, section D.2.2
REFERENCES: Guide VI, section VI.A , Figure 37

INTRA AND INTER-INDIVIDUAL

MEANING: Types of random errors
CONTEXT: NONMEM terminology

DISCUSSION:

Random intra-individual ("within an individual") and inter-individual ("between individuals") variability are terms that are used in discussing NONMEM output.

Case 1. When the data are population

Random inter-individual variability refers to the unexplained difference between individuals' parameter values and the typical values in the population. It is described in NM-TRAN using `ETA(n)` variables. PRED describes the effect of these variables to NONMEM via the partial derivatives in the G array. The variance of these variables is given in the OMEGA matrix.

Random intra-individual variability refers to unexplained (residual) error in a model (the difference between observed and predicted values). It is described in NM-TRAN using `EPS(n)` or `ERR(n)` variables. PRED describes the effect of these variables to NONMEM via the partial derivatives in the H array. The variance of these variables is given in the SIGMA matrix.

Case 2. When the data are single-subject

Random inter-individual variability refers unexplained (residual) error in a model (the difference between observed and predicted values). This, despite the fact that data may come from a single subject, is because each observation is placed into a separate individual record, and because only level-one random effects are needed to describe it. It is described in NM-TRAN using `ERR(n)` or `ETA(n)` variables. PRED describes the effect of these variables to NONMEM via the partial derivatives in the G array. The variance of these variables is given in the OMEGA matrix.

However, when the data are in fact from a single subject, and when there is no need to use NONMEM terminology, then users often associate the `ETA(n)` and OMEGA with intra-individual variability, which corresponds to their experimental point of view.

REFERENCES: Guide I, section A.5 , E

STOP TIME,ITASK (NM75)

MEANING: PREDPP-PK global variables
CONTEXT: User-supplied routines

USAGE:

```
USE PRLS01_REAL, ONLY: STOP_TIME
USE PRLS01_INT, ONLY: ITASK_=>ITASK
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: DPSIZE
REAL (KIND=DPSIZE) STOP_TIME
USE SIZES, ONLY: ISIZE
INTEGER(KIND=ISIZE) ITASK_
```

DISCUSSION:

These reserved variables are used to avoid overshoot in ADVAN9, ADVAN13, ADVAN14, and ADVAN15. These LSODA based routines use an algorithm for integration that overshoots the integration interval during calls to DES, but still accurately evaluates at the end of the integration interval when all calculations are completed. However, you may wish to capture a maximal or minimal value during $DES, and the overshoot should be turned off for this purpose. This is readily done by setting ITASK_=4 in $PK or $ERROR. E.g.,

```
$PK ITASK_=4
```

ITASK_ may take values between 1 and 5.

For other values of ITASK_ and a discussion of these variables,
See **INTRODUCTION TO NONMEM 7, "ITASK_ and STOP_TIME: Avoiding overshoot in ADVAN9, ADVAN13, ADVAN14, and ADVAN15"**

You may also specify a STOP_TIME (Tcrit) past which it should not integrate, if it is different from the end of the normal integration interval:

```
IF(TIME==4.0) STOP_TIME=5.0
```

To set back to default (end of normal integration interval),

```
STOP_TIME=-1.0d+300
```

REFERENCES: Guide Introduction_7

L2 DATA ITEM

MEANING: Level Two (L2) data item for NONMEM
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... L2 ...`

DISCUSSION:
`L2` labels level-two (L2) data items. The level two data item is optional.

Recall that the ID data item is used to group together the data records containing observations which have the same realization of the level-one random effects (etas). Similarly, the L2 data item is used to group together the data records containing observations which have the same realization of the level-two random effects (epsilons). The observations of such a group is called a level-two observation. The group itself is called a level-two (L2) record. Data records of an L2 record must be contiguous (and contained within the same individual record). By default, level-two observations are treated as being statistically independent multivariate observations. (However, within a level-one observation, the level-two random effects can be made to be correlated between level-two observations.)

Here is an example of a fragment of a data set using L2 data items. There are two types of observations, designated by the two different values of the user data item TYPE. Note that the L2 data items are the same for both of the observations at TIME=2, and they are also the same for both of the observations at TIME=112.5.

ID	TIME	AMT	APGR	WT	DV	TYPE	L2
1	0.	25.0	1.4	7	.	.	1
1	2.0	.	1.4	7	6.0	1	1
1	2.0	.	1.4	7	17.3	2	1
1	12.5	3.5	1.4	7	.	2	2
1	24.5	3.5	1.4	7	.	2	2
1	37.0	3.5	1.4	7	.	2	2
1	48.0	3.5	1.4	7	.	2	2
1	60.5	3.5	1.4	7	.	2	2
1	72.5	3.5	1.4	7	.	2	2
1	85.3	3.5	1.4	7	.	2	2
1	96.5	3.5	1.4	7	.	2	2
1	108.5	3.5	1.4	7	.	2	2
1	112.5	.	1.4	7	8.0	1	2
1	112.5	.	1.4	7	31.0	2	2

When the data are single-subject and the L2 data item is defined to NM-TRAN in a $INPUT record, the ID data item is taken to be the L2 data item. The ID item may be defined, but then it is not in turn identified to NONMEM as such. The labels L2 and ID (if defined) may be used in abbreviated code, and then they refer to the corresponding items of the data record, as specified in the $INPUT record. However, if the user uses the label L1 instead of the label ID, or uses the label L1 as a synonym with the ID label, then NM-TRAN does not change the designations: For NONMEM, the items labeled L1 are taken to be the ID data items, and the items labeled L2 are taken to be the L2 data items.

REFERENCES: Guide II, section D.3.3
REFERENCES: Guide V, section 12.4.2 , 12.5

LBLR

MEANING: Insert User-Defined Labels in Additional Files (NM75)
CONTEXT: NONMEM run

The label substitution process is described in
I.28.General New Options for $ESTIMATION Record (NOSUB=0)
I.6.Expansions on Abbreviated and Verbatim Code and Other Items
It is normally done only in the NONMEM report file, but not in the additional files, such as the .ext, .cov, .coi, .phi, etc. These additional files need often be read by 3rd party software, which may rely on traditional names for thetas, omegas, sigmas, phi, eta, etc. Nonetheless, one may wish to make label substitutions on these additional files, which can be done after the NONMEM analysis, using the label_replace utility.

USAGE:

```
lblr file1 file2 file3 [S|T|C] [0|1|2]
```

file1

Input file to have label substitutions done. Typically this is the name of the NONMEM report file (e.g., root.res).

file2

NONMEM report file containing label substitution patterns, or any file containing label substitution patterns such as:
THETA(1)=THETA(CL)
ETA(2)=ETA(ETV1)
EPS(1)=EPS(RSW)

file3

The results are written to file3.

Option 1

Option 1 may be summarized as [S|T|C]

```
S=Expand and adjust with spaces (default)
T=Do not expand, and truncate label if necessary
C=Expand, and compress so there are no spaces (suitable only for comma de
```

Option 2

Option 2 may be summarized as [0|1|2]

```
0=Full name of parameter type (THETA/OMEGA/SIGMA/ETA/EPS/PHI/ETC/PHC) (de
1=two letter truncation of parameter type name (TH/OM/SG/ET/EP/PH/EC/PC)
2=single letter truncation of parameter type name (T/O/S/E/P/H/C/C)
```

Truncated parameter type name (1 or 2) works well with T, when no label expansion is permitted, and you want to make the label as compact as possible.

If the user specified NOSUB=1, then the thetas, omegas/etas, and sigmas/eps will not have been label substituted during the NONMEM execution. The user may wish to make the label substitution afterword with label_replace. For example,

```
lblr myresult.res myresult.res myresult_new.res T 2
```

Notice in this example that the report file myresult.res serves as the input file, and the file from which to get label substitution pattern.

If you also want to perform an lpcc, do lblr first, then the lpcc utility.

See **INTRODUCTION TO NONMEM 7, lblr: Insert User-Defined Labels in Additional Files**

REFERENCES: Guide Introduction_7

LPCC

MEANING: Line Printer Controls Converter Program (NM75)
CONTEXT: NONMEM run

The utility program Line Printer Controls Converter will convert the line printer control characters in the NONMEM report file into their proper actions. The line printer control characters are in the first character in each line, and at one time was interpreted by old 132 column line-printers.

USAGE:

```
lpcc inputfile outputfile codes ...
```

Inputfile
: should be the name of the NONMEM report file (e.g., root.res).

outputfile
: The results are written to outputfile.

codes ...
: A list of 1 or more codes.

The conversion is acted upon only between the lines
NM-TRAN MESSAGES
and just before
Stop Time:
as the lines before and after these do not use column 1 as a line-printer control column.

The code command characters are interpreted by lpcc as follows:

Code Option character	Acting on Control	Action
A0	+	Advance 0 lines (overprint on (merge with) previous line)
A1	space	Advance 1 line (no action is taken)
A2	0	Advance 2 lines (insert a blank line)
A3	-	Advance 3 lines (insert 2 blank lines) Not used in NONMEM
AS		Remove action character column once code is acted upon (But will not remove character in col 1 if not one of the above)
FF	1	Advance page (insert Form feed character)
FFA	1	Advance page and add blank line (insert Form feed with a next line)
FFA2	1	Advance page and add 2 blank lines
FF1	1	Replace ’1’ as ’ ’ (interpret FF as A1)
FF2	1	Replace ’1’ as ’0’ (interpret FF as A2)
FF3	1	Replace ’1’ as ’-’ (interpret FF as A3)
ALL		FF,A0,A1,A2,A3
ALLS		FF,A0,A1,A2,A3,AS

After an action, the control character is replaced with a space. That space will be printed unless AS is requested. So AS shifts the line one column to the left.
An N followed by the code prevents that action from occurring.

EXAMPLE (options are acted upon in order):

```
lpcc example1.res example1.lst FF A0
```
Only replace "1" in column 1 with form feed, and merge line with '+' with previous line.

```
lpcc example1.res example1.lst ALL FFA
```
Act on all codes, and replace "1" with FF/next line.

```
lpcc example1.res example1.lst ALLS FFA
```
Act on all codes, replace "1" with FF/next line, and shift line one column to the left.

See **INTRODUCTION TO NONMEM 7, lpcc: Line Printer Controls Converter Program**

REFERENCES: Guide Introduction_7

MDVRES

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPRD_INT, ONLY: MDVRES
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: MDVRES
```

DISCUSSION:

MDVRES stands for missing dependent variable (MDV) for residual (RES). Setting MDVRES to 1 is equivalent to temporarily declaring an observation as missing during the computation of residuals and weighted residuals.

MDVRES=0 (default)

Set MDVRES to 1 in the $ERROR or $PRED routine if you do not want to include a particular observation in the computation of residual and weighted residuals.

One situation in which this may be useful is when F_FLAG is used because some observations are predictions (F_FLAG=0) and others are likelihood or -2 log likelihood values (F_FLAG=1 or F_FLAG=2). By default, if F_FLAG is set to 1 or 2 for any observation within an individual record, the RES and WRES items will be 0 for all data records in the individual record. When MDVRES is set to 1 for any observation, this overrides the default. NONMEM will compute RES and WRES for all observations for which MDVRES is set to 0. MDVRES should be set to 1 with all observations having F_FLAG=1 or F_FLAG=2, but may be set to 1 for other observations as well.

EXAMPLE:

Supose that some observations are assessed by a non-normal distribution likelihood such as the PHI() function for below detection limit values, in which F_FLAG is set. By setting MDVRES=1 to these particular below detection values, the weighted residual algorithm can assess the remaining normally distributed values for that subject.

```
$ERROR
SD = THETA(5)
IPRED = LOG(F)
DUM = (LOQ - IPRED) / SD
CUMD = PHI(DUM)
IF (TYPE .EQ. 1) THEN
      F_FLAG = 0
      Y = IPRED + SD * ERR(1)
ENDIF
IF (TYPE .EQ. 2) THEN
      F_FLAG = 1
      Y = CUMD
      MDVRES=1
ENDIF
```

REFERENCES: Guide Introduction_7

MINIMUM VALUE OF OBJ. FUNCTION

MEANING: The value of the objective function
CONTEXT: NONMEM output

DISCUSSION:
The NONMEM OBJECTIVE FUNCTION is a goodness of fit statistic; the lower the value, the better the fit. Negative values are possible and have no special significance. Under certain assumptions, the (default) objective function value is minus twice the log likelihood.

A page with the title `MINIMUM VALUE OF OBJECTIVE FUNCTION"` is printed as part of every NONMEM output, as in this example.

```
******        MINIMUM VALUE OF OBJECTIVE FUNCTION        ***********
*****************          11.570     ******************************
```

If the Estimation Step was NOT run, then the value printed is the value of the objective function using the initial parameter estimates specified for the problem.

If the Estimation Step WAS run, then the value printed is the value of the objective function with the final parameter estimates.

Differences in the objective function of fits of the same data to hierarchical models can often be used to test approximately the plausibility of the smaller (fewer free parameters) model by referencing the difference to a chi-square distribution with degrees of freedom equal to the difference in the number of free model parameters between models (hypothesis tests).

Sometimes NM-TRAN may generate this warning message:

```
(WARNING  68) THE INTEGER FUNCTION IS BEING USED OUTSIDE OF A SIMULATION
 BLOCK. IF THE INTEGER VALUE AFFECTS THE VALUE OF THE OBJECTIVE FUNCTION,
 THEN AN ERROR WILL PROBABLY OCCUR.
```

In general, any code that affects the value of the prediction F (with $PRED) or Y (with $ERROR), or their eta and eps partials, affects the value of the objective function. Such code must be continuous, e.g., must not use functions such as INT or MOD, and must not involve conditional statements that introduce discontinuities. The following examples use Model Event Time parameters (MTIME) to avoid discontinuities in differential equations:

Model Time examples
Examples Using MTIME to Model Periodic Discontinuities in $DES
Enterhepatic circulation examples

(See **mtime**, **model time examples**).
(See **Circadian Example: Examples Using MTIME to Model Periodic Discontinuities in $DES**)
(See **Enterhepatic circulation examples**).

REFERENCES: Guide I, section C.3.5.1
REFERENCES: Guide V, section 5.5

MIX CONTR: TEMPLT

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied PRED routine

USAGE:

```
USE ROCM_REAL, ONLY: TEMPLT=>VRAW
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: VSIZE,DPSIZE
REAL(KIND=DPSIZE) :: VRAW(VSIZE)
```

DISCUSSION:

`TEMPLT`

This variable serves two different purposes.
At ICALL=6, the template data record is stored in TEMPLT.
When MIX is called, the first data record of the individual record is stored in TEMPLT.

TEMPLT(n) may be used as a right-hand quantity in blocks of abbreviated code that test for ICALL=6 and in $MIX blocks. The "n" may be either a position in the data record or the label of the data item, e.g. TEMPLT(1) or TEMPLT(ID).

Location prior to NONMEM 7: rocm31

REFERENCES: None.

MIX CONTR: THETA

MEANING: NONMEM read-only global variables

CONTEXT: MIX, CONTR, CCONTR routine

USAGE:

```
USE ROCM_REAL, ONLY: THETA=>THETAC
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: LTH,DPSIZE
REAL(KIND=DPSIZE) :: THETAC(LTH)
```

DISCUSSION:

`THETA`

The current value of theta, made available by NONMEM for the MIX, CONTR and CCONTR routines. THETA is a reserved varible in $MIX abbreviated code.

Location prior to NONMEM 7: rocm0

REFERENCES: Guide VI, section III.L.2 , Figure 6

MIX

MEANING: MIX subroutine
CONTEXT: User-supplied subroutine; replaces a NONMEM dummy routine

USAGE:

```
SUBROUTINE MIX (ICALL,NSPOP,P)
USE SIZES, ONLY: ISIZE,DPSIZE,LTH
USE ROCM_REAL, ONLY: THETA=>THETAC
INTEGER(KIND=ISIZE), INTENT(IN) :: ICALL
INTEGER(KIND=ISIZE), INTENT(OUT) :: NSPOP
REAL(KIND=DPSIZE),   INTENT(OUT) :: P(*)
```

GLOBAL DECLARATION:

```
REAL(KIND=DPSIZE) :: THETAC(LTH)
```

DISCUSSION:
MIX is a NONMEM routine that is replaced by a user-supplied routine when a mixture model is used. The MIX subroutine is used to describe the mixture parameters of a mixture model. It is called by NONMEM with one individual record after another.

Input argument:

`ICALL`
: Similar to ICALL for PRED subroutine.

Output argument:

`NSPOP`
: An integer variable or integer constant. The maximum number of sub-populations that are possible. Must be given a value when ICALL=1.

`P`
: An array. For each i (i=1, ... , NSPOP), P(i) gives the modeled fraction of the population in the ith subpopulation. The sum of the P(i) should equal 1. In principle, the P(i) can change from individual to individual. If for a given individual, the second (for example) subpopulation doesn't apply, then set P(2)=0 for that individual.

Other inputs are available to MIX in NONMEM read-only global variables. In particular, data items that are requested using the $CONTR record, and the current value of THETA, as shown above.
(See **MIX:_DATA**, **MIX CONTR:_THETA**).

The TEMPLT data record in a NONMEM read-only global variable also serves to provide an additional way for individual-specific information to be made available
(See **MIX_CONTR:_TEMPLT**)

With mixture models, `MIXNUM`, `MIXEST` and `MIXP` are variables that may be used as right-hand quantities (or in logical conditions) in various other abbreviated codes or user-supplied routines. `MIXNUM` is the index of the subpopulation for which variables are to be computed. At ICALL=3, and at ICALL=2 when COMACT is not 0, `MIXEST` is the index of the subpopulation estimated to be that from which the individual's data most probably arises. `MIXP` refers to the mixture probabilities P computed by subroutine MIX.

This is a general description. For details, see mixnum.

(See **MIXNUM_MIXEST_MIXP**)
(See **mixnum_mixest_for_mixture_model**)

(See **Mixture_model:_MIXP**)
(See **mixture_model_example**, **mixnum_mixest**).

REFERENCES: Guide III, section V.5.0

REFERENCES: Guide IV, section III.B.4 , III.B.6
REFERENCES: Guide IV, section IV.E.1 , 4.E.2
REFERENCES: Guide VI, section III.L.2 , Figure 6

MIX: DATA

MEANING: NONMEM read-only global variables
CONTEXT: MIX routine

USAGE:
```
USE ROCM_REAL, ONLY: DATA=>RDATA
```

GLOBAL DECLARATION:
```
USE SIZES, ONLY: NO,DPSIZE
REAL(KIND=DPSIZE) :: RDATA(NO,3)
```

DISCUSSION:

`DATA(k,i)`
: The value of the ith type of data item specified in NM-TRAN's $CONTR record, appearing on the kth observation record of the individual record. It changes values with each individual record. DATA is a reserved variable in $MIX abbreviated code.

DATA is also used in CONTR
(See **CCONTR:_Y,DATA,N1,N2,DIM**)

Location prior to NONMEM 7: rocm1

REFERENCES: Guide IV, section III.B.4

MIXNUM MIXEST MIXP

MEANING: Variables used with mixture models
CONTEXT: Abbreviated code and user-supplied subroutines

USAGE:

```
IF (MIXNUM.EQ.1) THEN
  ...
ENDIF
IF (MIXNUM.EQ.2) THEN
  ...
ENDIF
IF (MIXNUM.EQ.3) THEN
  ...
ENDIF
```

DISCUSSION:

With mixture models, `MIXNUM`, `MIXEST` and `MIXP` are global variables that may be used as right-hand quantities (or in logical conditions) in abbreviated code or user-supplied routines.
In general:
MIXNUM is an input to PRED (or PREDPP) set by NONMEM.
MIXEST is an output (result) or consequence from the estimation set by NONMEM.

`MIXNUM`

This is the index of the subpopulation for which variables are to be computed.

During ICALL=4, MIXNUM is the index of the sub-population that was randomly selected to simulate the subject's data. When ICALL=3 and the PASS subroutine used, or when ICALL=5 or 6, MIXNUM is the index of the subpopulation estimated to be that from which the individual's data most probably arises, and is equal to MIXEST (see below).

`MIXEST`

When ICALL=3,5,6, and at ICALL=2 when COMACT is not 0, MIXEST is the index of the subpopulation estimated to be that from which the individual's data most probably arises. Before nm7.3, at all other conditions, the value of MIXEST is not meaningful, such as when ICALL=4 or ICALL=2 and COMACT=0. As of NONMEM 7.3, when ICALL=4, MIXEST will equal MIXNUM, and when ICALL=2 and COMACT=0, MIXEST will be 0 to indicate it has not been assigned a meaningful value.

`MIXP`

These are the mixture probabilities `P(i)` computed by subroutine MIX or by the $MIX block of abbreviated code.

In abbreviated code, MIXP may be coded in either of three ways:

```
MIXP(MIXNUM)
MIXP
MIXP(i)
```

The first two ways are identical; i.e., when no subscript is coded, NM-TRAN supplies the subscript `MIXNUM`.

With the third, the index i must not exceed the value of MMX in SIZES (or as set by $SIZES record).

(See **Mixture_model:_MIXNUM,MIXEST**)
(See **Mixture_model:_MIXP**)
(See **mix**, **mixture model example**).

REFERENCES: Guide IV, section III.B.4 , III.B.6
REFERENCES: Guide IV, section IV.E.1 , 4.E.2
REFERENCES: Guide VI, section III.L.2 , Figure 6

MIXTURE MODEL: MIXNUM,MIXEST

MEANING: NONMEM read-only global variables
CONTEXT: PRED routine

USAGE:

```
USE ROCM_INT,  ONLY: MIXNUM=>MIXCALL,MIXEST=>IMIXEST
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: MIXCALL,IMIXEST
```

DISCUSSION:

`MIXNUM`
: The number of the subpopulation for which PRED is to compute outputs.

`MIXEST`
: The number of the subpopulation estimated to be that from which the individual's data most probably arises. Should be used at ICALL = 3 and at ICALL = 2 when COMACT > 0.

Used with mixture models. MIXNUM and MIXEST are reserved variables in all blocks of abbreviated code.

This is a general description. For details, see mixnum.

(See **MIXNUM_MIXEST_MIXP**)
(See **Mixture_model:_MIXP**, **mix**)
(See **mixture_model_example**)

Location prior to NONMEM 7: rocm11

REFERENCES: Guide VI, section III.L.2 , Figure 6

MIXTURE MODEL: MIXP

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

USAGE:
```
USE ROCM_REAL, ONLY: MIXP
```

GLOBAL DECLARATION:
```
USE SIZES, ONLY: MMX,DPSIZE
REAL(KIND=DPSIZE) :: MIXP(MMX)
```

DISCUSSION:

Used with mixture models.

`MIXP(i)`
The mixture probability for the ith subpopulation, computed by subroutine MIX.

May be used as right-hand quantities in all abbreviated codes other than $MIX. There the index i must be an integer constant.

This is a general description. For details, see mixnum.

(See **MIXNUM_MIXEST_MIXP**)
(See **mixnum_mixest_for_mixture_model**)
(See **mix**, **mixture model example**)

Location prior to NONMEM 7: rocm25

REFERENCES: Guide VI, section III.L.2 , Figure 6

MIXTURE MODEL: MIXPT

MEANING: NONMEM read-only global variables
CONTEXT: PRED routine

USAGE:

```
USE ROCM_REAL, ONLY: MIXPT=>MIXP_RAW
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: MMX,DPSIZE
REAL(KIND=DPSIZE) :: MIXP_RAW(MMX)
```

DISCUSSION:

`MIXPT`

At ICALL=6, (the final estimate of) the mixture probabilities associated with the individual record containing the template data record are stored in MIXPT.

Location prior to NONMEM 7: rocm47

REFERENCES: None.

MODEL EVENT TIME: MNOW,MPAST,MNEXT

MEANING: PREDPP read-only global variables
CONTEXT: User-supplied routines

USAGE:

```
USE ROCM_INT, ONLY: MNOW=>MTNOW,MPAST=>MTPAST,MNEXT=>MTNEXT
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: PCT
INTEGER(KIND=ISIZE) :: MTNOW,MTPAST(PCT),MTNEXT(PCT)
```

DISCUSSION:

These variables are of interest when model event time parameters are defined in PK.

(See **mtime**).

Location prior to NONMEM 7: procma

REFERENCES: None.

MODEL EVENT TIME: MTDIFF

MEANING: PREDPP-PK global variables
CONTEXT: PK routine

USAGE:
```
USE PRMOD_INT, ONLY: MTDIFF
```

GLOBAL DECLARATION:
```
INTEGER(KIND=ISIZE) :: MTDIFF
```

DISCUSSION:

`MTDIFF` is of interest when model time parameters `MTIME` are used.

The value of `MTDIFF` is 0 when PK is called. If PK sets `MTDIFF` to a value other than 0, e.g., `MTDIFF=1`, then PREDPP will understand that with that call to PK, the values of one or more of the `MTIME(i)` have possibly been reset.

It is not necessary to set `MTDIFF` at a call to PK with the first record of an individual or with a reset record. At such calls, it is assumed that all `MTIME(i)` are being set (if only to their default value of 0).

Suppose PK has defined a value `MTIME(i)`. PREDPP calls PK prior to the advance to `MTIME(i)`. PK may change `MTIME(i)` at this call. However, the new value of `MTIME(i)` is ignored till after the advance. Only then is the new value of `MTIME(i)` effective.

`MTDIFF=0` (the default) can save considerable run time when there are many model time parameters. Note that the results are unpredictable if the times are in fact changed when `MTDIFF=0`.

(See **mtime**, **model time examples**).

Location prior to NONMEM 7: prdpk2

REFERENCES: None.

MODEL EVENT TIME: MTIME

MEANING: PREDPP-PK global variables
CONTEXT: PK routine

USAGE:

```
USE PKERR_REAL, ONLY :: MTIME
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: PCT,DPSIZE
REAL(KIND=DPSIZE) :: MTIME(PCT)
```

DISCUSSION:

Model time parameters `MTIME` are stored in MTIME if they are defined in $PK abbreviated code or if verbatim code is present in $PK. This makes them also available to the ERROR routine. However, their eta derivatives (if any) are not included. Hence the ERROR routine should not use model time parameters in such a way as to influence the value of Y if those parameters have eta derivatives.

A user-written PK routine need not define these variables unless perhaps the model time parameter values are used in the ERROR routine.

(See **mtime**, **model time examples**).

Location prior to NONMEM 7: prdpk1

REFERENCES: None.

MODEL

MEANING: MODEL subroutine
CONTEXT: User-supplied subroutine; for use with PREDPP

USAGE:

```
SUBROUTINE MODEL(IDNO,NCM,NPAR,IR,IATT,LINK)
USE PRMOD_CHAR, ONLY: NAME
USE SIZES,      ONLY: DPSIZE,ISIZE
INTEGER(KIND=ISIZE) :: IDNO,NCM,NPAR,IR,IATT,LINK
DIMENSION :: IATT(IR,*),LINK(IR,*)
```

GLOBAL DECLARATION:

Versions before NONMEM 7.4:

```
CHARACTER(LEN=8) :: NAME(PC)
```

Versions after NONMEM 7.4:

```
USE SIZES,      ONLY: SD
CHARACTER(LEN=SD) :: NAME(PC)
```

DISCUSSION:
The MODEL subroutine is called by PREDPP only once at the start of a run when a general ADVAN (ADVAN 5, 6, 7, 8, 9, 13, 14, 15, 16, 17, 18) is used. It allows the user to specify aspects of the particular model he wishes to use. When NM-TRAN is used, the $MODEL record supplies this information.

Input argument: None.

Output argument:

`IDNO`
: An identification number for the MODEL routine. The value assigned by MODEL to IDNO is printed on the first PREDPP problem summary page.

`NCM`
: The total number of compartments in the model, excluding the output compartment. Contains 0 when MODEL is called. NCM must be no greater than PC-1.

`NPAR`
: The number of basic PK parameters used in the PK routine. Contains 0 when MODEL is called. NPAR must be no greater than constant PG in file SIZES. With the general non-linear models (ADVAN6, ADVAN8, ADVAN9, ADVAN13, ADVAN14, ADVAN15, ADVAN16, ADVAN17, ADVAN18), NPAR may remain 0.

`IATT`
: Values of compartment attributes. The values of IATT(I,*) refers to the ith compartment.
  `IATT(I,1)=0` initially off
  `IATT(I,1)=1` initially on
  `IATT(I,2)=0` may not be turned on and off
  `IATT(I,2)=1` may be turned on and off
  `IATT(I,3)=0` may not receive doses
  `IATT(I,3)=1` may receive doses
  `IATT(I,4)=0` not the default observation compartment
  `IATT(I,4)=1` the default observation compartment

`IATT(I,5)=0` not the default dose compartment
`IATT(I,5)=1` the default dose compartment

The remainder is used only with ADVAN9, ADVAN15, and ADVAN17:

`IATT(I,8)=0` not an equilibrium compartment
`IATT(I,8)=1` an equilibrium compartment
`IATT(I,9)=0` should not be included in the total drug amount in the system interior
`IATT(I,9)=1` should be included in the total drug amount in the system interior

`NAME`

Labels for the compartments (to be printed on the PREDPP summary page under "FUNCTION").
NAME(I) is the label for compartment i. With versions before NONMEM 7.4, NAME was limited to 8 characters. With NONMEM 74 NAME may be up to SD characters long. SD is defined in SIZES and is set to 30 with NONMEM 7.4.

`LINK`

Only used with a general linear model (ADVAN5 and ADVAN7).
`LINK(I, J)=0`: no drug may distribute from compartment i to compartment j.
`LINK(I, J)=K`: drug distributes from compartment i to compartment j. The rate constant quantifying this first order distribution is computed by PK and stored in the kth row of GG.
When MODEL is called, all elements of LINK are 0.

(See **sizes**).

An initial steady state may be requested by routine MODEL;
(See **Initial Steady State: I_SS,ISSMOD**)
(See **i_ss**, **initial_condition**).

REFERENCES: Guide IV, section V.C.4
REFERENCES: Guide VI, section VI.B , Figures 28, 29, 43
REFERENCES: Guide VI, Appendix II

MODEL SPECIFICATION FILE

MEANING: A NONMEM output and input file
CONTEXT: NONMEM input/output file

DISCUSSION:
A Model Specification File (MSF) can be output by NONMEM. It includes information from the Estimation Step (initial estimate information, current (or final) estimate information, search information) and from the Covariance Step (if this step is implemented). It is requested in a certain field of the ESTIMATION control record (or, with NM-TRAN, by an option of the $ESTIMATION record). NONMEM will use the information in the MSF in a later problem, either to continue the parameter search if the search did not terminate in the current run, or to perform other tasks that will be based on final parameter estimates in the MSF. These other tasks can be Simulation, Covariance, Table, Scatterplot, or Finalization Steps. If the MSF contains the results of a search that terminated successfully, even an additional Estimation Step can be performed, using the final parameter estimates in the MSF as initial estimates and an estimation method other than the one used in the problem with which the MSF was output. If an MSF is output, and intermediate output with iteration summaries is requested, then NONMEM also writes the parameter estimates of these summaries to file INTER.

MSF files are often used because they eliminate needless duplication of computation that has already occurred in the Estimation Step of a previous run. If there should be a computer crash during or after the Estimation Step of the previous run and a MSF was output, then with the current run, the MSF can be input, and the Estimation Step of the current run will continue the search begun with the previous run, starting from the last iteration whose summary was output before the crash occurred.

With NM-TRAN, specification that an existing Model Specification File is to be input is done by placing the file's name on the $MSFI record.

With NONMEM 7.3, when MSF or MSFO option is used to specify an MSFO file in the $EST record e.g.,
$EST ... MSFO=msfroot.msf
then in addition to the main MSF file msfroot.msf, an extra file msfroot_ETAS.msf will also be produced, and provide additional information when a $MSFI record is used in a subsequent problem or control stream. File msfroot_ETAS.msf contains individual etas and phis generated during estimation (except FO method, which does not generate nonzero etas during estimation, and not POSTHOC evaluated etas.) The use of an extension in the file name, e.g., .msf, is optional. If file _ETAS does not exist, NONMEM issues a warning, e.g.,

```
WARNING: EXTRA MSF FILE COULD NOT BE OPENED: c5msf2x_ETAS
```

If the Covariance Step is also implemented, extra files msfroot_RMAT.msf and msfroot_SMAT.msf containing intermediate information on the R matrix and S matrix will also be produced. These files provide information when a $MSFI record along with a $COV ... RESUME record is used in a subsequent problem or control stream. There is no warning if files _SMAT and/or _RMAT are missing.

EXAMPLE OF USAGE:

Here is an example using NM-TRAN. A fragment of the control stream for the first run follows. The Estimation Step is run for a limited number of evaluations of the objective function (200).

```
$THETA  (.1,3,5) (.008,.08,.5) (.004,.04,.9)
$OMEGA BLOCK(3)  6 .005 .0002 .3 .006 .4
$SIGMA  .4
$EST     MAXEVAL=200  PRINT=5 MSFO=msf1
```

After review of the first run's output, it is decided to let the search continue. The continuation will start from the iteration during which the 200th objective function evaluation occurred in the previous search. A fragment of the control stream for the second run follows. $THETA, $OMEGA, and $SIGMA need not, and may not, be present. The Estimation Step is allowed to run to completion, and standard errors and a table are requested. A new MSF is obtained, which will allow additional tables and scatterplots to be obtained later.

```
$MSFI msf1
$EST PRINT=5 MSFO=msf2
$COV
$TABLE ID TIME
```

REFERENCES: Guide I, section B.3
REFERENCES: Guide IV, section III.B.12

MONITORING OF SEARCH

MEANING: History report on the Estimation Step search
CONTEXT: NONMEM output

DISCUSSION:
NONMEM output includes a report of the history of the search undertaken in the Estimation Step for parameter estimates. This report is called the intermediate output from the Estimation Step, because it consists of summaries of the progress of the search, from iteration to iteration, and because it may be viewed as the search progresses, provided the NONMEM output file may be viewed as the search progresses. This report can also be viewed in a special (unbuffered) file.

The search is for parameter values that minimize the value of the OBJECTIVE FUNCTION (See **minimum value of objective function**). Basically, it entails the following steps.

1. The search is carried out in a different parameter space. The parameters are transformed to unconstrained parameters (UCP). In the transformation process a scaling occurs so that the initial estimate of each of the UCP is 0.1. (NONMEM Users Guide I Sections C.3.5.1 "ESTIMATION" and G.3 "Messages from the Estimation Step" refer to the UCP as "scaled transformed parameters" or STP. NONMEM Users Guide II Section F "Rescaling" refers to the UCP as "rescaled canonical parameters" or RCP.)
2. At the current parameter estimate the GRADIENT vector (i.e., the vector of first partial derivatives of the objective function with respect to the UCP) is computed. An approximate Hessian matrix (See **hessian**) is also computed. An ITERATION SUMMARY, including the current parameter estimate and the gradient vector, may placed into the intermediate output.

   With NONMEM 7, the parameter estimates are also displayed in their natural (unscaled) space. These lines are identified as NPARAMETR and precede the PARAMETER lines, which display the UCP values. Note that when OMEGA (or SIGMA) have block structure, the values in NPARAMETR are listed in upper triangular order, whereas elsewhere in NONMEM output they are displayed in lower triangular order. For example, suppose the NONMEM output displays a 3x3 OMEGA matrix as

   OMEGA(1,1)
   OMEGA(2,1) OMEGA(2,2)
   OMEGA(3,1) OMEGA(3,2) OMEGA(3,3)

   It will be listed in NPARAMTR as:

   OMEGA(1,1) OMEGA(2,1) OMEGA(3,1) OMEGA(2,2) OMEGA(3,2) OMEGA(3,3)

   This is inconsistent, but it cannot be changed; option `ORDER` of the $ESTIMATION record does not affect the NPARAMTR line.
3. Using the gradient vector and Hessian matrix, a direction in parameter space, emananting from the current parameter estimate, is computed, and a search is undertaken along this direction for an approximate minimum point. When this point is found, NONMEM returns to step 2. (An ITERATION consists of the computation of the direction, the search along the direction, and the computation of the gradient vector and Hessian matrix at the approximate minimum point.)

4. Iteration stops when any one of the following conditions holds:

   A successful (local) minimum point has been found.

   The maximum number of FUNCTION EVALUATIONS allowed by the user (MAXEVALS option of $ESTIM) is exceeded.

   It was not possible to successfully locate a minimum point due to so-called ROUNDING ERRORS.

Here is an example of intermediate output from the Estimation Step:

```
MONITORING OF SEARCH:
 ITERATION NO.:    0    OBJECTIVE VALUE:   110.244034784025  NO. OF FUNC. EVALS.:   6
 CUMULATIVE NO. OF FUNC. EVALS.:        6
 NPARAMETR:  3.0000E+00  8.0000E-02  4.0000E-02  6.0000E+00  5.0000E-03
 3.0000E-01  2.0000E-04  6.0000E-03  4.0000E-01  4.0008E-01
 PARAMETER:  1.0000E-01  1.0000E-01  1.0000E-01  1.0000E-01  1.0000E-01
 1.0000E-01  1.0000E-01  1.0000E-01  1.0000E-01  1.0000E-01
 GRADIENT:   1.4640E+01 -1.6691E+01  5.6460E+01 -9.4129E+00 -1.1699E+01
 1.0623E+01 -6.3485E-01 -4.7803E+01 -9.6457E+00  5.2414E+00

 ITERATION NO.:    5    OBJECTIVE VALUE:   105.453694506861  NO. OF FUNC. EVALS.:   7
CUMULATIVE NO. OF FUNC. EVALS.:       45
 NPARAMETR:  2.8166E+00  8.0433E-02  3.8563E-02  6.6123E+00  9.7862E-03
-3.9509E-02  2.3438E-04  8.1375E-03  4.9657E-01  4.0355E-01
 PARAMETER: -5.3055E-02  1.0703E-01  5.7598E-02  1.4858E-01  1.8644E-01
-1.2545E-02  1.5794E-01  1.3452E-01  3.7714E-02  1.0431E-01
 GRADIENT:   3.5103E-01 -1.3214E+01  2.8003E+01  1.1115E+00  1.9193E+00
-5.3252E+00  7.1533E-02 -2.2091E+01 -1.4424E+00  8.2329E+00

 ITERATION NO.:   10    OBJECTIVE VALUE:   104.747695563098  NO. OF FUNC. EVALS.:   7
 CUMULATIVE NO. OF FUNC. EVALS.:       80
 NPARAMETR:  2.7362E+00  7.6935E-02  3.5982E-02  5.7362E+00  1.0840E-03
-2.9208E-01  2.3701E-04  9.4013E-03  5.1805E-01  3.8858E-01
 PARAMETER: -1.1928E-01  4.9234E-02 -2.3022E-02  7.7517E-02  2.2174E-02
-9.9574E-02  1.9498E-01  1.4956E-01 -1.7173E-01  8.5414E-02
 GRADIENT:  -1.3933E+00 -7.4632E+00  9.3144E+00  1.0280E+00  9.6395E-01
-3.1313E+00  1.0212E+00 -5.3784E+00 -2.0828E+00 -9.1477E-01

 ITERATION NO.:   15    OBJECTIVE VALUE:   104.561086502423  NO. OF FUNC. EVALS.:   7
 CUMULATIVE NO. OF FUNC. EVALS.:      116
 NPARAMETR:  2.7735E+00  7.8097E-02  3.6292E-02  5.5466E+00  5.1735E-03
-1.2911E-01  2.3941E-04  9.0947E-03  5.1461E-01  3.8760E-01
 PARAMETER: -8.8567E-02  6.8686E-02 -1.2994E-02  6.0711E-02  1.0762E-01
-4.4762E-02  1.9028E-01  1.4643E-01 -8.4016E-02  8.4157E-02
 GRADIENT:   7.7937E-03  5.7189E-03  3.0426E-02 -5.1648E-03 -1.2939E-02
 1.4490E-02  7.4602E-04 -2.5227E-02 -3.1443E-03  1.5799E-02

ITERATION NO.:   19    OBJECTIVE VALUE:   104.561067398632  NO. OF FUNC. EVALS.:   0
 CUMULATIVE NO. OF FUNC. EVALS.:      149
 NPARAMETR:  2.7739E+00  7.8129E-02  3.6307E-02  5.5498E+00  5.2393E-03
-1.2767E-01  2.3985E-04  9.1064E-03  5.1521E-01  3.8742E-01
 PARAMETER: -8.8304E-02  6.9219E-02 -1.2510E-02  6.0998E-02  1.0895E-01
-4.4249E-02  1.9095E-01  1.4652E-01 -8.3269E-02  8.4127E-02
 GRADIENT:   1.2144E-04 -1.4883E-04 -6.6257E-06 -7.5745E-05 -1.6157E-05
 1.1307E-04  6.4128E-05  1.1193E-04 -1.7655E-05  4.2676E-05
```

Note that the values of the PARAMETERs are the values of the UCP, so that at the 0th iteration, all the PARAMETERs have the value 0.1.

The first parameter (and gradient) elements correspond to the THETA elements which are not fixed. The remaining elements correspond to the OMEGA and SIGMA elements

which are not fixed, but not in a simple 1-1 manner unless OMEGA and SIGMA are constrained to be diagonal.

The PRINT option of the $ESTIMATION record determines how often iteration summaries are printed: not at all (with `PRINT=0`); only for the 0th and last iterations (with `PRINT=9999`); for the 0th iteration, for every 10th iteration thereafter, and for the last iteration (with `PRINT=10`, as illustrated above).

If a model specification file is output, then the estimates may also be seen in the original parameterization for those iterations whose summaries appear in intermediate output. These estimates may be found in file INTER in the same order and format as elsewhere in the NONMEM output. With NONMEM 7, INTER exists after the run is finished.

When the Estimation Step terminates, it reports its success or lack of it, as in this example:

```
MINIMIZATION SUCCESSFUL
NO. OF FUNCTION EVALUATIONS USED:      149
NO. OF SIG. DIGITS IN FINAL EST.:  4.7
```

Each (UCP) element of the mimimum point is determined to a number of significant digits. The number of significant digits reported is the number of significant digits in the least-well-determined element. The report "MINIMIZATION SUCCESSFUL" is issued when this number is no less than the number of significant digits requested using the SIGDIGITS option of the $ESTIMATION record. Note that this report alone does not assure that a global (or even a local) minimum point has been located; what appears to be a minimum point may be a saddle point. Nor, if a minimum point has been located, does the report alone assure that the objective function is not "flat" in a region of the point. For such assurances, one also needs to implement the Covariance Step.

REFERENCES: Guide I, section C.3.5.1
REFERENCES: Guide V, section 10.4.1

MRG DATA ITEM

MEANING: Marginal (MRG_) data item
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... MRG_ ...`

DISCUSSION:
`MRG_` labels NONMEM's marginal (MRG) data item. The marginal data item is optional. With a nonobservation record (i.e. MDV=1 or MDV=101), it controls the definition of the value of the PRED item associated with the record. Basically, 0 means the PRED item is the typical value of F, and 1 or 2 means it is the expectation of F. (Note that with NM-TRAN, F is defined by Y.) However, the marginal data item also controls the definition of the value of any PRED-defined item D that is displayed in a table or scatterplot (except items stored in the SAVE region) and associated with the record.

Values are:

0 The PRED item is the typical value of F and a displayed item D is either the typical value of the variable D or, if conditional estimates are requested, the conditional estimate of the variable D.

1 or 2

The values of the PRED and D items are determined by other things: If the Simulation Step is implemented, and ONLYSIMULATION is specified, the PRED item is the simulation expectation of F, over the simulated values of eta for all the individuals in the data set (including those individuals with no observation records), and the D item is the simulation expectation of the variable D. If conditional estimates are obtained (e.g., using the POSTHOC option), the PRED item is the posterior expectation of F, over the conditional estimates of eta for all individuals in the data set (that have some observation records), and the D item is the posterior expectation of the variable D. Otherwise, the PRED and D items are as with METHOD=0.

If the expectation is the posterior expectation:

If MRG_=1, the deletion adjustment is not used.

If MRG_=2, the deletion adjustment is used. With the deletion adjustment, the posterior expectation for a given individual is obtained by not including in the computed expectation the value of the variable evaluated at the conditional estimate of his eta.

With MRG_=1 or 2, MDV must also be 1 or 101. This use of MDV=1 does not prevent, as would ordinarily happen, the DV or RES items from being plotted.

A plot of PRED (and of any PRED-defined item not stored in the SAVE region) will only include points from records with MRG_=1, unless one explicitly partitions on MRG_.

If PREDPP is used and NM-TRAN generates MDV, MDV is set to 1 if MRG_ is 1 or 2. If PREDPP is used, NM-TRAN sets MRG_ to 0 (if it is not already 0) when EVID is not equal to 0. LP

Warning: If PREDPP is used and NM-TRAN generates EVID, it sets EVID=2 for records with MDV=1 and no dosing information. This causes MRG_ to be set to 0. When MRG_ is used, the data set should include MDV and EVID data items. The values should be MDV=1 and EVID=0 when MRG_>0.

(See **expectation example**, **expectation block**).
REFERENCES: none.

MTIME MNEXT MPAST MNOW MTDIFF

MEANING: Variables related to the model event time feature of PREDPP.
CONTEXT: Abbreviated code, verbatim code, user-supplied routines, NM-TRAN

USAGE:
`MTIME(1)=THETA(1)`

Model event times are additional PK parameters defined in the PK routine or $PK block. A model event time is not associated with any compartment, but, like an absorption lag time, defines a time to which the system is advanced. When the time is reached, indicator variables are set and a call to PK is made. At this call (and/or subsequent to this call) PK or DES or AES or ERROR can use the indicator variables to change some aspect of the system, e.g., a term in a differential equation, or the rate of an infusion. This feature may be used with any ADVAN routine. Model times are independent of non-event dose times. The following are reserved variables when used in abbreviated code.

`MTIME(i)`
: The i-th model event time. The maximum number of model event times is given by constant PCT in file SIZES (See **sizes**).

  `MTIME(i)` may be less than, equal to, or greater than `MTIME(i+1)`. Any `MTIME(i)` may be negative or have the value 0 (in which case `MPAST(i)=1` always and `MNEXT(i)=0` always; see below). If PK defines `MTIME(i)` and `MTIME(i+2)` but not `MTIME(i+1)`, then this has the same effect as defining `MTIME(i+1)=0`. PK may redefine `MTIME(i)`. An ETA may be used in the definition of `MTIME(i)`.

`MTDIFF`
: The value of `MTDIFF` is 0 when PK is called. If PK sets `MTDIFF` to a value other than 0, e.g., `MTDIFF=1`, then PREDPP will understand that with that call to PK, the values of one or more of the `MTIME(i)` have possibly been reset. It is not necessary to set `MTDIFF` at a call to PK with the first record of an individual or with a reset record.

The following are the read-only indicator variables. They are not to be set by PK.

`MNOW`
: `MNOW`=i if `MNEXT(i)=1` for some i. `MNOW`=0 otherwise.

`MNEXT(i)`
: `MNEXT(i)`=1 during the advance from the previous time to `MTIME(i)`. Otherwise, `MNEXT(i)`=0. The previous time may be an event time, a non-event time, or a model event time.

`MPAST(i)`
: `MPAST(i)`=0 until the call to PK subsequent to the one for which `MNEXT(i)`=1. At that call `MPAST(i)`=1. It then retains this value, unless `MTIME(i)` is redefined, in which case `MPAST` will be appropriately redefined as another step function.

(See **model time examples**)
(See **Model Event Time: MNOW,MPAST,MNEXT**)
(See **Model Event Time: MTIME**)
(See **Model Event Time: MTDIFF**)
(See **Circadian Example: Examples Using MTIME to Model Periodic Discontinuities in $DES**)

REFERENCES: Guide VI, section III.F.9

MULT

MEANING: MULT subroutine
CONTEXT: NONMEM utility routine

USAGE:

```
USE SIZES, ONLY: DPSIZE
REAL(KIND=DPSIZE) :: A,B
CALL MULT (A,B)
```

DISCUSSION:
MULT is a NONMEM utility routine that may be called by PRED. It allows reweighting of the objective function. That is, it allows terms A(i) and B(i) of the contribution to the (default) objective function of the ith individual to be set to values other than the defaults, which are 1. A(i) multiplies the logarithm of the determinant of the variance - covariance matrix for the ith individual and B(i) multiplies the sum of the squared weighted residuals for the ith individual.

Input argument:

`A` A variable or constant.

`B` A variable or constant.

To set A(i) and B(i) to values a and b for the ith individual, PRED should issue a single call to MULT during any one or more calls to it from NONMEM where ICALL is 2 and some data record from the ith individual is in DATREC.

At these calls, set A=a and B=b.

At least one such call to MULT must be issued for each individual. If more than one such call is issued for the same individual, then A and B should have the same values across all such calls.

MULT can be used whenever ELS is called. MULT can also be used whenever NCONTR is called and NCONTR calls CELS. NCONTR calls CELS whenever CCONTR is not called, or when CCONTR is called and CCONTR only calls CELS.

REFERENCES: Guide II, section D.4 , D.4.3
REFERENCES: Guide II, Figure 20

MU MODEL

MEANING:
CONTEXT: Abbreviated code

USAGE:

```
MU_3=LOG(THETA(3))
V=EXP(MU_3+ETA(3))
```

DISCUSSION:

The new methods in NONMEM are most efficiently implemented if the user supplies information on how the THETA parameters are associated arithmetically with the etas and individual parameters, wherever such a relationship holds. Calling the individual parameters phi, the relationship should be
phi_i=mu_i(theta)+eta(i)
for each parameter i that has an eta associated with it, and mu_i is a function of THETA. The association of one or more THETA's with ETA(1) must be identified by a variable called MU_1. Similarly, the association with ETA(2) is MU_2, that of ETA(5) is MU_5, etcetera. This is called "MU Referencing", or "MU Modelling".

Providing this information is as straight-forward as introducing the MU_ variables into the $PRED or $PK code by expansion of the code.

For a very simple example, the original code may have the lines

```
CL=THETA(4)+ETA(2)
```

This may be rephrased as:

```
MU_2=THETA(4)
CL=MU_2+ETA(2)
```

Another example would be:

```
CL=(THETA(1)*AGE**THETA(2))*EXP(ETA(5))
V=THETA(3)*EXP(ETA(3))
```

which would now be broken down into two additional lines, inserting the definition of a MU as follows:

```
MU_5= LOG(THETA(1))+THETA(2)*LOG(AGE)
MU_3=LOG(THETA(3))
CL=EXP(MU_5+ETA(5))
V=EXP(MU_3+ETA(3))
```

Note the arithmetic relationship identified by the last two lines, where MU_5+ETA(5) and MU_3+ETA(3) are expressed. This action does not change the model in any way.

If the model is formulated by the traditional typical value (TV, mean), followed by individual value, then it is straight-forward to add the MU_ references as follows:

```
TVCL= THETA(1)*AGE**THETA(2)
CL=TVCL*EXP(ETA(5))
TVV=THETA(3)
V=TVV*EXP(ETA(3)
MU_3=LOG(TVV)
MU_5=LOG(TVCL)
```

This also will work because only the MU_x= equations are required in order to take advantage of EM efficiency. It is not required to use the MU_ variables in the expression EXP(MU_5+ETA(5)), since the following are equivalent:
CL=TVCL*EXP(ETA(5))=EXP(LOG(TVCL)+ETA(5)=EXP(MU_5+ETA(5))

but it helps as an exercise to determine that the MU_ reference was properly transformed (in this case log transformed) so that it represents an arithmetic association with the eta.

An incorrect usage of MU modeling would be:

```
MU_1=LOG(THETA(1))
MU_2=LOG(THETA(2))
MU_3=LOG(THETA(3))
CL=EXP(MU_1+ETA(2))
V=EXP(MU_2+MU_3+ETA(1))
```

In the above example, MU_1 is used as an arithmetic mean to ETA(2), and a composite MU_2 and MU_3 are the arithmetic means to ETA(1), which would not be correct. The association of MU_x+ETA(x) must be strictly adhered to.

Once one or more thetas are modeled to a MU, the theta may not show up in any subsequent lines of code. That is, the only usage of that theta may be in its connection with MU. For example, if

```
CL=THETA(5)*EXP(ETA(2))
```

it can be rephrased as

```
MU_2=LOG(THETA(5))
CL=EXP(MU_2+ETA(2))
```

But later, suppose THETA(5) is used without its association with ETA(2):

```
 ...
CLZ=THETA(5)*2
```

Then THETA(5) cannot be MU modeled, because it shows up as associated with ETA(2) in one context, but as a fixed effect without association with ETA(2) elsewhere.

However, if

```
MU_2=LOG(THETA(5))
CL=EXP(MU_2+ETA(2))
CLZ=CL*2
```

Then this is legitimate, as the individual parameter CL retains the association of THETA(5) with ETA(2), when used to define CLZ. That is, THETA(5) and ETA(2) may not used separately in any other part of the model, except indirectly through CL, in which their association is retained.

Suppose you have:

```
CL=THETA(5)+THETA(5)*ETA(2)
```

One should see this as:

```
CL=THETA(5)*(1+ETA(2))
```

So the way to MU model this is:

```
MU_2=1.0
CL=THETA(5)*(MU_2+ETA(2))
```

Which would mean that in the end, THETA(5) is not actually MU modeled, since MU_2 does not depend on THETA(5). One would be tempted to model as follows:

```
MU_2=THETA(5)
CL=MU_2+MU_2*ETA(2)
```

But this would be incorrect, as MU_2 and ETA(2) may not show up together in the code except as MU_2+ETA(2) or its equivalent. Thus, THETA(5) cannot be MU modeled. In such cases, remodel to the following similar format:

```
CL=THETA(5)*EXP(ETA(2))
```

So that THETA(5) may be MU modeled as:

```
MU_2=LOG(THETA(5))
CL=EXP(MU_2+ETA(2))
```

Sometimes, a particular parameter has a fixed effect with no random effect, such as:

```
Km=THETA(6)
```

with the intention that Km is unknown but constant across all subjects. In such cases, the THETA(6) and Km cannot be Mu referenced, and the EM efficiency will not be available in moving this Theta. However, one could assign an ETA to THETA(5), and then fix its OMEGA to a small value, such as 0.0225 =0.15ˆ2 to represent 15% CV, if OMEGA represents proportional error. This often will allow the EM algorithms to efficiently move this parameter, while retaining the original intent that all subjects have similar, although not identical, Km's. Very often, inter-subject variances to parameters were removed because the FOCE had difficulty estimating a large parametered problem, and so it was an artificial constraint to begin with. EM methods are much more robust, and are adept at handling large, full block OMEGA's, so you may want to incorporate as many etas as possible when using the EM methods.

You should Mu reference as many of the THETA's as possible, except those pertaining to residual variance (which should be modeled through SIGMA whenever possible). If you can afford to slightly change the theta/eta relationship a little to make it MU referenced without unduly influencing the model specification or the physiological meaning, then it should be done.

When the arithmetic mean of an ETA is associated with one or more THETA's in this way, EM methods can more efficiently analyze the problem, by requiring in certain calculations only the evaluation of the MU's to determine new estimates of THETAs for the next iteration, without having to re-evaluate the predicted value for each observation, which can be computationally expensive, particularly when differential equations are used in the model. For those THETA's that do not have a relationship with any ETA's, and therefore cannot be MU referenced (including THETA's associated with ETAS whose OMEGA value is fixed to 0), computationally expensive gradient evaluations must be made to provide new estimates of them for the next iteration.

There is additional increased efficiency in the evaluation of the problem if the MU models are linear functions with respect to THETA. Recalling one of the previous examples above, we could re-parameterize THETA such that

```
MU_5=THETA(1)+THETA(2)*LOG(AGE)
CL=EXP(MU_5+ETA(5))
MU_3=THETA(3)
V=EXP(MU_3+ETA(3))
```

This changes the values of THETA(1) and THETA(3) such that the re-parameterized THETA(1) and THETA(3) are the logarithm of the original parameterization of THETA(1) and THETA(3). The models are identical, however, in that the same maximum likelihood value will be achieved. The only inconvenience is having to anti-log these THETA's during post-processing.

The added efficiency obtained by maintaining linear relationships between the MU's and THETA's is greatest when using the SAEM method and the MCMC Bayesian method. In the Bayesian method, THETA's that are linearly modeled with the MU variables have linear relationships with respect to the inter-subject variability, and this allows the Gibbs sampling method to be used, which is much more efficient than the Metropolis-Hastings (M-H) method. By default, NONMEM tests MU-THETA linearity by determining if the second derivative of MU with respect to THETA is nearly or equal to 0. Those THETA

parameters with 0 valued second derivatives are Gibbs sampled, while all other THETAS are M-H sampled. In the Gibbs sampling method, THETA values are sampled from a multi-variate normal conditional density given the latest PHI=MU+ETA values for each subject, and the samples are always accepted. In M-H sampling, the sampling density used is only an approximation, so the sampled THETA values must be tested by evaluating the

Some additional rules for MU referencing are as follows:

1) As much as possible, define the MU's in the first few lines of $PK or $PRED. Do not define MU_ values in $ERROR. Have all the MU's particularly defined before any additional verbatim code, such as write statements. NMTRAN produces a MUMODEL2 subroutine based on the PRED or PK subroutine in FSUBS.F90, and this MUMODEL2 subroutine is frequently called with the ICALL=2 settings, more often than PRED or PK. The fewer code lines that MUMODEL2 has to go through to evaluate all the MU_s' the more efficient.

2) Whenever possible, have the MU variables defined unconditionally, outside IF...THEN blocks.

3) Time dependent covariates cannot be part of the MU_ equation. For example

   ```
   MU_3=THETA(1)*TIME+THETA(2)
   ```

   should not be done. Or, consider

   ```
   MU_3=THETA(2)/WT
   ```

   Where WT varies with time. This would also not be suitable. However, we could phrase as

   ```
   MU_3=THETA(2)
   CL=WT*(MU_3+ETA(3))
   ```

   is fine, where MU_3 represents a population mean clearance per unit weight, which is constant with time, and more universal among subjects, whereas CL is the non-wieght normalized clearance, than depends on a person's weight, which could vary with time as well. The MU variables may vary with inter-occasion, but not with time.

4) Starting with NONMEM 7.2, NMTRAN's CHECKMU subroutine attempts to look for errors in MU modeling. If it appears that there may be errors, then there are messages such as

   ```
   (MU_WARNING 13) MU_001: DOES NOT HAVE ADDITIVE ASSOCIATION WITH ETA(001)
   ```

   Such warnings do not affect the outputs from NMTRAN. FSUBS is generated as usual. Sometimes the warnings may be ignored (see "Model parameters as log t-Distributed", below.) Sometimes warnings may not be generated when they should be. Thus, the user must pay close attention to following the rules.

   Option `NOCHECKMU` of the $ABBR record may be used to prevent NM-TRAN from attempting to check the MU model statements.

Examples show examples of MU modeling for various problem types. Study these examples carefully. When transposing your own code, begin with simple problems and work your way to more complex problems.

At this point one may wonder why bother inserting MU references in your code. MU referencing only needs to be done if you are using one of the new EM or Gibbs sampling methods to improve their efficiency. The EM methods may be performed without MU references, but it will be several fold slower than the FOCE method, and the problem may not even optimize successfully. For simple two compartment models, the new EM

methods are slower than FOCE even with the MU references. But, for 3 compartment models, or numerical integration problems, the improvement in speed by the EM methods, properly MU modeled, can be 5-10 fold faster than with FOCE.

Example 6 described at the end of the SIGL section is one example where importance sampling solves this problem in 30 minutes, with R matrix standard error, versus FOCE which takes 2-10 hours or longer, and without even requesting the $COV step. So, for complex PK/PD problems that take a very long time in FOCE, it is well worth putting in MU references and using one of the EM methods, even if you may need to rephrase some of the fixed/random (theta/eta) effects relationships. In addition, FOCE is a linearized optimization method, and is less accurate than the EM and Bayesian methods when data are sparse or when the posterior density for each individual is highly non-normal.

Model parameters as log t-Distributed in the Population

Sometimes one may suspect that PK/PD model parameters are actually log t-distributed among the population, with degrees of freedom NU, instead of the usual log normal distributed.

See **INTRODUCTION TO NONMEM 7, MU Referencing**

An example of simulation and analysis of such data is given as

..\examples\tdist6_sim.ctl
..\examples\tdist6.ctl:
..\examples\tdist7.ctl

Note that constructions such as

```
CL=EXP(MU_1+ETA(1)*SQRT((EXP(CLR)-1.0)/CLR))
```

violate the strict MU_x+ETA(x) rule recommended for EM analysis, because the term `SQRT((EXP(CLR)-1.0)/CLR)` is multiplied by ETA(1). NM-TRAN will generate a number of MU_WARNING messages. Nonetheless for this example, the importance sampling works quite well, and the MU_WARNING messages may be ignored.

REFERENCES: Guide Introduction_7

MXSTEP

MEANING: PREDPP-PK global variables
CONTEXT: User-supplied routines

USAGE:

```
USE PRDATA, ONLY: MXSTEP=>MXSTP0  (ADVAN9)
USE PRDATA, ONLY: MXSTEP=>MXSTP01 (ADVAN13,ADVAN14,ADVAN15,ADVAN16,ADVAN17,AI
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) MXSTP0
INTEGER(KIND=ISIZE) MXSTP01
```

DISCUSSION:

MXSTEP

The maximum number of integration steps for ADVAN9 and ADVAN13 and ADVAN14 and ADVAN15 and ADVAN16 and ADVAN17 and ADVAN18.

For ADVAN13, ADVAN14, ADVAN15, ADVAN16, ADVAN17, and ADVAN18, the default value of MXSTP01 is set to 10000 in resource\PRDATA.f90

For ADVAN9, the default value of MXSTP0 is set to the largest possible integer value 2147483647 in resource\PRDATA.f90, so MXSTEP can only be used to set a smaller value.

This variable is only a reserved variable for ADVAN9 and ADVAN13 and ADVAN14 and ADVAN15 and ADVAN16 and ADVAN17 and ADVAN18.

REFERENCES: Guide Introduction_7

NCONTR

MEANING: NCONTR subroutine
CONTEXT: NONMEM utility routine

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of NONMEM code and previous examples that may be available from advanced users.

USAGE:

```
USE SIZES, ONLY: ISIZE,DPSIZE
INTEGER(KIND=ISIZE) :: IER1,IER2,L2R
REAL(KIND=DPSIZE)   :: CNT
 ...
CALL NCONTR (CNT,IER1,IER2,L2R)
```

DISCUSSION:
NCONTR is a NONMEM utility routine. It computes the normal-based contribution to the objective function from the data from a single individual ("L1") record.

NCONTR may be called by a user-supplied CONTR. NCONTR can be used when eta is from a normal distribution with mean 0 and variance-covariance OMEGA. It can be used with population data. It should not be used with single-subject data.

Output argument:

`CNT`
Contribution to -2log likelihood for data from the L1 record

`IER1`
IER1 return codes:

1 HESSIAN OF OBJ. FUNC. FOR COMPUTING SOME CONDITIONAL ESTIMATE IS SINGULAR, IF LAPLACIAN IS NOT USED, OR NONPOSITIVE DEFINITE, IF LAPLACIAN IS USED.

2 ETA-DERIVATIVE OF SOME CONDITIONAL -2LOG LIKELIHOOD VALUE IS TOO LARGE

3 NUMERICAL HESSIAN OF OBJ. FUNC. FOR COMPUTING SOME CONDITIONAL ESTIMATE IS NON POSITIVE DEFINITE

11 IF CCONTR IS NOT USED AND F = PREDICTION, SOME INTRAINDIVIDUAL VARIANCE-COVARIANCE IS SINGULAR.

IF CCONTR IS NOT USED AND F = LIKELIHOOD, SOME CONDITIONAL LIKELIHOOD VALUE IS NEGLIGIBLE.

12 IF CCONTR IS NOT USED AND F = PREDICTION, WEIGHTED SUM OF "SQUARED" INDIVIDUAL RESIDUALS IS INFINITE.

10+
IF CCONTR IS USED, 10 IS ADDED TO A NONZERO VALUE OF IER1 FROM CCONTR.

`IER2`
0 if error-recovery is to be implemented when IER1 is nonzero.
1 if NONMEM is to stop when IER1 is nonzero.

`L2R`
Index of L2 record giving rise to error condition.

Other inputs:
NCONTR obtains all its inputs from NONMEM read-only global variables.

REFERENCES: None.

NEFF,NEFFI

MEANING: NEFF and NEFFI programs
CONTEXT: NONMEM run

DISCUSSION:
With NONMEM 7.4, the NEFF and NEFFI utility programs are used to analyze the quality of a MCMC run. NEFF stands for "Number of EFFective samples". NEFFI stands for "NEFF for individual".

For details and examples,
See **INTRODUCTION TO NONMEM 7: NEFF and NEFFI Utility Programs**

REFERENCES: Guide Introduction_7

NEW INDIVIDUAL INDICATOR: NEWIND

MEANING: PREDPP read-only global variables
CONTEXT: User-supplied PK and ERROR routines

USAGE:
```
USE PROCM_INT, ONLY: NEWIND=>PNEWIF
```

GLOBAL DECLARATION:
```
INTEGER(KIND=ISIZE) :: PNEWIF
```

DISCUSSION:

`NEWIND`

`NEWIND` is of interest only when ICALL=2, 4, 5, or 6. It is the same as the `NEWIND` argument passed by NONMEM to PREDPP (with the first data record of the current event record). Values are:

`NEWIND`=0: First event record of the data set.

`NEWIND=1`: First event record of the data set, THETA value does not differ from value at last call with this record, and PRED is nonrecursive (See **Recursive PRED Indicator**), or,

`NEWIND`=1: First event record of a subsequent individual record.

`NEWIND`=2: Subsequent event record of an individual record.

Location prior to NONMEM 7: procm1

REFERENCES: Guide VI, section III.I

NINDR INDR1 INDR2

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

USAGE:

```
USE ROCM_INT,  ONLY: NINDR=>NINDOBS,INDR1=>IDXOBSF,INDR2=>IDXOBSL
```

GLOBAL DECLARATION:

```
REAL(KIND=ISIZE):: NINDOBS,IDXOBSF,IDXOBSL
```

DISCUSSION:

`NINDR`
: The number of individual records in the data set containing an observation record.

`INDR1`
: The index of the first individual record in the data set containing an observation record.

`INDR2`
: The index of the last individual record in the data set containing an observation record.

These variables may be used as right-hand quantities in $PRED, $PK, $ERROR, $INFN and $MIX abbreviated code.

Location prior to NONMEM 7: rocm46

REFERENCES: None.

NM-TRAN OUTPUT FILES

MEANING: NM-TRAN output files
CONTEXT: NONMEM input file

DISCUSSION:

NM-TRAN produces several output files, several of which are input files to NONMEM itself.

In the first column, the files are listed by the default file names supplied by constants in NM-TRAN's ABLOCK routine. (Some installations may use different names for these files.)

The second column lists the inputs to NM-TRAN that contribute to these files.

The third column indicates how the files are used.

Note that file FSIZES is a convenient reference for users to view, but is not used. SUBROUTINE FSIZESR in FSUBS is what is actually used during the NONMEM run.

File name	Relevant portion of NM-TRAN inputs	How file is used
FCON	All	NONMEM (control stream)
FDATA	Data file, $DATA, $INPUT	NONMEM (data file)
FSTREAM	Filenames from $DATA, $MSFI, $MSFO, $TABLE	NONMEM (file control stream)
FSUBS	Abbreviated code	Generated FORTRAN subroutines
FREPORT	$SUBROUTINES	Input to whatever generates the commands that create the NONMEM executable (e.g., the NMLINK routine).
FORIG	Abbreviated code, $ABBR	When $ABBR REPLACE is used, this file contains the original abbreviated code.
FREPL	Abbreviated code, $ABBR	When $ABBR REPLACE is used, this file contains the post-replacement abbreviated code.
FWORK	Data file	NM-TRAN (work file)
FWARN	All	Warning and informational messages.
FSIZES	Abbreviated code, $THETA, $OMEGA, $SIGMA, $SIZES, SIZES.f90	Dynamic array sizes for NONMEM
prsizes	Abbreviated code, $THETA, $OMEGA, $SIGMA, $SIZES, SIZES.f90	Static array sizes for PREDPP

With NONMEM 7, files FSUBS2 and FSUBS_MU are used by NMTRAN as work files during the creation of FSUBS.

REFERENCES: Guide III, section V.6.0
REFERENCES: Guide Introduction_7

NMFE UTILITY PROGRAM

MEANING: The name of a computer program
CONTEXT: NONMEM support utility routine

USAGE:

```
nmfe74 controlfile reportfile [-background]
       [-prsame] [-prcompile]
       [-prdefault] [-prseq]
       [-tprdefault]
       [-trskip] [-nobuild]
       [-maxlim=1] [-maxlim=2] [-maxlim=3]
       [-rundir=directoryname] [-runpdir=directoryname]
       [-parafile=filename]
       [-parafprint=n]
       [-nmexec=filename] [-xmloff]
       [-licfile=filename]
       [-locfile=filename]
       [-simparon] [-flushtime>=0.0]
```

Default, if used in the run directory:
nmfe74 controlfile reportfile
-prsame (if possible; else -prcompile)
-rundir=run
-runpdir=temp_dir -nmexec=nonmem
-licfile=../license/nonmem.lic

Options may need to be in quotes, e.g.,

```
nmfe74 CONTROL5 REPORT5.res "-prsame"
```

The name of the command indicates the version of NONMEM, e.g. nmfe74 denotes NONMEM 7.4.0. The command will be shown generically as nmfe7 in this help file. For details specific to an earlier release than NONMEM 7.4.0, see the help file for that release.

OPTIONS:

controlfile
: Name of the input file containing the NM-TRAN control stream.

reportfile
: Name of the output file into which NONMEM writes the results. Also called "the standard results" file.

`-prcompile,-prsame,-prdefault,-tprdefault`
: By default, nmfe7 copies the required PREDPP routines from the nonmem ..\pr and ..\resource directories into a subfolder temp_dir of the current directory and compiles the routines there. (Some of these routines are needed even when $PRED is used.) This is referred to as "pr recompilation". The resulting object files are then linked with NONMEM, and the nonmem executable is created. In a subsequent run, nmfe7 will skip the pr recompilation if the same ADVAN/TRANS (or a $PRED) are used, and the size of the problem (e.g., size of OMEGA and SIGMA) has not

changed.

`-prcompile`
Force pr recompilation, in case the run does not appear to execute properly when no pr recompilation occurs.

`-prsame`
Force the skipping of pr recompilation. Used when nmfe7 detects a change in the previous run from the present run, but the user is convinced there is no significant change. Uses the compiled files from the previous run.

`-prdefault`
Do not recompile any routines other than FSUBS. Use the precompiled routines in ..\pr and ..\resource directories. May be used if it is sufficient to use default values in sizes.f90 for the various parameters (see ..\util\prsizes.f90 for the list of those that are used.). Any NONMEM parameters that are not used (that is, those not listed in prsizes.f90) will still be dynamically sized.

`-tprdefault`
Tests if is okay to do `-prdefault`. If it is not okay, then it will test if there are compiled pr routines in temp_dir directory,and if they are appropriate for the present model. If so, it will not recompile those pr routines, and use those that are already there. If not, it will perform a fresh PREDPP compile. May be combined with either `-prcompile` or `-prsame`.

`-tprdefault -prcompile`
Tests if is okay to do `-prdefault`. If it is not okay, then it will perform a fresh compile of the particular ..\pr routines, even if there were appropriate ones in temp_dir already. (`-prcompile` forces a compile regardless of whether it was necessary).

`-tprdefault -prsame`
Tests if is okay to do `-prdefault`. If it is not okay, then it will use those in temp_dir, assuming that they are appropriate for the present problem.

`-prseq`
As of nm75, the option `-prseq` is available to cause the PREDPP files to be compiled sequentially, rather than in parallel (default). Sometimes the parallel method of compiling may actually take longer, or the compiler license may limit the number of simultaneous instances of the compiler that are permitted to be operating at once.

`-trskip`
Requests that the NMTRAN step be skipped. The `-trskip` option is useful if you wish to modify FSUBS.f90 created by a previous run, and insert extra debug lines into FSUBS.f90, and prevent your modified FSUBS.f90 from being over-written by NMTRAN (it will still be compiled).

`-trskip` and any one of `-prskip`, `-prnoskip`, or `-prstatic` options may be used together, or `-trskip` may be used by itself.

`-nobuild`
Prevents a new nonmem executable from being built, particularly useful for a series of nonmem runs during boostrap procedures.

`-maxlim=[1|2|3]`
With maxlim=1, then LIM1, LIM3, LIM4, LIM13, and Lim15 (those used during estimation, and therefore by workers in a parallelization problem), will be set to the size needed to assure no buffer files are used, and everything is stored in memory,

for the particular prolem. With -maxlim=2, then LIM1, LIM2, LIM3, LIM4, LIM5, LIM6, LIM7, LIM8, LIM13, LIM15, and LIM16 are also sized to what is needed to assure that buffer files are not needed. With -maxlim=3, then MAXRECID will also be sized, to MAXDREC, the largest number of records in any individual. With maxlim=3, it is preferred to also use -tprdefault, or -prcompile, but not -prdefault, as NMTRAN's optional resizing of the PREDPP size parameter MAXRECID may conflict with the -prdefault option

`-rundir=`directoryname
Specifies a directory for the NONMEM run if it is different from the present working directory. The directory must exist, and must contain all necessary input files, such as the controlfile, msf files, and data files.

`-runpdir=`directoryname
Specifies a directory for the compile if it is different from temp_dir. This is useful if you are repeatedly going between two or more problems, so that often they need to be pr recompiled, and you want to save time. Specify a unique temporary directory for the compilation for each problem.

`-parafile=`filename
Name of the "parallel file" (the parallelization profile) that controls parallelization (distributed computing). Default file name if not specified: parallel.pnm. For details on the content, see the INTRODUCTION TO NONMEM 7. Versions prior to NONMEM 7.2 do not support parallel computing.

`-parafprint=n`
Sometimes the parallelization log files can become very large during the $EST and $COV steps. Each of these records have parafprint options to control the print intervals. Or, you can control the print iterations globally with the -parafprint option at the command line.

`-locfile=`filename
Name of a file to override the default file called "nmloc" or "nmloc.bat". This is the location file, which gives the path for compiler or MPI system. The nmfe7 script will display a statement as to what path it uses. E.g., "Pathlist information for compiler and MPI systems are located in nmpathlist.txt" where nmpathlist.txt describes which verson of nmloc is being used.

`-nmexec=`filename
Specifies an alternate name for the NONMEM executable instead of the default nonmem.exe (windows) or nonmem (Linux). Note that when parallelization is used, this name must be specified in the pnm file. Example:

```
nmfe7 controlfile reportfile -nmexec=nonmem2 -parafile=mpi2.pnm
```

Suppose mpi2.pnm is based on mpilinux_onecomputer.pnm, which contains

```
1:mpirun -wdir "$PWD" -n 1 ./nonmem  $*
2:-wdir "$PWD/wrk_mpi" -n 1 ./nonmem <control_stream> <licfile>
```

Then mpi2.pnm must be changed as follows:

```
1:mpirun -wdir "$PWD" -n 1 ./nonmem2  $*
2:-wdir "$PWD/wrk_mpi" -n 1 ./nonmem2 <control_stream> <licfile>
```

`-background`
If "-background" is present NONMEM does not poll the terminal for ctrl characters (See below.)

`-xmloff`
Turns off production of the XML output file root.xml, where root is the root name of

the control stream file. This may speed up fast computational problems such as $SIM on simple models and small data sets.

`-simparon`
Turn on parallelization during simulation step.

`-flushtime>=0.0`
File buffer contents are flushed to file not more frequently than flushtime seconds (1.0 second) is default. Floating number allowed.

DISCUSSION:

NMFE7.bat and NMFE7 are front-end tools for running NONMEM 7. Their use is optional. NMFE7.BAT is a MS-DOS batch program. NMFE7 is a UNIX C-shell script. Both are supplied on the NONMEM 7 distribution medium.

They should be placed in a directory in the user's path. Both make use of the support utilities nmlink7 and compile7, which are also supplied on the NONMEM 7 distribution medium.

Both execute the following steps, stopping after any step in which errors occur:

1) NM-TRAN processes the NM-TRAN control stream, which is found in a file whose name is given as the first command-line argument. With NONMEM 7.2 and higher, both lower and upper case may be used for all user-defined and reserved words in the control stream. With NONMEM 7.3 and higher, & may be used at the end of any line of the control stream to indicate that the line is to be continued, including control records as well as abbreviated code. Lines may be longer than 160 characters long. The maximum length is given by FSD in resource/SIZES.f90. (FSD=67000 with NONMEM 7.3). This step is omitted if option -trskip is used.

2) NMLINK7 creates the file LINK.LNK (a list of object files to be included in the NONMEM executable). With NONMEM 7, compile7 creates the file compile7.lnk (a list of PREDPP routines to be re-compiled with the current array sizes) and a work directory temp_dir is created. All files listed in compile7.lnk are copied to temp_dir and compiled. They are then copied to the current run directory. The recompilation of PREDPP routines can take a noticeable amount of time. This step is skipped if option -prsame or -prdefault is used.
(See **$sizes.ctl**).

3) If a file FSUBS.f90 of generated and/or user-supplied FORTRAN code was created by NM-TRAN, it is compiled by the FORTRAN 90 compiler.

4) Utility program nonmem_mpi.exe is called to check if parallel processing is requested, what transfer type is to be used, and make sure the parallel file is syntactically okay. Either of two sets of messages will appear at the console:

```
Exit status = 1
  IN MPI
```

This indicates that parallel proccesing using the MPI method is requested. MPI libraries will be included in the NONMEM executable.

```
Exit status = 0
IN REGULAR/FILE TRANSFER
```

This indicates that no parallel processing was requested, or parallel processing uses the FPI (file transfer) method. No MPI library is needed.

5) The NONMEM executable nonmem.exe (nonmem in UNIX) is created.

6) The NONMEM executable is run. It is passed an option -licfile, which tells it where to find the NONMEM license file. Typically, this is the file nonmem.lic in the subdirectory license of the directory in which NONMEM was installed.
First, arrays are allocated dynamically, according to information supplied by NMTRAN in files FSIZES.
(See **$sizes**).
Processing may be performed as a single CPU process, or distruted among multiple cores or nodes, as specified in the parallel file.
(See **parallel**).
The output file whose name is given as the second command-line argument is created. During the run, NONMEM output can be found in the file OUTPUT. When the run terminates, file OUTPUT is copied to the output file and then removed.

While NONMEM 7 and higher is running, the following may be used if "-backgound" was not present:

ctrl-J
Turn console iteration printing on/off during the Estimation Step (Default is on).

ctrl-K
Stop the Estimation Step, which completes its output, and goes on to next mode or estimation method.

ctrl-E
Exit program gracefully.

ctrl-T
Monitor the progress of each individual during an estimation by toggling ctrl-T. Wait 15 seconds or more to observe a subject's ID, and individual objective function value. It is also good to test that the problem did not hang if a console output had not been observed for a long while.

The signal program may be used in situations where NONMEM does not respond to the ctrl key.
(See **signal**).

EXAMPLES OF USAGE:

```
nmfe7 CONTROL5 REPORT5.res
nmfe7 CONTROL5 REPORT5.res -background >& consout &
```
(Trailing & is used in UNIX to run the command in the background.)

Other files are created by nmfe7 and NONMEM:

FSUBS_MU.F90
Contains SUBROUTINE MUMODEL2. This subroutine contains only the code that is needed to compute any MU parameters (MU_1, MU_2, etc.) that were defined in $PK or $PRED abbreviated code. It is also included in FSUBS.

FSUBS and FSUBS.f90
These are identical. They contain both the content of FSUBS as it was in previous versions of NONMEM (generated subroutines PK, PRED, MODEL, etc., plus any user-supplied code), and also subroutine MUMODEL2. Note that user-supplied code must be in Fortran 90 format.

With NONMEM 7 and higher, additional output files are created, with names controlfile.xxx, where xxx is ext, cov, etc.
(See **additional_output_files**).
(See **raw_output_file**).

REFERENCES: Guide III, section IV.1.0, V.7.0
REFERENCES: Guide Introduction_7

NMTEMPLATE

MEANING: NMTEMPLATE program
CONTEXT: NONMEM run

USAGE:

`..\util\nmtemplate` source-template-file destination-file var1=val1 var2=val2 ...

where `var1=val1` is a variable name, and value to substitute in the template file. The variable `var1` must in turn appear as `<var1>` in the template file, and is case sensitive. Similarly for var2, var3, etc.

DISCUSSION:

The utility program nmtemplate in the ..\util directory will perform variable substitution on appropriately tagged control stream template files, and produce executable control stream files.

SAMPLE:

```
nmtemplate nmtemp.nmt nmtemp.ctl NMID=47 TH1=1.7 TH2=1.4 TH3=0.8 TH4=2.0
```

The lines of interest in file ..\util\nmtemp.nmt are:

```
$DATA nmtemp2.csv IGNORE=C ACCEPT=(ID.EQ.<NMID>)
$THETA <TH1> <TH2> <TH3> <TH4>
```

Note that <NMID> is to be replaced with a particular NONMEM ID number by nmtemplate, and the <THX> are to be replaced with specific values of thetas. The resulting file nmtemp.ctl will have the various values substituted into the various <> placeholders, and is ready to be read by NMTRAN:

nmfe7 nmtemp.ctl nmtemp.res

In the above nmtemp.nmt example, because FNLETA=2, then NONMEM will simply evaluate the IPRED values using the inputted etas from the $ETAS record without performing an estimation.

Another example template file is example6.nmt listed in the ..\util directory, that you may inspect for other ideas.

Actually, nmtemplate is a general variable substitution program, and can process any text file in the manner shown above. Consider a FINEDATA control stream file template (nmtemp.fnt):

```
$INPUT C SET ID JID TIME  DV=CONC AMT=DOSE RATE EVID MDV
       CMT CLX V1X QX V2X SDIX SDSX
$DATA nmtemp.csv IGNORE=C
$FINEDATA AXIS=TIME(LIN) TSTOP=<TSTOP> TSTART=<TSTART>
          NEVAL=<NEVAL> FILE=nmtemp2.csv
```

in which the tstart, tstop, and neval parameters are to be inserted:

```
nmtemplate   nmtemp.fnt   nmtemp.fnd   TSTART=0   TSTOP=100
NEVAL=200
```

resulting in the FINEDATA control stream file nmtemp.fnd:

```
$INPUT C SET ID JID TIME  DV=CONC AMT=DOSE RATE EVID MDV CMT
       CLX V1X QX V2X SDIX SDSX
$DATA nmtemp.csv IGNORE=C
$FINEDATA AXIS=TIME(LIN) TSTOP=100 TSTART=0 NEVAL=200
```

```
FILE=nmtemp2.csv
```

Note that only words that match the variable list at the nmtemplate command line, and have enclosing brackets <>, will be replaced with the suggested values. The values may also be text with no spaces in them.

Nmtemplate may be used with a DOS patch script (which could also be converted to an R/S-PLUS script or function).

Another feature of nmtemplate is that the user may request a random number to be generated to serve as a value, by referring to ˜R(a1,a2,a3). R(a1,a2,a3) is a special function of nmtemplate, which obtains a uniform random variate between a1 and a2. If a seed a3 is given that is not 0, it means to initialize the seed. The initialization should be done once in a series.

For example, the following line sets the seed:

`nmtemplate wexample12.nmt dummy.ctl SAMPLE=˜R(1,10000,113345)`
to be substituted wherever <SAMPLE> shows up in the template file with a throw-away result file dummy.ctl. The template file wexample12.nmt may contain:
`$EST METHOD=CHAIN FILE=wexample12.txt NSAMPLE=0 ISAMPLE=<SAMPLE>`
and the resulting files will contain random ISAMPLE values.

See Guide Introduction_7 for details and examples.

REFERENCES: Guide Introduction_7

NON-ACTIVE ETA LIST FOR PRED

MEANING: NONMEM read-only global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPRD_INT, ONLY NAETA,LVOUT
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: LVR
INTEGER(KIND=ISIZE) :: NAETA,LVOUT(LVR)
```

DISCUSSION:

`NAETA`
: The number of positions in LVOUT that are set to 1.

`LVOUT`
: When `LVOUT(i)=1`, NONMEM is ignoring partial derivatives with respect to eta(i) with the current call to PRED.

Location prior to NONMEM 7: rocm15

REFERENCES: None.

NONMEM CONTROL STREAM

MEANING: NONMEM control records
CONTEXT: NONMEM input file

DISCUSSION:
NONMEM control records contain the instructions to NONMEM. The set of control records is called the NONMEM control stream. These are different from NM-TRAN control records.

When NM-TRAN is used, NM-TRAN creates the NONMEM control stream in a file named FCON. When NM-TRAN is not used, the user must create it himself.

Each NONMEM control record begins with a 3 or 4-character preface (record label) which identifies the record. They are listed here in the same order in which they must appear in the NONMEM control stream. For details of the contents of the records, see NONMEM Users Guide V "Introductory Guide", Appendix 4 - NONMEM Control Records (FCON).

Record type	Prefix	Corresponding NM-TRAN record
FILE	FILE	none
SUPER PROBLEM	SUPR	$SUPER
PROBLEM	PROB	$PROBLEM
DEFAULT	DFLT	$DEFAULT
Data set specification records		
DATA	DATA	$DATA
ITEM	ITEM	$INPUT
INDEX	INDX	$INDEX
LABEL	LABL	$INPUT,$TABLE,$SCATTER
EXTRA EVID	XVID	$INPUT
FORMAT	FORM	$DATA
Model specification records		
FIND	FIND	$MSFI
VERSION	MSFV	$MSFI
STRUCTURE	STRC	$THETA, $OMEGA, $SIGMA
LEVEL	OLEV	$LEVEL
Initial estimate records		
THETA CONSTRAINT	THCN	$THETA
THETA	THTA	$THETA
LOWER BOUND	LOWR	$THETA
UPPER BOUND	UPPR	$THETA
DIAGONAL	DIAG	$OMEGA, $SIGMA
BLOCK SET	BLST	$OMEGA, $SIGMA
ETA VALUES	ETA	$ETAS
PHI VALUES	PHI	$PHIS
CHAIN	CHN	$CHAIN
CHAIN FILE	CFIL	$CHAIN

CHAIN DELIMITER	CDLM	$CHAIN
CHAIN Order	ORDR	$CHAIN
CHAIN PARAFILE	CHFL	$CHAIN (not used)
OMEGA VALUES	ANNL	$ANNEAL

Task specification records

SIMULATION	SIML	$SIMULATION
RANDOM SOURCE	SORC	$SIMULATION
SIMULATION PARAFILE	SFIL	$SIMULATION
ESTIMATION	ESTM	$ESTIMATION
ESTIMATION	BEST	$ESTIMATION
ESTIMATION PARAFILE	BFIL	$ESTIMATION
ESTIMATION PARAFILE	PFIL	$ESTIMATION
ESTIMATION FPARAFILE	FFIL	$ESTIMATION
ZERO	ZERO	$ESTIMATION
OMIT	OMIT	$OMIT
ESTIMATION GRID	GRID	$ESTIMATION
NONPARAMETRIC	NONP	$NONPARAMETRIC
COVARIANCE	COVR	$COVARIANCE
COVARIANCE	COVT	$COVARIANCE
COVARIANCE PARAFILE	CPAR	$COVARIANCE
COVARIANCE SIRPARAFILE	SPAR	$COVARIANCE
TABLE	TABL	$TABLE
TABLE PARAFILE	PPAR	$TABLE
SCATTERPLOT	SCAT	$SCATTERPLOT

Output specification records

ESTIMATION DELIMITER	BDLM	$ESTIMATION
ESTIMATION MUM	BMUM	$ESTIMATION
ESTIMATION GRD	BGRD	$ESTIMATION
ESTIMATION ORDER	ORDR	$ESTIMATION
LFORMAT	FRML	$TABLE
RFORMAT	FRMR	$TABLE
ADDITIONAL DIAGNOSTIC LABELS	LBW1	$TABLE,$SCATTER
SYMBOLIC LABELS	LTHT	$ABBR REPLACE
SYMBOLIC LABELS	LETA	$ABBR REPLACE
SYMBOLIC LABELS	LEPS	$ABBR REPLACE

REFERENCES: Guide I, section B.2
REFERENCES: Guide V, Appendix 4

NONMEM ERROR MESSAGES

MEANING: Error messages
CONTEXT: NONMEM output
USAGE:

DISCUSSION:
This is not a complete list. It includes only commonly seen error messages.

Section I. Errors in the control stream

`FILE RECORD MISSING`

There is something wrong with the NONMEM control file. Probably it is an NM-TRAN control file (starting with $PROBLEM), rather than the FCON file produced by NM-TRAN.

`TOT. NO. OF OBSERVATIONS IN INDIVIDUAL REC NO. 1 (IN INDIVIDUAL REC ORDERING) EXCEEDS 50`
`SEE INSTALLATION GUIDE`

There are too many observation records in some individual record. To increase the limit beyond 50, NONMEM must be recompiled. see Guide III, Section 2.7.

`INITIAL ESTIMATE OF OMEGA HAS A NONZERO BLOCK WHICH IS NUMERICALLY NOT POSITIVE DEFINITE`

Possible cause: initial estimates of the variance and covariance terms in a block of OMEGA are not appropriate. Try other initial estimates. (Hint: it helps of the covariance estimates are small relative to the variance estimates.)

`USER CCONTR ROUTINE NOT USED, BUT THERE ARE NO EPSILONS, AND ETAS ARE TO BE ESTIMATED`

This can happen when POSTHOC etas are requested, but the data are single-subject data. The user may have included the POSTHOC option in error.

`UNABLE TO OBTAIN A SATISFACTORY INITIAL ESTIMATE OF VARI-ANCE-COVARIANCE COMPONENTS`
`BLOCKS IN BLOCK SET 2 OF OMEGA ARE NUMERICALLY NOT POSITIVE DEFINITE`

This message is from the Initial Estimates Step. It identifies certain parameters whose initial estimates could not be obtained.

Part II. Error in NONMEM tasks

`NUMBER OF CALLS TO SIMETA EXCEEDS NO. OF DATA RECORDS FOR INDIVIDUAL 1 (IN INDIVIDUAL RECORD ORDERING)`

During the Simulation Step, an excessive number of calls to SIMETA have occurred. It may be that the users's PRED is attempting to obtain ETA from a truncated distribution but, due to some error, is rejecting all or virtually all values. (Possibly, the NEW option was omitted from the random source.)

Some messages are preceded by lines such as the following that identify the NONMEM routine that has detected the error:

```
PROGRAM TERMINATED BY OBJ
PROGRAM TERMINATED BY OBJ, ERROR IN CELS
PROGRAM TERMINATED BY OBJ, ERROR IN ELS
PROGRAM TERMINATED BY PRRES, ERROR IN ELS
```

PROGRAM TERMINATED BY FNLETA

OBJ computes the objective function;
ELS computes Extended Least-Squares contribution to the objective function from the data from a single individual ("L1") record.
CELS computes the conditional contribution to the ELS objective function;
PRRES prints final results.
FNLETA computes subject-specific (posthoc) eta values.

WITH INDIVIDUAL 1 (IN INDIVIDUAL RECORD ORDERING)

A line such as this identifies the record.

Here are some commonly seen error messages.

INTRAINDIVIDUAL VARIANCE OF DATA FROM OBS RECORD 1 ESTIMATED TO BE 0
VAR-COV OF DATA FROM INDIVIDUAL RECORD ESTIMATED TO BE SINGULAR

A possible cause is the use of a proportional intra-individual error model while some predicted values for actual observations are zero or close to zero. (For example, if the first dose is an infusion and there is a "base- line" observation at the start of the infusion, the predicted level will be zero.)

MINIMIZATION TERMINATED DUE TO PROXIMITY OF NEXT ITERATION
EST. TO A VALUE AT WHICH THE OBJ. FUNC. IS INFINITE

NONMEM Theta-recovery has failed.

OCCURS DURING SEARCH FOR ETA AT A NONZERO VALUE OF ETA

An error occurred while NONMEM was obtaining conditional estimates of eta.

Section III. Errors in PREDPP

Errors in LSODA

As of NONMEM 7, A differential equation solver has been introduced, called LSODA, and is accessed using ADVAN=13 or ADVAN13. The code to the differential equation solver is found in source\LSODA.f90. On occasion, coded errors will be displayed if the algorithm is having trouble integrating the equations. These errors may usually be ignored, unless the error shows up frequently, and ultimately results in failure for the problem to complete. Typically the remedy is to increase or decrease TOL, but for those who desire to understand what the error codes mean, there are well documented comments on these at the beginning of LSODA.f90. They are printed here for convenience:

```
!ISTATE=An index used for input and output to specify the the state of the calculation.
!
! On input,the values of istate are as follows.
! 1  Means this is the first call for the problem (initializations will be done).
!    See note below.
! 2  Means this is not the first call,and the calculation is to continue
!    normally, with no change in any input parameters except possibly TOUT
!    and ITASK. (If ITOL,RTOL,and/or ATOL are changed between calls with
!    ISTATE=2,the new values will be used but not tested for legality.)
! 3  Means this is not the first call,and the calculation is to continue
!    normally,but with a change in input parameters other than TOUT and ITASK.
!    changes are allowed in NEQ,ITOL,RTOL,ATOL,IOPT,LRW,LIW,JT,ML,MU and any
!    optional inputs except H0,MXORDN,AND MXORDS.
!    (see IWORK description for ML and MU.)
! Note: A preliminary call with TOUT=T is not counted as a first call here,as
! no initialization or checking of input is done. (Such a call is sometimes
! useful for the purpose of outputting the initial conditions.) Thus the first
```

```
! call for which TOUT /= T requires ISTATE=1 on input.
!
! On output,istate has the following values and meanings.
!  1  Means nothing was done; TOUT=T and ISTATE=1 on input.
!  2  Means the integration was performed successfully.
! -1  Means an excessive amount of work (more than MXSTEP steps) was done on
!     this call,before completing the requested task,but the integration was
!     otherwise successful as far as T. (MXSTEP is an optional input and is
!     normally 500.) TO continue,the user may simply reset ISTATE to a value > 1
!     and call again (the excess work step counter will be reset to 0).
!     In addition,the user may increase MXSTEP to avoid this error return
!     (see below on optional inputs).
! -2  Means too much accuracy was requested for the precision of the machine
!     being used. This was detected before completing the requested task,but
!     the integration was successful as far as T. To continue,the tolerance
!     parameters must be reset,and ISTATE must be set to 3. The optional output
!     TOLSF may be used for this purpose. (Note: If this condition is detected
!     before taking any steps,then an illegal input return (ISTATE=-3) occurs
!     instead.)
! -3  Means illegal input was detected,before taking any integration steps.
!     See written message for details.
!     Note: If the solver detects an infinite loop of calls to the solver with
!     illegal input,it will cause the run to stop.
! -4  Means there were repeated error test failures on one attempted step,before
!     completing the requested task,but the integration was successful as far as T.
!     The problem may have a singularity,or the input may be inappropriate.
! -5  Means there were repeated convergence test failures on one attempted step,
!     before completing the requested task,but the integration was successful as
!     far as T. This may be caused by an inaccurate jacobian matrix, if one is
!     being used.
! -6  Means EWT(I) became zero for some I during the integration. Pure relative
!     error control (ATOL(I)=0.0) was requested on a variable which has now
!     vanished. The integration was successful as far as T.
! -7  Means the length of RWORK and/or IWORK was too small to proceed,but the
!     integration was successful as far as T. This happens when DLSODA chooses
!     to switch methods but LRW and/or LIw is too small for the new method.
!
! Note: Since the normal output value of ISTATE is 2, it does not need to be
! reset for normal continuation. Also,since a negative input value of ISTATE
! will be regarded as illegal, a negative output value requires the user to
! change it, and possibly other inputs,before calling the solver again.
```

REFERENCES: Guide I, section G
REFERENCES: Guide V, section 13

NONMEM FILE CONTROL STREAM

MEANING: NONMEM File Control Records
CONTEXT: NONMEM input file

DISCUSSION:
The NONMEM File Control Stream provides the names of optional files to be opened by NONMEM. It is read by NONMEM from a file whose name is given on the FILE record of the NONMEM Control Stream.

When NM-TRAN is used, NM-TRAN creates the NONMEM File Control Stream in a file named FSTREAM.

Each NONMEM File Control Record begins with a 4-character preface (record label) which identifies the record.

Record type	Prefix	Corresponding NM-TRAN record
DATA	DATA	$DATA ...
MODEL SPECIFICATION FILE INPUT	MSFI	$MSFI ...
MODEL SPECIFICATION FILE OUTPUT	MSFO	$ESTIMATION MSFO=...
TABLE	TABL	$TABLE FILE=...
Problem delimiter	****	none

Multiple TABLE records may be present for a given problem, one for each table to be written to a new table file.

REFERENCES: Guide I, section B.3

NONMEM MODEL

MEANING: The name of a type of statistical model.
CONTEXT: NONMEM input/output

DISCUSSION:

The acronym NONMEM denotes both the model and the program used to analyze data according to such a model. (See **nonmem_program**) This entry discusses the NONLINEAR MIXED EFFECTS MODEL (NONMEM).

Regression models structurally link (possibly multivariate) observations (Dependent variables, DV) to independent variables (fixed effects, represented by other data items) through a functional form (model) quantified by (fixed effect) parameters. These model forms may be nonlinear in the parameters.

Random effects may also enter the model. In NONMEM they are of two types which usually enter the model at two different levels. The first type, ETA, describes differences between individuals; the second type, EPSILON, describes errors between model predictions and observations. (When all data come from the same individual, or when each observation is to be treated as statistically independent from all others, then ETA-type random effects describe all errors, and EPSILON-type random effects do not appear).

Within an individual, ETA may be a vector. Likewise, within one observation, EPSILON may be a vector, especially if the observations are multivariate, as several random effects of each type may be needed to characterize the data adequately.

The parameter vector THETA contains all fixed effect population parameters.

The OMEGA matrix, a random effects parameter, is the variance-covariance matrix of ETA (across individuals); the SIGMA matrix serves the same function for EPSILON (its variance covariance is assumed identical across all observations).

(See **eta**, **eps**, **theta**, **effect**). (See **parameter**, **model**, **omega**, **sigma**).

REFERENCES: Guide I, section C , D , E
REFERENCES: Guide V, section 3 , 4

NONMEM MODULES

MEANING: Global variables in NONMEM
CONTEXT: User-supplied routines

DISCUSSION:
FORTRAN modules are used to communicate values between various components of the NONMEM system. They supplement the subroutine arguments.

1) NONMEM-PRED modules

These modules contain values that are (for the most part) communicated from PRED to NONMEM. Prior to NONMEM 7 these values were COMMON blocks. Following is a list of the MODULE, the (old) COMMON, and a description of the variables.

NMPRD_INT (NMPRD1)
: PRED return code and user message count (`IERPRD,NETEXT`)

NMPRD_CHAR (NMPRD2)
: PRED user message (`ETEXT`)

NMPRD_INT (NMPRD3)
: "Copying pass" flag and SAVE region size for NMPRD4 (`COMACT,COMSAV`)

NMPRD4P (NMPRD4)
: PRED-defined items for tables and scatterplots

NMPRD_REAL (NMPRD5)
: Correlation matrix for ε's.

NMPRD_INT (NMPRD6)
: Return code from routine SIMEPS.

NMPRD_REAL (NMPRD7)
: Simulated etas for tables and scatterplots.

NMPRD_REAL (NMPRD8)
: PRED's "recursive" flag

NMPRD_REAL (NMPRD9)
: Data record at ICALL 0, 1 and 3

NMPR_INT (NMPR10)
: Control information for PRED repetition feature.

NMPR_INT (NMPR11)
: Flags to override effect of NEW option on $SIMULATION record.

NMPR_REAL (NMPR12)
: Conditional limits on observation.

NMPR_REAL (NMPR13)
: Lower limits defining interval datum, and their derivatives.

NMPR_REAL (NMPR14)
: Upper limits defining interval datum, and their derivatives.

NMPR_INT (NMPR15)
: Skip control variable.

NMPR_REAL (NMPR16)
: Parameter values produced during Simulation Step.

NMPR_INT (NMPR17)
Flag indicating character of PRED variable F.

2) NONMEM read-only modules

These modules contain read-only values that are communicated from NONMEM to PRED and other user-supplied routines.

ROCM_REAL (ROCM0)
Current theta (`THETA`)

ROCM_REAL (ROCM1)
DV, data items in current L1 record

ROCM_INT (ROCM1C)
Size of current L1 record (prior to NONMEN 7, this was combined with ROCM1).

ROCM_INT (ROCM2)
Number of L2 records and length of L2 record

ROCM_REAL (ROCM3)
Predictions and derivatives with current L1 record

ROCM_REAL (ROCM4)
Selected data from an individual record

ROCM_INT (ROCM4)
Selected data from an individual record

ROCM_REAL (ROCM5)
Predictions and derivatives with current L2 record

ROCM_REAL (ROCM6)
Initial/final theta, omega, sigma

ROCM_REAL (ROCM7)
Standard errors of estimates of theta, omega, sigma

ROCM_REAL (ROCM8)
Final value of the objective function

ROCM_INT (ROCM9)
Return codes from Estimation and Covariance Steps

ROCM_INT (ROCM10)
Simulation: no. of replications (total and current)

ROCM_INT (ROCM11)
Mixture: index of subpopulation (current and most probable)

ROCM_INT (ROCM12)
Second and first derivative flags for PRED

ROCM_INT (ROCM13)
Simulation: Final pass flag for PRED and CONTR

NMPRD_INT (ROCM14)
Problem and subproblem counters.

NMPRD_INT (ROCM15)
List of inactive etas.

ROCM_REAL (ROCM16)
The number of significant digits in the the vector of final estimates.

ROCM_INT (ROCM17)
New level 2 record flag

ROCM_REAL (ROCM18)
Nonparametric estimates: height and heights of cumulative marginals

ROCM_REAL (ROCM22)
Current value of omega.

ROCM_REAL (ROCM25)
Mixing probabilities

ROCM_REAL (ROCM28)
Superproblem printing indicator

NMPRD_INT (ROCM29)
Population vs. single-subject data flag.

ROCM_REAL (ROCM31)
Template record at ICALL=6.

ROCM_INT (ROCM32)
Number of the individual record and of the data record.

ROCM_INT (ROCM34)
NEWIND at ICALL 0, 1 and 3.

ROCM_INT (ROCM35)
Numbers of thetas, etas and epsilons in the problem.

ROCM_REAL (ROCM36)
Individual's posterior variance-covariance matrix. No help entry.

ROCM_INT (ROCM37)
Indicator that Estimation Step is omitted and obj. func. is being evaluated for first time. No help entry.

ROCM_REAL (ROCM38)
Conditional limits for observations in individual record. No help entry.

ROCM_REAL (ROCM39)
Conditional limits for observation. No help entry.

ROCM_REAL (ROCM40)
Lower Limits defining interval datum, and their derivatives in individual record. No help entry.

ROCM_REAL (ROCM41)
Upper Limits defining interval datum, and their derivatives in individual record. No help entry.

ROCM_REAL (ROCM42)
Lower Limits defining interval datum, and their derivatives. No help entry.

ROCM_REAL (ROCM43)
Upper Limits defining interval datum, and their derivatives. No help entry.

ROCM_REAL (ROCM44)
Probability that observation is within (outside) limits.

ROCM_REAL (ROCM45)
Probability that category occurs.

ROCM_INT (ROCM46)
Number of individual records with observations, and indices of first and last such

record.

ROCM_REAL (ROCM47)
Mixture probabilities with individual record containing template record.

ROCM_INT (ROCM48)
Length of individual record.

ROCM_REAL (ROCM49)
Prediction, residual, and weighted residual values for all methods, as well as normalized probability distribution error.

ROCM_INT (ROCM50)
ID data item for the individual contributions to the objective function.

ROCM_REAL (ROCM50)
Individual contributions to the objective function.

Except as noted, each has its own entry in the Help document. Some modules are not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of NONMEM code and previous examples that may be available from advanced users.

(See **PREDPP_modules**).

REFERENCES: Guide IV, section IV.E
REFERENCES: Guide VI, section III.E.4 , III.I , Figures 5, 6

NONMEM OUTPUT FILES

MEANING: NONMEM output files
CONTEXT: NONMEM output file

DISCUSSION:

NONMEM produces several output files.

First is the NONMEM report file. The name of this file is specified by the user on the nmfe command. Help entries describe the various sections of the report.

(See **DATA_SET**,**DATA_RECORDS**,**DATA_ITEMS**).
(See **INITIAL_ESTIMATE**,**MONITORING_OF_SEARCH**).
(See **MINIMUM_VALUE_OF_OBJ._FUNCTION**,**FINAL_PARAMETER_ESTIMATE**).
(See **ETABAR**,**STANDARD_ERROR_OF_ESTIMATE**).
(See **SHRINKAGE**).
(See **CORRELATION_MATRIX_OF_ESTIMATE**,**COVARIANCE_MATRIX_OF_ESTIMATE**).

With NONMEM 7, tag labels are present in the report file to allow a third-party program to find the appropriate positions in the file without having to search for specific words in the text. (a more efficient way of extracting numerical results from the analysis is from the raw and additional output files (see below)). Tag labels are:

#PARA (NM72)
: This tag identifies the parallelization file and number of nodes used, if parallel estimation is performed.

#TBLN (NM72)
: This tag specifies that following it, on the same line, will be found an integer that refers to the number of this estimation method. This number is also the table number listed in the title to tables in the various output files (raw output file, .cov, .cor, etc). The table number is incremented for each $EST statement, across all problems in the control stream file.

#METH
: This tag specifies that following it, on the same line, will be found a text that describes the method, for example First Order Conditional Estimation Method with Interaction.

#TERM
: This tag indicates that beginning on the next lines, text describes the termination status of the analysis.

#TERE
: This tag indicates the end of the lines describing the termination status of the analysis. Thus, a software program may transfer all lines between #TERM: and #TERE: to a summary file.

#OBJT
: Indicates that following it, on the same line, is the text describing the objective function, such as Minimal Value Of Objective Function.

#OBJV
: Indicates that following it, on the same line, is the objective function value.

#OBJS
Indicates that following it, on the same line, is the objective function standard deviation (MCMC Bayesian analysis only).

#OBJN (NM73)
Indicates that following it, on the same line, is the nonparametric objective function value.

#CPUT (NM73)
Total CPU time. It is an accurate representation of the computer usage, whether single or parallel process. The same problem when run singly or in parallel will report a similar cpu time. This is in contrast with elapsed time, which is improved with parallelization.

With NONMEM 7, raw and additional output files are generated to capture parts of the report file.

The raw output file is called root.ext, or named by the FILE option of the $ESTIMATION record.
(See **raw_output_file**).

The additional output files capture portions of the report file.
(See **additional_output_file**).
They are listed in the same order as in **additional_output_fil**.

root.phi
root.iph (NM75)
root.phm (NM72)
root.shk (NM72)
root.shm (NM73)
root.grd (NM72)
root.xml (NM72)
root.cov
root.clt (NM74)
root.cor
root.coi
root.cnv (NM72)
root.smt (NM72)
root.rmt (NM72)
root.imp (NM73)
root.npd (NM73)
root.npe (NM73)
root.npi (NM73)
root.npl (NM74)
root.fgh (NM73)
root.agh (NM73)
root.vpd (NM74)
root.ets (NM74)
root.bfm (NM75)
root.cpu (NM73)
msfroot_ETAS
msfroot_RMAT, msfroot_SMAT (NM73)

REFERENCES: Guide Introduction_7

NONMEM PROGRAM

MEANING: The name of a computer program.
CONTEXT: NONMEM input/output

DISCUSSION:
The acronym NONMEM denotes both a type of statistical model and the program used to analyze data according to such a model. (See **nonmem_model**). This entry briefly describes the NONLINEAR MIXED EFFECTS MODEL (NONMEM) program.

NONMEM is a computer program written in FORTRAN 90 used to analyze data according to a Non-linear Mixed Effects Model. NONMEM is batch-oriented. It is accompanied by an input preprocessor (NM-TRAN) that provides a SAS-like user communication interface, and by a suite of subprograms (PREDPP) that, within the context of the NONMEM program, implements most of the common simple pharmacokinetic models used by pharmacokineticists. PREDPP also has code for solving general linear and non-linear differential equations describing more complicated pharmacokinetic models.

The user generally will have to supply data and a code (either in an abbreviated form understood by NM-TRAN, or in a FORTRAN subroutine) specifying the dependence on the mixed effects. He will also have to make choices regarding the NONMEM process: e.g., whether to estimate parameters, and if so, which of several estimation methods to use (each method minimizes a different objective function).

NONMEM output reports the minimum value of the objective function, parameter estimates, standard errors, tables, and line-printer scatterplots, etc., as requested. NONMEM computes model predictions using the parameter estimates, and similarly, residuals and weighted residuals, and makes these available for tables and scatterplots.

NONMEM can also easily be used to simulate data under a NONMEM model.

With NONMEM 7, there are additional output files, called the "raw" and "additional" output files. Here, "root" is the name of the NM-TRAN input file specified on the nmfe7 command (not including extension). These provide an efficient way of way of extracting numerical results from the analysis. By default, the raw output file is root.ext. The additional output files include root.ext root.cov root.coi root.cor root.phi, etc., where root is the name of the control file (not including extensions).

(See **raw_output_files**).
(See **additional_output_files**).
(See **estimate**, **method**, **parameter**, **parameter_estimate**).
(See **objective_function**, **table**, **scatterplot**).
(See **residual**, **weighted-residual**, **predpp**).

REFERENCES: Guide I, section A
REFERENCES: Guide V, section 1 , 2

NONMEM STEPS

MEANING: Sequence of events in a NONMEM problem
CONTEXT: NONMEM output

DISCUSSION:

NONMEM loads information from a model specification file, if $MSFI is present NONMEM makes a single call to PRED (with ICALL=0), at the beginning of the run, to allow PRED to undertake run initialization (INITL).

NONMEM makes a single call to PRED (with ICALL=1), at the beginning of every problem, to allow PRED to undertake problem initialization (INITL).

SIMULATION STEP

NONMEM inputs initial fixed effects parameters from file specified by $CHAIN, if record is present NONMEM performs simulation (with ICALL=4) (if $SIMULATION present).

INITIAL ESTIMATE STEP (with ICALL=2)

NONMEM obtains initial estimates for theta, omega, sigma (if the control stream does not provide all initial estimates).

ESTIMATION STEP (with ICALL=2)

NONMEM inputs initial etas from file specified by $ETAS/$PHIS, if record is present NONMEM searches for parameter values that minimize the objective function (if $ESTIMATION present). With NONMEM 7, multiple Estimation Steps can be implemented in a single problem. A sequence of two or more Estimation Steps will result in the sequential execution of each.

COVARIANCE STEP (with ICALL=2)

NONMEM obtains a variance-covariance matrix of the estimator and related information such as standard errors (if $COVARIANCE present).

FINALIZATION STEP (with ICALL=2,5,6)

NONMEM obtains the following as needed:

Classification into subpopulations (if a MIXTURE model is used)(FNLMOD).

Posthoc estimates of eta (FNLETA).

Nonparametric step (NONPAR, with ICALL=2): NONMEM obtains either marginal cumulatives or conditional (nonparametric) estimates of etas (if $NONPARAMETRIC present).

PRED-defined values from NMPRD4 (NP4F) (for tables and scatterplots).

NONMEM computes WRES and other diagnostics if they are to be displayed in tables or scatters. (PRRES)

NONMEM makes a single call to PRED (with ICALL=3), at the end of every problem (and subproblem), to allow PRED to undertake problem finalization.

TABLE STEP

NONMEM constructs tables (if $TABLE present).

SCATTERPLOT STEP

NONMEM constructs scatterplots (if $SCATTERPLOT present).

REFERENCES: Guide I, section A.4 , G

NONPARAMETRIC DENSITY: DEN ,CDEN.ta 5.5iR

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

USAGE:

```
USE ROCM_REAL, ONLY: DEN_=>DEN_NP(1),CDEN_=>DEN_NP(2:LVR+1)
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: LVR,DPSIZE
REAL(KIND=DPSIZE) :: DEN_NP(LVR+1)
```

DISCUSSION:

`DEN_`
: The nonparametric density.

`CDEN_(1)`
: The marginal cumulative value for the first eta.

`CDEN_(2)`
: The marginal cumulative value for the second eta.
etc.

Values are computed by NONMEM when the Nonparametric step is performed and marginal cumulatives are requested (with NM-TRAN: $NONPARAMETRIC MARGINALS).

Values are available during the pass with COMACT=2 and (if PASS is called) at ICALL=3.

These variables may be used as right-hand quantities in abbreviated code blocks $PK, $ERROR, $INFN and $PRED.

Location prior to NONMEM 7: rocm18

REFERENCES: None.

NPD,NPDE,NPDE MODE,DV LOQ,CDF L,DV LAQ CDF LA

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPRD_INT, ONLY: NPDE_MODE
USE NMPRD_REAL, ONLY: DV_LOQ
```

GLOBAL DECLARATION:

```
       INTEGER(KIND=ISIZE) ::  NPDE_MODE
       REAL(KIND=DPSIZE) :: DV_LOQ
```

DISCUSSION:

NONMEM 7 can compute the following:

`NPDE`
: Monte-Carlo generated normalized probability distribution error.

`NPD`
: The correlated (or non-decorrelated) NPDE value.

These are calculated during the Table Step, when NPD or NPDE is listed in $TABLE or $SCATTER.

NONMEM sets NPDE_MODE=1 when it is computing them. Otherwise, NPDE_MODE=0.

EXAMPLE:

To incorporate LOQ (Limit of Quantification) data into NPDE assessments [4], use the following method (as an example).
See **INTRODUCTION TO NONMEM 7, Reference [4].**
Here, TYPE and LOQ are user-defined in previous code, or data item.

```
$ERROR
SD = THETA(5)
IPRED = LOG(F)
DUM = (LOQ - IPRED) / SD
CUMD = PHI(DUM)
IF (TYPE .EQ. 1.OR.NPDE_MODE.EQ.1) THEN
      F_FLAG = 0
      Y = IPRED + SD * ERR(1)
ENDIF
IF (TYPE .EQ. 2.AND.NPDE_MODE.EQ.0) THEN
      F_FLAG = 1
      Y = CUMD
      MDVRES=1
ENDIF
IF(TYPE.EQ.2) DV_LOQ=LOQ
  ....
$TABLE NPD NPDE
```

By default, DV_LOQ is set to -1.0d-300 by the NONMEM routine that calls PREDPP/PRED. If the user's ERROR/PRED sets DV_LOQ to some other value and NPDE_MODE=1, then the NPDE is being evaluated during that time, and this censored value is to be treated as if it is a non-censored datum with value of LOQ

(DV_LOQ=LOQ), in accordance with [4], utilizing a standard F_FLAG=0 definition for Y. Note that during estimation of the objective function (when NPDE_MODE=0), NPDE is not being evaluated, and censored values should be treated using F_FLAG=1, and Y must be defined as the integral of the normal density from -inf to LOQ.

New in nm74, for use with NPD, the user may supply the cumulative distribution function using the reserved variable CDF_L.

New in nm743, you can specify an above quantifiable limit with the reserved parameter DV_LAQ as well. The CDF reserved variable associated with above quantitation level DV_LAQ is CDF_LA.

The DV_LOQ/CDF_L and DV_LAQ/CDF_LA reserved variables may be also used for evaluating NPD diagnostics for categorical data.

See **INTRODUCTION TO NONMEM 7, MDVRES=0**

REFERENCES: Guide Introduction_7

NWPRI

MEANING: NWPRI subroutine
CONTEXT: NONMEM utility routine

USAGE:

```
USE SIZES, ONLY: ISIZE,DPSIZE
INTEGER (KIND=ISIZE) :: NTHETA,NETA,NEPS,NTHP,NETP,NEPP,NPEXP
INTEGER (KIND=ISIZE) :: ITYP,NSAM,ISS
REAL (KIND=DPSIZE), ONLY: CNT
REAL (KIND=DPSIZE), ONLY: PLEV
INTEGER (KIND=ISIZE) :: ICALL
CALL NWPRI (NTHETA,NETA,NEPS,NTHP,NETP,NEPP,NPEXP,ITYP,PLEV, &
            NSAM,ISS,CNT)
```

DISCUSSION:
The user-written PRIOR subroutine allows a penalty function based on a frequency prior to be specified and added to the -2log likelihood function (Gisleskog et al, JPP, 2002, p. 473-505). This serves as a constraint on THETA, OMEGA, and SIGMA estimates and thus as a way for stable estimates of these parameters to be obtained with insufficient data. NWPRI may be called by PRIOR. (See **prior**). It computes a function based on a frequency prior that has a multivariate normal form for THETA, and also, in the case of population data, an inverse Wishart form for OMEGA (independent from the normal for THETA). The parameters of these forms are called "hyperparameters", and the modeler fixes these to values of his/her choice.

Using NWPRI, several penalties functions, each with a different set of values for the hyperparameters, may be used simultaneously. Each set of values may be thought to arise from a different prior experiment (study). In actuality, there is a way to combine the several penalties functions into one (see above reference), and this is what NWPRI computes.

When NWPRI is used during a Simulation Step, it produces a random value of THETA and a random value of OMEGA from the frequency prior. (See **Simulation example**).
(See **nwpri example**).
If NWPRI is used at ICALL=2, it need not be used at ICALL=4, and vice-versa.

NWPRI should always be called at ICALL=0 or ICALL=1.

Use only NWPRI for the new $ESTIMATION methods of NONMEM 7.

Arguments

Input arguments:

`NTHETA,NETA,NEPS`

NTHETA=number of Thetas to be estimated
NETA=number of Etas to be estimated
NEPS=number of epsilons to be estimated
These are The dimensions of the THETA, OMEGA, and SIGMA arrays of the parameters that enter into the model for the data.

Before NONMEM 7.3, NEPS was ignored. With odd-type population data or with non-odd-type single-subject data, where OMEGA takes the place of SIGMA, the input argument NETA must be set.

NTHP,NETP,NEPP

NTHP=number of thetas which have a prior
NETP=number of Omegas with prior
NEPP=Number of Sigmas with prior

The prior will only affect the initial subvector of THETA of dimension NTHP and the initial submatrix of OMEGA of dimension NETP (i.e. the submatrix consisting of the intersection of the first NETP rows and the first NETP columns of OMEGA). The initial subvector and submatrix are called the prior-affected parts of THETA and OMEGA. If NTHP=0 (NETP=0), the prior does not affect THETA (OMEGA) at all. During the Simulation Step, simulated values for the affected parts of THETA and OMEGA are obtained according to the prior distribution, and "the simulated values" for the unaffected parts of THETA and OMEGA are simply taken to be the values given in the NONMEM control stream or input Model Specification record. Before NONMEM 7.3, NEPP was ignored.

NPEXP

The number of prior experiments.

ITYP

Relevant only if NWPRI is called during a Simulation Step. Values are

0: The value of the THETA vector is obtained from simple random sampling.

1: Within the given problem, NWPRI is to be called a specified number (NSAM) of times to obtain this number of different values of the THETA vector. These values are obtained by generating a Latin sample of size NSAM from equiprobable partitions of an ellipsoid in THETA space (hyper-ellipsoidal sampling), followed by sampling a point "uniformly" from each partition. This scheme may be used, for example, when the problem has NSAM subproblems, in which case, NWPRI would be called NSAM times, once each time during the problem when ICALL=4, and at each of these calls, a different random value of THETA will be produced.

2: Just as with value 1, but the NSAM values are obtained by generating a Latin sample of size NSAM from equiprobable partitions of an ellipsoid in THETA space (hyper-ellipsoidal sampling), followed by taking the "center point" of each partition.

In all three cases, each new value of the OMEGA matrix is obtained from simple random sampling.

After each call to NWPRI, the simulated values for THETA, OMEGA and SIGMA may be found in global variables and thus they are communicated directly to NONMEM. (See **PRIOR_Simulation:_Parameters**).

PLEV

When NWPRI is being used at ICALL=0 or 1, but NWPRI will not be used at ICALL=4 (i.e. during the Simulation Step), PLEV can be set to 0. When it is being used at ICALL=2, PLEV can also be set to 0. When NWPRI is being used at ICALL=0 or 1 and it will be used at ICALL=4, or when it is being used at ICALL=4, then PLEV must be set to a fraction strictly less than 1, e.g. 0.999. PLEV is double precision with NONMEM 7, and is single precision with NONMEM VI.

A value of THETA will actually be obtained using a truncated multivariate normal distribution, i.e. from an ellipsoidal region R1 over which only a fraction of mass of

the normal occurs. This fraction is given by PLEV. The distribution is further truncated to R2, the subregion of R1 lying within the rectangular boundaries defined on the `$THETA` record. Simple random sampling occurs in R2. Latin sampled partitions are partitions of R1. However, when ITYP=1, if a uniformly sampled point from a partition lies outside R2, it is replaced by a point obtained by simple random sampling from R2. When ITYP=2, if the center point of a partition lies outside R2, an abort occurs.

`NSAM`

Relevant only if NWPRI is called during a Simulation Step. Consider two cases. a) Latin hyper-ellipsoid sampling is used with ITYP=1, and b) simple random sampling along with the adjustment for small sample correlation effect is used (see next input argument). In case a) NSAM should equal the exact total number of different values of THETA that must eventually be produced over the entire NONMEM problem. In case b) NSAM should be no less than this number.

`ISS`

Relevant only if NWPRI is called during a Simulation Step. A THETA value is obtained by transforming a value from the standard multivariate normal distribution - called here "the standard value". The correlation matrix of the standard normal is the identity matrix. When NSAM is small, the estimated correlation matrix from the sampled standard values might not be quite close to the identity matrix - this is here called "the small sample correlation effect".

1: An adjustment is made for the small sample correlation effect, by first transforming the NSAM standard values *altogether* into new values which are very nearly standard multivariate normal values and such that the sample correlation matrix of these new values is exactly the identity matrix.

0: No adjustment is made for the small sample correlation effect.

Output argument:

`CNT`

Relevant only if NWPRI is called at ICALL=2. CNT is the penalty.

Use of $PRIOR

A $PRIOR record may be used instead of a user-written PRIOR subroutine, in which case the input arguments listed above may be specified as options of $PRIOR, and the value of CNT may be displayed in the NONMEM output. (See **$prior**).

Use of $THETA, $OMEGA, $SIGMA records

Specifying the multivariate normal form for THETA (NTHP>0):

After the usual set of $THETA records, add a second set of $THETA records giving the mean of the multivariate normal. The values on these records must be fixed. They should number NTHP altogether.

After the usual set of $OMEGA records, add a second set of $OMEGA records giving the variance-covariance matrix of the multivariate normal. (If there are no etas in the model, there will not be a usual set of $OMEGA records.) The values on these records must be fixed. They should number (NTHP+1)xNTHP/2 altogether, including implicit 0's, which may occur because the variance-covariance matrix of the normal may be block-diagonal. If a (regular) theta is fixed, the corresponding values of the mean and variance-covariance matrix of the normal are ignored.

Specifying the inverse Wishart form for OMEGA (NETAP>0; population data only):

After the second set of $OMEGA records (or if NTHP=0, after the first set of $OMEGA records), add another set of $OMEGA records giving the mode of the inverse Wishart. The values on these records must be fixed. They should number (NETP+1)xNETP/2 altogether, including implicit 0's, which may occur because the mode of the inverse Wishart may be block-diagonal. If the prior-affected part of OMEGA is given as a block-diagonal matrix, then the mode must conform to this structure. The SAME attribute can be used.

With each diagonal block of (the prior-affected part of) OMEGA, there corresponds a number of "degrees of freedom" of the inverse Wishart. All blocks of a given block set are constrained to be equal (by using the SAME attribute), and therefore, to each of these blocks there corresponds the same number of degrees of freedom. After the second set of $THETA records (or if NTHP=0, after the first set of $THETA records), add another set of $THETA records, giving for each block set in turn the number of degrees of freedom for the blocks of the set. The values on these records must be fixed. There should be as many values as there are block sets with the prior-affected part of OMEGA.

The inverse Wishart for a given block of OMEGA may be explicitly given as "perfectly flat" by specifying the number of degrees of freedom to be the negative of the dimension of the block, minus 1. In this case the mode for the block may be simply taken to be the 0 matrix (or any positive definite matrix). If a block is fixed, the corresponding values of the mode and number of degrees of freedom are ignored.

With NONMEM 7.3 and higher, you can use more informative record names that obviate the need for specific ordering of the additional $THETA and $OMEGA records used by NWPRI, and provide an alternative source for values of arguments NTHETA, NETA, et al.

$THETAP for theta priors
$THETAPV for variance-covariance matrix for theta's
$OMEGAP for OMEGA prior
$OMEGAPD for degrees of freedom (or dispersion factor) for omega prior
$SIGMAP for SIGMA prior
$SIGMAPD for degrees of freedom (or dispersion factor) for sigma prior

If the informative record names are used, the records may be in any order, and the options of $PRIOR need not be specified. Note that the name of the record describes the kind of information it gives, rather than the structure of the information. E.g., in the example below, $THETAPV is used instead of an $OMEGA record and $OMEGAPV is used instead of a $THETA record.
(See **$thetap**, **$omegap**, **$sigmap**)

Examples

Here are three examples of extra $THETA and $OMEGA records specifying prior information from an experiment. This information concerns all the regular elements of THETA and OMEGA. All examples are shown with a $PRIOR record although a PRIOR subroutine could be used instead.

```
$PRIOR NWPRI NTHETA=3 NETA=3 NTHP=3 NETP=3 NPEXP=1
  ...
$THETA 3 FIX .08 FIX .04 FIX
$THETA 100 FIX
$OMEGA BLOCK (3) .494 .00207 .0000847 .000692 .0000471 .0000292 FIX
```

```
$OMEGA BLOCK (3) .7 .04 .05 .02 .06 .08 FIX
```

Perhaps it might be more perspicuous to organize the prior information thusly:

```
$PRIOR NWPRI NTHETA=3 NETA=3 NTHP=3 NETP=3 NPEXP=1
  ...
;prior information for THETA
$THETA 3 FIX .08 FIX .04 FIX
$OMEGA BLOCK (3) .494 .00207 .0000847 .000692 .0000471 .0000292 FIX

;prior information for OMEGA
$OMEGA BLOCK (3) .7 .04 .05 .02 .06 .08 FIX
$THETA 100 FIX
```

With NONMEM 7.3 and higher, informative record names can be used:

```
$PRIOR NWPRI
  ...
;prior information for THETA
$THETAP 3 FIX .08 FIX .04 FIX
$THETAPV BLOCK (3) .494 .00207 .0000847 .000692 .0000471 .0000292 FIX

;prior information for OMEGA
$OMEGAP BLOCK (3) .7 .04 .05 .02 .06 .08 FIX
$OMEGAPD 100 FIX
```

Multiple Experiments

There may be prior information from a number of experiments, in which case the $THETA and $OMEGA records specifying this information for each experiment may be stacked, e.g.

```
;usual records
$THETA  (0,4,10) (0,.09,.5) (.004,.01,.9)
$OMEGA BLOCK (3) .7 .04 .05 .02 .06 .08

;prior information from experiment 1
$THETA 3 FIX .08 FIX .04 FIX
$THETA 100 FIX
$OMEGA BLOCK (3) .494 .00207 .0000847 .000692 .0000471 .0000292 FIX
$OMEGA BLOCK (3) .7 .04 .05 .02 .06 .08 FIX

;prior information from experiment 2
$THETA 2 FIX .05 FIX .04 FIX
$THETA 50 FIX
$OMEGA BLOCK (3) .6 .003 .0001 .0004 .00001 .00003 FIX
$OMEGA BLOCK (3) .9 .02 .05 .01 .06 .09 FIX
```

Blocking on the variance-covariance matrix of the normal form for THETA need not be the same across experiments.

Limitation:

There must be at least one THETA parameter and one OMEGA parameter in the model.

REFERENCES: None

OBJ

MEANING: OBJ subroutine
CONTEXT: NONMEM routine

DISCUSSION:
OBJ is a NONMEM routine. It computes the objective function. Error messages from OBJ may indicate some difficulty in this computation.

REFERENCES: None.

OBJECTIVE FUNCTION VALUE INDIVIDUAL

MEANING: NONMEM read-only global variables
CONTEXT: PRED routine

USAGE:

```
USE ROCM_INT, ONLY: IIDX=>IDVALX
USE ROCM_REAL, ONLY: CNTID=>OFV_IND
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: MAXIDS,DPSIZE
INTEGER(KIND=ISIZE) :: IDVALX(MAXIDS)
REAL(KIND=DPSIZE) :: OFV_IND(MAXIDS)
```

DISCUSSION:

Note: With NONMEM 7, the additional output file root.phi contains the same information.
(See **additional_output_file**).

These variables contain values of the ID data item and individual contributions to the objective function. The values are in data-set order.

`IIDX`
: Values of the ID data item.

`CNTID`
: Values of the individual contribution to the objective function for the corresponding values of IIDX.

E.g., IIDX(n) is the ID data item for the nth. individual record, and CNTID(n) is the contribution to the objective function for the nth. individual record.

These values should only be displayed at ICALL = 3 (finalization block).

With NONMEM VI 1.0, they can only be displayed using verbatim code.
With NONMEM VI 2.0 and later releases, they can be used on the right and displayed using abbreviated code in $PRED, $PK, $ERROR and $INFN blocks (See **Individual objective function example**).

(See **write**).

They may also be displayed in a table, using

```
$ABBR COMRES=2
```

and code such as the following in the $ERROR or $PK block:

```
IF (COMACT.EQ.1) THEN
 COM(1)=IIDX(NIREC)
 COM(2)=CNTID(NIREC)
ENDIF
```

The following, for example, will produced a separate table for the values:

```
$TABLE IID=COM(1) CNT=COM(2) FILE=comvals NOAPPEND NOPRINT FIRSTONLY
```

Note: With earlier versions than NONMEM 7.3, verbatim code is needed:

```
"  COM(1)=IIDX(NIREC)
"  COM(2)=CNTID(NIREC)
```

Location prior to NONMEM 7: rocm50

REFERENCES: None.

OBJECTIVE FUNCTION VALUE

MEANING: NONMEM read-only global variables
CONTEXT: PRED routine

USAGE:

```
USE ROCM_REAL, ONLY: OBJECT
```

GLOBAL DECLARATION

```
USE SIZES, ONLY: DPSIZE
REAL(KIND=DPSIZE) :: OBJECT
```

DISCUSSION:

`OBJECT`
: The final value of the objective function.

This value should only be used at ICALL = 3 (finalization block).

May be used as a right-hand quantity in abbreviated code.

Location prior to NONMEM 7: rocm8

REFERENCES: None.

OBSERVATION RECORDS

MEANING: Those data records containing observation information
CONTEXT: NONMEM input/output

DISCUSSION:
A data record may contain an observation, in which case the missing dependent variable (MDV) data item (which may be specified by the user, or created for him by NM-TRAN when PREDPP is used) should be zero, and the dependent variable (DV) data item should be the observation. If a data record does not contain an observation, the MDV data item should be 1 or 101, and the DV data item may be any value. NONMEM counts observation records. It also must know in what position (column) of a data record the DV data item appears. It reports its information about these things as a check to the user, as in this example:

```
 DEP VARIABLE IS DATA ITEM NO.:    4
     .
     .
 TOT. NO. OF OBS RECS:      394
```

(See **dv**, **mdv**).

REFERENCES: Guide III, section V.3.2
REFERENCES: Guide IV, section II.C.4.1
REFERENCES: Guide V, section 6
REFERENCES: Guide VI, section II

ORDER OPTION

MEANING: Instructions for the NONMEM Estimation Step
CONTEXT: NM-TRAN Control Record

`ORDER` is an option of the $ESTIMATION record:

`ORDER=xxxf`

The values of x may be T (Theta), S (Sigma), and O (Order). The value of f may be U (Upper) or L (Lower). Affects the way theta, omega, and sigma are displayed in the raw and additional output files. xxx gives the overall order, and f gives the order within OMEGA and SIGMA. Affects the raw output file and all additional output files. The default is TSOL: THETA, SIGMA, OMEGA in Lower triangular form. Does not affect the NONMEM report file.

DISCUSION:

On the $OMEGA record, elements of Omega are given in lower triangular order, e.g.,

```
$OMEGA BLOCK(3) OM11 OM21 OM22 OM31 O32 OM33
```

This can also be coded as

```
$OMEGA BLOCK(3)
OM11
OM21 OM22
OM31 O32 OM33
```

The NONMEM report file is not consistent. The initial and final parameter estimates are printed in Lower triangular order, e.g.,

```
INITIAL ESTIMATE OF OMEGA:
OM11
OM21 OM22
OM31 OM32 OM33
```

However, in the intermediate printout of the Estimation step, OMEGA and SIGMA are in Upper triangular order. This was not evident with previous version of NONMEM, because the PARAMTER values corresponding to OMEGA and SIGMA are displayed as unconstrained parameters (UCP), and these are not in a 1-1 mapping with the elements of OMEGA and SIGMA. But with NONMEM 7.2 and higher, the intermediate output of the Estimation step includes a line NPARAMETR, in which OMEGA and SIGMA are converted to their "natural" space. E.g.,

```
OM11 OM12 OM13 OM22 OM23 OM33
```

Because OMEGA is symmetric, this is the same as

```
OM11 OM21 OM31 OM22 OM32 OM33
```

In the NONMEM report, upper triangular form is used for all of the "COVARIANCE MATRIX OF THE ESTIMATE", "CORRELATION MATRIX OF ESTIMATE" and the "INVERSE COVARIANCE MATRIX OF ESTIMATE". E.g.,

```
OM11 OM12 OM13 OM22 OM23 OM33
```

This is TOSU order.

Case 1. ORDER is not used (Default)

The raw output file .ext and additional output files .cov, etc, are in a different order: TSOL. SIGMA precedes OMEGA. The order within OMEGA is lower triangular, consistent with the Initial and final estimates of OMEGA, but different than the intermediate output in the NONMEM report. The order of the last line of the

"COVARIANCE MATRIX OF ESTIMATE" in the NONMEM report differs from the order of the lines of the .cov file, and similarly for .cor and .coi.

Case 2. `ORDER=TOSU`

Each line of the raw output file .ext is in the same order as in the intermediate output portion of the NONMEM report. Also, each line of the .cov file is in the same order as the last line of the "COVARIANCE MATRIX OF ESTIMATE" in the NONMEM report, and similarly for .cor and .coi.

Note that the ORDER option does not affect the NONMEM report file, only the raw and additional output files. The SIGMA matrix always printed in the same order as OMEGA, and the ORDER option applies to it as well.

REFERENCES: Guide Introduction_7

OUTPUT FRACTION PARAMETER

MEANING: Output fraction (F0) parameter for PREDPP
CONTEXT: Additional PK Parameters

USAGE:

```
$PK
 F0= ....
```

DISCUSSION:

The output fraction parameter is used with PREDPP. It is an optional additional PK parameter. With NM-TRAN, it are symbolized in the $PK block by any one of the reserved variables `FO`, `F0`, or `F`n, where n is the compartment number of the output compartment.

With any of the kinetic models a (peripheral) output compartment is always present. Associated with this compartment is a PK parameter, the output fraction , denoted here by F_o. Of the entire amount, A_o, of drug introduced into the system by various dosage patterns and then eliminated from the system during a given time interval, a fraction F_o of A_o goes into this output compartment.

If the output compartment is never turned on, the output fraction can be ignored. If the value of the output fraction is not computed in PK, it is always understood to be 1

The use of F_o depends on the assumption that the rate of change of drug amount in the output compartment is linear in the other compartment amounts. Other than this linearity restriction, the system can be nonlinear.

Tyically, the output compartment is used for urine observations, and the central for plasma observations, with any ADVAN.

(See **cmt**, **pk**, **$pk**, **default_compartment**).

REFERENCES: Guide IV, section V.C.5
REFERENCES: Guide V, section 6.9
REFERENCES: Guide V, section 7.4.1
REFERENCES: Guide V, section 7.4.5
REFERENCES: Guide V, section 7.4.3.3
REFERENCES: Guide VI, section III.F.7

PARALLEL COMPUTING

MEANING: The parallel computing feature of NONMEM

Parallel computing is described in the INTRODUCTION TO NONMEM 7.

(See **nmfe**).

REFERENCES: Guide Introduction_7

PARAMETER DIMENSIONS

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

USAGE:

```
USE NMPRD_INT, ONLY: NTHES_=>NWTHT,NETAS_=>NWETA, NEPSS_=>NWEPS
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: NWTHT,NWETA,NWEPS
```

DISCUSSION:

`NTHES_`
: The dimension of theta. If the dimension is 0, NTHES_=1.

`NETAS_`
: The dimension of omega. If the dimension is 0, NETAS_=1.

`NEPSS_`
: The dimension of sigma. If the dimension is 0, NEPSS_=1.

Location prior to NONMEM 7: rocm35

REFERENCES: None.

PARAMETER VALUES: INITIAL AND FINAL

MEANING: NONMEM read-only global variables
CONTEXT: PRED routine

USAGE:

```
USE ROCM_REAL, ONLY: THETAF,OMEGAF,SIGMA,THETAFR
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: LTH,LVR ,DPSIZE
REAL(KIND=DPSIZE) :: THETAF(LTH),OMEGAF(LVR,LVR), &
                     SIGMAF(LVR,LVR),THETAFR(LTH)
```

DISCUSSION:

`THETAF`

THETAF(i) = zero, initial, or final value of theta(i), according to the current value of ICALL.

`THETAFR`

THETAFR(i) = zero, initial, or final value of reported thetar(i), according to the current value of ICALL. If record $THETAR is not used, THETAF and THETAFR are equal. If record $THETAR is used, then THETAF is the internal theta as used in $PK/$PRED, and THETAFR is the theta reported in the report file.

`OMEGAF`

OMEGAF(i,j) = zero, initial, or final value of omega(i,j), according to the current value of ICALL.

`SIGMAF`

SIGMAF(i,j) = zero, initial, or final value of sigmaf(i,j), according to the current value of ICALL.

At ICALL = 0, these are zero values.

At ICALL = 1, these are initial values.

If NONMEM will be computing the initial value of theta(i), at ICALL=1 the initial value THETAF(i) is set to some point between the lower and upper bounds.
If NONMEM will be computing the initial value of a diagonal block of OMEGA (SIGMA), at ICALL=1 the initial value of the block is set to the identity matrix.

At ICALL = 3, these are final values.

At ICALL = 2, the values of these variables should not be used unless the value of COMACT is 1 or 2, in which case these are final values.
(See **COMACT,COMSAV**)

Location prior to NONMEM 7: rocm6

REFERENCES: None.

PARAMETERS OMEGA SIGMA: CURRENT

MEANING: NONMEM read-only global variables
CONTEXT: PRED routine

USAGE:

```
USE ROCM_REAL, ONLY: OMEGA=>VARNF
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: LVR,DPSIZE
REAL(KIND=DPSIZE) :: VARNF(LVR,LVR)
```

DISCUSSION:

OMEGA
The current value of OMEGA being used.

The current value of SIGMA is also located in this array as follows:
SIGMA(I,J)=OMEGA(NETAS_+I,NETAS_+J)
NETAS_ is described in Parameter dimensions.

(See **Parameter dimensions**).

At run and problem initialization and at problem finalization use the variables OMEGAF and SIGMAF.
(See **Parameter values: Initial and Final**).

Should not be used if an initial estimate is being computed.

When an initial estimate is being computed, the value is just that found at problem initialization (when ICALL=1).

EXAMPLE:

Compute individual weighted residuals using a slope-intercept residual error model:

```
$ERROR
Y=F+EPS(1)+F*EPS(2)
IF (COMACT.GE.1) THEN
   STD=SQRT(SIGMA(1)+F**2*SIGMA(2))
   IWRES=(DV-F)/STD
ENDIF
```

Location prior to NONMEM 7: rocm22

REFERENCES: None.

PARTIAL DERIVATIVE INDICATORS

MEANING: NONMEM read-only global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPRD_INT, ONLY: MFIRST=>IFRSTDER,MSEC=>ISECDER,IFIRSTEM
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: IFRSTDER,ISECDER,IFIRSTEM
```

DISCUSSION:

`MSEC`

MSEC=1 when NONMEM is expecting second-partial eta-derivatives with the current call to PRED.
MSEC=0 when NONMEM is ignoring second-partial eta-derivatives with the current call to PRED.

Second-partial eta-derivatives are never expected unless the Laplacian method is used.

`MFIRST`

MFIRST=1 when NONMEM is expecting first-partial eta-derivatives with the current call to PRED.
MFIRST=0 when NONMEM is ignoring first-partial eta-derivatives with the current call to PRED. This variable is used only by PREDPP, not by generated FSUBS.

Location of MFIRST and MSEC prior to NONMEM 7: rocm12

`IFIRSTEM`

With NONMEM 7.2 and higher, first-partial eta-derivatives are computed by generated code in FSUBS for classical NONMEM methods, but not for IMP, SAEM, and BAYES methods. This improves the speed at which the problem is evaluated. However, on occasion such derivatives are needed, for example, when steady state values are to be calculated, or when stochastic differential equations are to be evaluated. In such cases, insert as the first line in a each block of abbreviated code ($PK, $ERROR, $DES, $AES, $PRED) the following line of code:

```
 FIRSTEM=1
```

First derivatives will be evaluated for the new methods as well.

Note that FIRSTEM is a local variable. In FSUBS, FIRSTEM is copied from IFIRSTEM in MODULE NMPRD_INT prior to being used:

```
  FIRSTEM=IFIRSTEM
 ...
  IF (FIRSTEM == 1) THEN
   block of first derivative code
  ENDIF
```

The statement `FIRSTEM=1` is inserted in generated code prior to the first test of FIRSTEM. Thus, the local variable FIRSTEM is changed by the user, not the global variable IFIRSTEM.

In order to implement this feature, NM-TRAN rearranges the order of statements in FSUBS. Statements that compute first-partial eta-derivatives are collected together into blocks of first derivative code to be executed only with FIRSTEM is 1. Option `NOFASTDER` of the $ABBREVIATED record prevents NM-TRAN from doing this

and restores the order of statements in FSUBS to what it was in previous versions. (See **Abbreviated**)

REFERENCES: Guide VI, section III.E.4 , IV.B.2

PASS NEW L2 RECORD: NEWL2

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

USAGE:

```
USE NMPRD_INT, ONLY: NEWL2
```

GLOBAL DECLARATION: `INTEGER(KIND=ISIZE) :: NEWL2`

DISCUSSION:

`NEWL2`

NEWL2 = 1 if the data record is the first of an L2 record.
NEWL2 = 2 otherwise.

If there is no L2 data item, each data record is a different L2 record.

NEWL2 also changes value in conjunction with calls to PASS.

NEWL2 may be used as a right-hand quantity in $PRED and $ERROR blocks, and in an $INFN block in conjunction with PASS.

Location prior to NONMEM 7: rocm17

REFERENCES: None.

PASS NEWIND: NWIND

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

USAGE:
```
USE NMPRD_INT, ONLY: NWIND
```

GLOBAL DECLARATION:
```
INTEGER(KIND=ISIZE) :: NWIND
```

DISCUSSION:

`NWIND`
: The NEWIND value at ICALL 0, 1 and 3. NWIND changes value in conjunction with calls to PASS. (See **pred**, **newind**).

Location prior to NONMEM 7: rocm34

REFERENCES: Guide I, section C.3.5.2

PASS

MEANING: PASS subroutine
CONTEXT: NONMEM utility routine

USAGE:

```
USE SIZES, ONLY: ISIZE
INTEGER(KIND=ISIZE) :: MODE
CALL PASS (MODE)
```

DISCUSSION:

The NONMEM utility routine PASS can be used to read and/or to modify ("transgenerate") data records. This can be done at both the beginning and ending of a problem.

When ICALL=0, 1, or 3, PASS may be called by PRED, or by a subroutine of PRED (such as PREDPP's INFN), or PASS may be called from $PRED or $INFN abbreviated code. With abbreviated code, the variable MODE should be used as the argument to PASS and set as described below.

Input argument:

`MODE`

MODE=0: Initialize PASS

MODE=1: store next data record in DATREC (and in PASSRC of module NMPRD9); any modifications of this record performed by PRED are ignored by NONMEM.

MODE=2: store next data record in DATREC (and in PASSRC of module NMPRD9); any modifications of this record performed by PRED are placed into NONMEM's internal data set.

MODE=3: terminate PASS (can be used only after MODE=1)

Output argument:

`MODE`

Upon return from a call to PASS with MODE=1 or 2, PASS sets MODE=0 when there are no more data records to be passed.

The first call to PASS with MODE=1 or 2 produces the first data record, the second call produces the second data record, etc. A value of NEWIND appropriate for the data record currently in DATREC and PASSRC is also stored by PASS in NWIND of module NMPRD_INT.

The NONMEM utility routine GETETA can be used to obtain subject-specific estimates of the eta's. The values returned for the eta's with each call to GETETA are appropriate for the individual whose data record is currently in DATREC. When PASS is used in abbreviated code, GETETA is used automatically. Any PRED-defined items that are displayed are available in module NMPRD4 with each data record.

The value of MIXEST is appropriate for the individual whose data record is currently in DATREC.

Since the call with ICALL=3 actually occurs before the Table and Scatterplot Steps, new data items can be tabled and scatterplotted.

The user should not use PASS to modify either the ID or MDV data items.

Within an initialization or finalization block, code to test MODE and call PASS is generated by NM-TRAN when abbreviated code such as the following is present:

```
IF (ICALL.EQ.0) THEN
DO WHILE (DATA)
 ... transgeneration statements ...
ENDDO
RETURN
ENDIF
```

(See **Infn_pass_interpolation_example**, **Infn_example**).
(See **Initialization-Finalization block**).

REFERENCES: Guide II, section D.2.2
REFERENCES: Guide VI, section VI.A , Figure 37

PASS: PASSRC

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:
```
USE NMPRD_REAL, ONLY: PASSRC
```

GLOBAL DECLARATION:
```
USE SIZES, ONLY: VSIZE,DPSIZE
REAL(KIND=DPSIZE) :: PASSRC(VSIZE)
```

DISCUSSION:

`PASSRC`
: PASSRC contains the data record at ICALL values 0, 1 and 3. PASSRC changes in conjunction with calls to PASS, at which time transgeneration is permitted.

Location prior to NONMEM 7: nmprd9

REFERENCES: None.

PASTZERO (NM75)

MEANING: PREDPP-PK global variables
CONTEXT: User-supplied routines

USAGE:

```
USE PRDDESLVU,     ONLY: PASTZERO
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: DPSIZE
REAL (KIND=DPSIZE) PASTZERO
```

DISCUSSION:

This variable is reserved with ADVAN18, for use with Delay Differential Equations. Sometimes you may wish to have the equations transition from the past to the present other than at time T=0. In this case, set PASTZERO to a non-zero (including negative) value. For example,

```
$DES PASTZERO=-10.0
```

REFERENCES: Guide Introduction_7

PASTZERO (NM75)

MEANING: PREDPP-PK global variables
CONTEXT: User-supplied routines

USAGE:

```
USE PRRADAR5U, ONLY: PASTZERO
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: DPSIZE
REAL (KIND=DPSIZE) PASTZERO
```

DISCUSSION:

This variable is reserved with ADVAN16 and ADVAN17, for use with Delay Differential Equations. Sometimes you may wish to have the equations transition from the past to the present other than at time T=0. In this case, set PASTZERO to a non-zero (including negative) value. For example,

```
$DES PASTZERO=-10.0
```

REFERENCES: Guide Introduction_7

PHI

MEANING: PHI function
CONTEXT: NONMEM utility routine

USAGE:

```
USE SIZES, ONLY: DPSIZE
REAL(KIND=DPSIZE) :: PHI,X,CUMDIS
CUMDIS= PHI(X)
```

DISCUSSION:
Gives the value of the cumulative distribution function.

Input argument:

`X`

EXAMPLE OF USAGE:

PHI may be used in abbreviated code.

```
A=THETA(1)*EXP(ETA(1))
B=PHI(A)
```

The argument may be a random variable, but no eta partial derivatives are computed.

REFERENCES: None.

PK

MEANING: PK subroutine
CONTEXT: User-supplied subroutine; for use with PREDPP

USAGE:

Versions before NONMEM 7.2:

```
SUBROUTINE PK(ICALL,IDEF,THETA,IREV,EVTREC,NVNT,INDXS,IRGG,GG,NETAS)
USE SIZES,     ONLY: DPSIZE,ISIZE,LVR
REAL(KIND=DPSIZE) :: EVTREC
INTEGER(KIND=ISIZE) :: ICALL,IDEF,IREV,NVNT,INDXS,IRGG,NETAS
DIMENSION :: IDEF(7,*),THETA(*),EVTREC(IREV,*),INDXS(*)
DIMENSION :: GG(IRGG,LVR+1,*)
```

With NONMEM 7.2 and higer:

```
SUBROUTINE PK(ICALL,IDEF,THETA,IREV,EVTREC,NVNT,INDXS,IRGG,GG,NETAS)
USE SIZES,     ONLY: DPSIZE,ISIZE
USE PRDIMS,    ONLY: GPKD
IMPLICIT REAL(KIND=DPSIZE) (A-Z)
REAL(KIND=DPSIZE) :: EVTREC
INTEGER(KIND=ISIZE) :: ICALL,IDEF,IREV,NVNT,INDXS,IRGG,NETAS
DIMENSION :: IDEF(7,*),THETA(*),EVTREC(IREV,*),INDXS(*)
DIMENSION :: GG(IRGG,GPKD+1,*)
```

DISCUSSION:
The PK subroutine is called by PREDPP to obtain values for basic and additional pharmacokinetic parameters. Basic PK parameters are typically the rate constants ("micro-constants") for use in kinetic formulas. PK can compute instead parameters such as clearance and volume, and a translator ("TRANS") subroutine can be used to convert these to rate constants. Additional PK parameters include compartment scale parameters, which PREDPP uses to convert compartment amounts to concentrations, and dose-related parameters such as modeled infusion rates.

Input argument:

`ICALL`

`ICALL=1`: PK has been called for initialization at the beginning of a NONMEM problem; one such call per problem. EVTREC contains the first event record. THETA contains the initial estimates. PK must return values in IDEF, which inform PREDPP what tasks it will perform at later calls. It may also set GG (k,1,1) to 1, indicating that the kth PK parameter will be modeled as a log (a feature which cannot be specified using abbreviated code).

`ICALL=2`: PK has been called to obtain parameter values; multiple calls occur. PK should call GETETA to obtain ETA values for the individual. PK should compute all relevant portions of GG.

`ICALL=4`: PK has been called during the Simulation Step; multiple calls occur. PK should call SIMETA to obtain simulated ETA values for the individual. PK should compute individual-specific parameters for column 1 of GG.

`ICALL=5`: PK has been called during the computation of expectations; multiple calls occur. Such calls occur when the marginal (MRG_) data item is defined in the data set and has non-zero values for some records. If the MRG_ data item has the value 1 or 2, the values of PK-defined items contribute to the expectations computed for these items.

ICALL=6: PK has been called during the computation of raw data averages; Such calls occur when the raw-data data item (RAW_) is defined in the data set and has non-zero values for some records.

THETA
The NONMEM THETA vector.

EVTREC
The PREDPP event record.

INDXS
The values specified in the $INDEX record of the NM-TRAN control stream. (This is the NONMEM Index array starting at position 12, the first position beyond those positions used by PREDPP itself).

NETAS
The number of population etas in the problem.

Output argument:

IDEF
PK should store values in IDEF only when ICALL=1.
Values may be stored in the following elements:
IDEF(1,1)=-9 (required)
IDEF(1,2) is the call limiting element (compare $PK's CALLFL).
Values are:
-2: call with every event record and at additional and lagged dose times.
-1: call with every event record (default)
0: call with first event record and new TIME values
1: call once per individual record

Compartment initialization may be performed by routine PK;
(See **Compartment Initialization: A_0**)
(See **sCompartment Initialization: A_0FLG**).
An initial steady state may be requested by routine PK;
(See **Initial Steady State: I_SS,ISSMOD**)
(See **i_ss**, **initial_condition**).
The value in IDEF(1,3) describes whether PK performs compartment initialization, i.e., whether or not PK initializes elements of the initial state vector A_0(n). Values are:
-1: PK may initialize A_0.
0: PK does not initialize A_0.
1: PK does initialize A_0.
The default used by PREDPP is IDEF(1,3)=-1. However, when compartment initialization is not implemented, then if IDEF(1,3) is set to 0, PREDPP can avoid some time-consuming processing. Indeed, when $PK abbreviated or verbatim code is supplied, and there is no reference to compartment initialization amounts A_0(n) in either the abbreviated or verbatim code, then NM-TRAN sets IDEF(1,3)=0.

Compartment amounts may be used by routine PK;
(See **State Vector: A**).
The value in IDEF(1,4) describes whether PK uses derivatives of compartment amounts (e.g. compartment amounts themselves are used as random variables in arithmetic statements in PK). Values are:
-1: PK may use derivatives of compartment amounts.
0: PK does not use derivatives of compartment amounts.

1: PK uses derivatives of compartment amounts.

The default used by PREDPP is IDEF(1,4)=-1. However, when derivatives of compartment amounts are not used, then if IDEF(1,4) is set to 0, PREDPP can avoid some time-consuming processing. Indeed, when $PK abbreviated or verbatim code is supplied, and there is no reference to A(n) (as a random variable in an arithmetic statement) in the abbreviated code (or to derivatives of A(n) in the verbatim code), then NM-TRAN sets IDEF(1,4)=0.

Remaining elements contain row numbers in GG:
`IDEF(2,1)`: row number of F0 (output fraction)
`IDEF(2,2)`: row number of XSCALE (Time Scale)
`IDEF(2,3)`: row number of lowest-numbered MTIME
`IDEF(2,4)`: row number of highest-numbered MTIME
`IDEF(3,n)`: row number of Sn (Scale for comp. n) (thru n+1 output)
`IDEF(4,n)`: row number of Fn (bioavailability fraction for comp. n)
`IDEF(5,n)`: row number of Rn (modeled rate for comp. n)
`IDEF(6,n)`: row number of Dn (modeled duration for comp. n)
`IDEF(7,n)`: row number of ALAGn (absorption lag for comp. n)

`GG` The array of PK parameters and their eta derivatives. The maximum number of rows in GG is given by IRGG, which is the same as constant PG found in file SIZES (See **sizes**).
At ICALL = 2:
`GG(k,1,1)` contains the value of the kth parameter.
`GG(k,i+1,1)` contains its derivative with respect to the ith eta.
`GG(k,i+1,j+1)` contains its second derivative with respect to the ith eta and the jth eta (lower-triangular; j=1, ..., i). (Second derivatives are only needed with estimation by the Laplacian method.)
At ICALL = 4:
`GG(k,1,1)` contains the individual-specific value of the kth parameter. Other columns of GG need not be computed.

Also see variables in NONMEM modules, NONMEM-PRED modules, and PREDPP modules.
(See **variables in modules**)

REFERENCES: Guide IV, section V.C.5
REFERENCES: Guide V, section 7
REFERENCES: Guide VI, section III

POPULATION SINGLE-SUBJECT INDICATOR

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

USAGE:
```
USE NMPRD_INT, ONLY: IPS
```

GLOBAL DECLARATION:
```
INTEGER(KIND=ISIZE) :: IPS
```

DISCUSSION:

`IPS`

IPS = 1 when the data are population data.
IPS = 2 when the data are single-subject data.

Location prior to NONMEM 7: rocm29

REFERENCES: Guide I, section B.1
REFERENCES: Guide IV, section II.C.4

PRED ERROR MESSAGE

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPRD_CHAR, ONLY :: ETEXT
```

GLOBAL DECLARATION:

```
CHARACTER(LEN=132) :: ETEXT(3)
```

DISCUSSION:

`ETEXT`

The user message which appears as part of the PRED error message. When PRED returns to NONMEM with IERPRD>0, PRED may store here one to three lines of text explaining the error, for printing. The number of lines of text to be printed must be stored in NETEXT. (See **PRED Exit Code**)

When abbreviated code is used, the EXIT statement causes appropriate text to be stored in ETEXT.

Location prior to NONMEM 7: nmprd2

REFERENCES: Guide VI, section III.K.2 , IV.F

REFERENCES: Guide IV, section IV.G

PRED EXIT CODE

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

With versions of NONMEM before 7.4.2:

USAGE:
```
USE NMPRD_INT, ONLY: IERPRD,NETEXT
```
GLOBAL DECLARATION:
```
INTEGER(KIND=ISIZE) :: IERPRD,NETEXT
```
With versions of NONMEM starting with 7.4.2:

USAGE:
```
USE NMPRD_INT, ONLY: IERPRD,IERPRDU,NETEXT
```
GLOBAL DECLARATION:
```
INTEGER(KIND=ISIZE) :: IERPRD,IERPRDU,NETEXT
```
DISCUSSION:

Values are stored by PRED for use with NONMEM.

`IERPRD`
: The PRED error return code; also called the PRED exit code. Is set to 0 before PRED is called.
IERPRD=0: Normal return
IERPRD=1: PRED is unable to compute. If possible, NONMEM should attempt recovery.
IERPRD=2: PRED is unable to compute. NONMEM should abort the run.

`NETEXT`
: NETEXT=0 to 3: the number of lines of text of an error message stored by PRED in ETEXT
(See **nmprd2**).
(See **PRED Error Message**).

Values may be stored in IERPRD two ways.
PREDPP may store error messages such as
`PK PARAMETER FOR KA IS NON-POSITIVE` in ETEXT and set IERPRD=1 and K=1. (In effect, EXIT 1 1).

In abbreviated code, whether or not PREDPP is used, the statement "EXIT n k" causes n to be stored in IERPRD. The value of k is part of the error message in ETEXT, which is reported in the NONMEM output report and in file PRDERR.

With NONMEM 7.4.2, there is a new variable, IERPRDU. Both n and k are stored in IERPRDU. k must be between 0 and 999. The value stored is n*10000+k. E.g., "EXIT 1 500" is stored in IERPRDU as 10500.

With NONMEM 7.5 and later, values of k between 1000 and 9999 are permitted. E.g., "EXIT 1 2000" is stored in IERPRDU as 12000.

PREDPP will not set these values; only the EXIT statement in abbreviated code can do this. IERPRDU is ignored if it is not the Simuation Step. For NONMEM 7.5 use of the

error code during simulation,
(See **Simulation Block**) and look for "Simulation Error Forgiveness".

Location prior to NONMEM 7: nmprd1

REFERENCES: Guide V, section 12.4.15

REFERENCES: Guide VI, section III.K.2 , IV.F
REFERENCES: Guide IV, section IV.G
REFERENCES: Guide Introduction_7

PRED

MEANING: PRED subroutine
CONTEXT: User-supplied routine; required with NONMEM

USAGE:

Versions before NONMEM 7.2:

```
SUBROUTINE PRED (ICALL,NEWIND,THETA,DATREC,INDXS,F,G,H)
USE SIZES,     ONLY: DPSIZE,ISIZE,LVR
REAL(KIND=DPSIZE) :: DATREC
INTEGER(KIND=ISIZE) :: ICALL,NEWIND,INDXS
DIMENSION :: THETA(*),DATREC(*),INDXS(*),G(LVR,*),H(LVR,*)
```

With NONMEM 7.2 and higher:

```
SUBROUTINE PRED (ICALL,NEWIND,THETA,DATREC,INDXS,F,G,H)
USE SIZES, ONLY: DPSIZE,ISIZE
USE PRDIMS,ONLY: GPRD,HPRD,GERD,HERD,GPKD
IMPLICIT REAL(KIND=DPSIZE) (A-Z)
REAL(KIND=DPSIZE) :: DATREC
INTEGER(KIND=ISIZE) :: ICALL,NEWIND,INDXS
REAL(KIND=DPSIZE) :: G(GPRD,*),H(HPRD,*)
DIMENSION :: THETA(*),DATREC(*),INDXS(*)
```

DISCUSSION:
The PRED subroutine is called by NONMEM to obtain modeled values. (PREDPP is a PRED subroutine that is distributed with NONMEM.)

Input argument:

ICALL

ICALL=-1: the routine has been called for the PRED_IGNORE_DATA feature of NONMEM 7.5. One call per data record, at the start of the run. These calls occur only if abbreviated code uses variables PRED_IGNORE_DATA or PRED_IGNORE_DATA _TEST, or if the PRED_IGNORE_DATA option of $DATA is used. Otherwise, there are no calls to PRED with ICALL=-1.

ICALL=0: PRED has been called for initialization purposes at the beginning of the NONMEM run; one such call per run. DATREC contains the first data record. THETA contains the initial estimates. PRED need not compute F, G, or H.

ICALL=1: PRED has been called for initialization purposes at the beginning of a NONMEM problem; one such call per problem. Otherwise, identical to ICALL=0.

ICALL=2: For the data record contained in DATREC, PRED has been called for the purpose of computing F, the value of the prediction, and/or the values of other PRED-defined items, appropriate for the record. PRED should compute F, and also G and H as appropriate. THETA contains the values to be used to compute F. With conditional estimation, to obtain ETA values, PRED should call GETETA.

ICALL=3: PRED has been called for finalization purposes at the end of a NONMEM problem; one such call per (sub)problem. DATEC contains the first data record. THETA contains the final estimates. Otherwise, identical to ICALL=0.

ICALL=4: For the data record contained in DATREC, PRED has been called during the Simulation Step for the purpose of computing a value of the dependent variable and, possibly, values of independent variables, appropriate for the record. PRED should compute

F (the value of the dependent variable). THETA contains the initial estimates, which are the values to be used to compute F. PRED should call SIMETA (and SIMEPS) to obtain simulated values of ETA (and EPS).

ICALL=5: For the data record contained in DATREC, PRED has been called for the purpose of computing the expectation of the PRED item and possibly, the expectations of other PRED-defined items, appropriate for the record. Such a call occurs when the marginal data item (MRG_) is defined in the data set and has a non-zero value for the data record in question. If the MRG_ data item on the record has the value 1 or 2, the value returned by PRED in F contributes to the expectation of the PRED item. Similarly, the values returned in other PRED-defined items contribute to expectations of these items. THETA contains the final estimates. The expectations in question are over possible values of ETA, and to obtain ETA values, PRED should call GETETA.

ICALL=6: PRED has been called for the purpose of computing the raw data average of the DV data items and, possibly, the raw data averages of PRED-defined items. Such a call occurs when the raw-data data item (RAW_) is defined in the data set and has a non-zero value for a template data record (See **template**). The value of the DV data item in the data record contained in DATREC will be included in the raw data average of the DV data items. However, when the raw data average corresponding to the label DV in a table or scatterplot is to be different from the raw data average of the DV items themselves, PRED may recompute the value of DV. PRED may return a value of 1 in F to omit the DV item in the data record from the average.

NEWIND

NEWIND=0: First record of the data set. THETA value may differ from value at last call with this record.
NEWIND=1: First record of the data set, THETA value does not differ from value at last call with this record, and PRED is nonrecursive (see **I_REC**), or,
First record of a subsequent individual record.
NEWIND=2: Subsequent data record of an individual record.

THETA

The NONMEM THETA vector.

DATREC

The current data record.

INDXS

The values specified in the $INDEX record of the NM-TRAN control stream.

Output argument:

F When ICALL=2, the prediction associated with the data record. With odd-type data, the likelihood of the observation in the record, but if there is no observation, F is ignored.

When ICALL=4, the value of the simulated observation associated with the data record. Alternatively, F can be ignored, and the DV item in the data record can be directly set to the value of the simulated observation. With odd-type data, F is ignored; PRED should directly set the DV item to the value of the simulated observation.

G An array of derivatives of F with respect to etas. Values should be set when ICALL=2.
G(i,1) is the partial derivative of F with respect to eta(i).

When the data are population, G(i,j+1) is the second partial derivative of F with respect to eta(i), eta(j) (lower-triangular; j=1,..., i).
Second derivatives are needed only when the Laplacian method is used to estimate parameters.

`H` An array of partial derivatives of F with respect to epsilons. When the data are population, values should be set when ICALL=2.
H(i,1) is the derivative of F with respect to eps(i).
H(i,j+1) is the partial derivative of H(i,1) with respect to eta(j)
These mixed second derivatives are needed only when the INTERACTION option is used to estimate parameters. (See **$ESTIMATION**).

Also see variables in NONMEM modules, NONMEM-PRED modules, and PREDPP modules.
(See **variables in modules**)

REFERENCES: Guide I, section C.2
REFERENCES: Guide IV, section III.B.8 , IV
REFERENCES: Guide V, section 12.3

PRED,RES,WRES

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied PRED and INFN routines

USAGE:

```
USE ROCM_REAL, ONLY:                                                    &
   PRED_=>APPND(001),RES_=>APPND(002),WRES_=>APPND(003)                 &
   IWRS_=>APPND(004),IPRD_=>APPND(005),IRS_=>APPND(006)                 &
   NPRED_=>APPND(007),NRES_=>APPND(008),NWRES_=>APPND(009)              &
   NIWRES_=>APPND(010),NIPRED_=>APPND(011),NIRES_=>APPND(012)           &
   CPRED_=>APPND(013),CRES_=>APPND(014),CWRES_=>APPND(015)              &
   CIWRES_=>APPND(016),CIPRED_=>APPND(017),CIRES_=>APPND(018)           &
   PREDI_=>APPND(019),RESI_=>APPND(020),WRESI_=>APPND(021)              &
   IWRESI_=>APPND(022),IPREDI_=>APPND(023),IRESI_=>APPND(024)           &
   CPREDI_=>APPND(025),CRESI_=>APPND(026),CWRESI_=>APPND(027)           &
   CIWRESI_=>APPND(028),CIPREDI_=>APPND(029),CIRESI_=>APPND(030)        &
   EPRED_=>APPND(031),ERES_=>APPND(032),EWRES_=>APPND(033)              &
   EIWRES_=>APPND(034),EIPRED_=>APPND(035),EIRES_=>APPND(036)           &
   NPDE_=>APPND(037),ECWRES_=>APPND(038),NPD_=>APPND(039)               &
   OBJI_=>APPND(040)
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: DPSIZE
REAL(KIND=DPSIZE) :: APPND(MAXXNAME)
```

DISCUSSION:

The PRED_,RES_,WRES_ items are the same as PRED, RES and WRES items.

The CPRED_,CRES_,CWRES_ are calculated as if the estimation was performed using a CONDITIONAL method without INTERACTION. (conditional, non-interactive in the manner of Hooker et al.)

The PREDI_,RESI_,WRESI_ are calculated as if the estimation was performed using the FO method with INTERACTION (non-conditional, interactive).

The CPREDI_,CRESI_,CWRESI_ are calculated as if the estimation was performed using a CONDITIONAL method with INTERACTION (conditional, interactive)

EPRED_,ERES_,EWRES_ are Monte-Carlo generated pred, res, and wres values, and are not linearized approximations like the other weighted residual types.

NPDE_ is the normalized prediction distribution error (takes into account within-subject correlations)

NPD_ is the correlated normalized prediction distribution error (does not take into account within-subject correlations)

EWRES_ and NPDE_ and NPD_ are evaluated using predicted function and residual variances evaluated over a Monte Carlo sampled range of etas with population variance Omega.

ECWRES_ is evaluated with only the predicted function evaluated over a Monte Carlo sampled range of etas with population variance Omega, while residual variance is always evaluated at conditional mode or mean. Thus, ECWRES is the Monte Carlo version of CWRES, while EWRES is the Monte Carlo version of CWRESI.

The EPRED_, ERES_, EWRES_, NPD_, NPDE_, ECWRES_ items are calculated by the Table Step. When EPRED_, ERES_, EWRES_, NPD_, NPDE_, ECWRES_ items are displayed, the options ESAMPLE and SEED of the $TABLE record are of interest. (See **$table**).

The PRED_,RES_,WRES_ (PRED,RES,WRES) items behave as they did with NONMEM VI. Consequently, if FO or a CONDITIONAL method without INTERACTION is used, NPRED_,NRES_,NWRES_ and PRED_,RES_,WRES_ are equivalent. If FO or a CONDITIONAL method with INTERACTION is used, IPRED_,IRES_,IWRES and PRED_,RES_,WRES_ are equivalent.

These items are available with passes through the data set during problem finalization (i.e. ICALL=3), in a data record specific way.

These items may be used as right-hand quantities in $PRED and $INFN during problem finalization. They can also be output in table files by specifying their names (without the underscore) on a $TABLE statement.

(See **pred res**, **wres**).

Location prior to NONMEM 7: rocm49

For other variables, see the INTRODUCTION TO NONMEM 7.

REFERENCES: Guide I, section C.3.5.3 , C.3.5.4
REFERENCES: Guide IV, section III.B.16 , III.B.17
REFERENCES: Guide V, section 9.5 , 10.7 , 11.4.4.2
REFERENCES: Guide Introduction_7

PRED-DEFINED VARIABLES

MEANING: NONMEM-PRED global variables
CONTEXT: User-supplied routines

USAGE:

```
USE NMPRD4, ONLY: COM=>VRBL
```

GLOBAL DECLARATION:

```
MODULE NMPRD4
USE SIZES, ONLY: DPSIZE
IMPLICIT NONE
SAVE
 REAL(KIND=DPSIZE), ALLOCATABLE, TARGET :: VRBL(:)
END MODULE
```

DISCUSSION:

NMPRD4 is a module for PRED-defined variables (including PK-defined and ERROR-defined variables). It contains an array VRBL. (COM is suggested above as an alternate to the name VRBL to be consistent with generated code, but this is optional.) The VRBL array is allocated dynamically by NONMEM. The default size is given by constant LNP4 in resource/sizes.f90, but this can be changed using the $SIZES record.

Variables are stored in NMPRD4 for communication with other blocks of abbreviated code or with user-written codes, or so that NONMEM has access to these variables for display in tables or scatterplots. (See **$table, $scatter**). Values are stored by PRED and by subroutines of PREDPP. A SAVE region of NMPRD4 may be designated; (See **COMACT,COMSAV**).

When abbreviated code is used, by default NM-TRAN lists all PRED-defined variables in the module, as well as certain NM-TRAN-defined variables which give the values of partial derivatives. Comment lines in the generated code (file FSUBS) identify these variables. This makes the listed variables globally accessible to all blocks of abbreviated code, which may not be desirable. (An exception is made for initialization-finalization variables; these are PRED-defined variables that are first defined within a block that tests for ICALL 0, 1, 3. Such variables are stored locally and will not be changed by NONMEM at ICALL 2 or 4.) (See **INFN-defined_variables.**)

Variables from a given block of abbreviated code can be excluded from the module by including the following pseudo-statement in that block:

```
COMRES=-1
```

Variables from all blocks of abbreviated code can be excluded from the module by including the identical option `(COMRES=-1)` on the $ABBREVIATED record.

Variables from NMPRD4 can be used in a user-written subroutine. If the user-written subroutine contains the USE NMPRD4 statement as shown above, the variables have the names COM(1), COM(2), etc. E.g., `X=COM(1)/COM(1)`. It is possible to give the variables the same (non-subscripted) names that they had in $PRED, $PK or $ERROR. Suppose variables CL and V are used in $PK, and the following is present in the generated PK routine:

CL=>COM(00001); V=>COM(00002)

In the user-written subroutine, after the last declaration, include:

POINTER :: CL,V

Prior to the first executable statement, include:

CL=>COM(1); V=>COM(2)

Now the code is `X=CL/V`.

Code generated by NM-TRAN is somewhat more complex, but achieves the same result.

REFERENCES: Guide III, section V.3.3
REFERENCES: Guide IV, section III.B.7 , III.B.16 , III.B.17 , IV.H , V.C.5-9
REFERENCES: Guide VI, section III.J , III.L.2 , IV.E , IV.G.4 Figures 5, 14

PREDPP MODULES

MEANING: Global variables in PREDPP
CONTEXT: User-supplied routines

DISCUSSION:

These modules contain values that are (for the most part) communicated from PRED to its subroutines. Prior to NONMEM 7 these values were COMMON blocks. Following is a list of the MODULE, the (old) COMMON, and a description of the variables. (See **NONMEM_modules**).

PREDPP read-only modules contain values that are communicated from PREDPP to various user-subroutines PK, ERROR, DES and AES.

PROCM_INT (PROCM1)
: `NEWIND`, for PK and ERROR

PROCM_REAL (PROCM2)
: Non-event dose time and derivatives, for PK

PROCM_REAL (PROCM3)
: Initiating dose record (at non-event dose time), for PK

PROCM_REAL (PROCM4)
: Compartment amounts and derivatives, for PK and ERROR

PROCM_INT (PROCM5)
: Number and map of active etas, for PK and ERROR (and TRANS)

PROCM_REAL (PROCM6)
: Theta vector from NONMEM, for DES and AES

PROCM_REAL (PROCM7)
: EVTREC from NONMEM, for DES and AES

PROCM_CHAR (PROCM8)
: Format statements, for all routines

PROCM_REAL (PROCM9)
: Time at which the state-vector (in PROCM4) was last computed, for PK

PROCM_INT (PROCMA)
: Indicator variables associated with model event times, for all routines

PROCM_INT (PROCMB)
: Flag indicating final call to DES or AES after advance to event or non-event time, for DES and AES

PROCM_INT (PROCMC)
: Flag indicating compartment initialization call to PK, for PK

The following are special modules containing values that are communicated from user routines to PREDPP.

PRCOM_INT (PRCOMG)
: Override default settings in ADVAN6, ADVAN8, ADVAN9, ADVAN13,and SS6

PRCOM_LOG (PRCOMU)
: Restore pre-1990 behavior of bio-availability fraction

PRCOM_REAL (PRCOMW)
: Fudge factor for error tests in ADVAN7/SS7

These modules contain values that are communicated from subroutines to PREDPP.

PKERR_REAL (PRDPK1)
: Communicate model event time parameters computed by PK to the ERROR routine

PRMOD_INT (PRDPK2)
: Flag indicating possible change in model event time parameters

PRMOD_REAL (PRDPK3)
: Compartment initialization values from PK

PRMOD_INT (PRDPK4)
: Values of ISSMOD and I_SS ("Initial Steady-State") flags.

PROCM_INT (PRDDE1)
: Initialization values that are communicated from subroutine DES and AES to PREDPP.

PRINFN (PRINFN)
: Contains values of INFN-defined variables which are communicated to other PREDPP user-routines.

Each has its own entry in the Help document.

REFERENCES: Guide VI, section III.I , IV.D , Figure 14

PRED IGNORE DATA BLOCK (NM75)

MEANING: Abbreviated code for ignoring (dropping) data records
CONTEXT: $PRED, $PK, $INFN abbreviated code

SAMPLE:

```
$INFN
    IF(PRED_IGNORE_DATA_TEST==1) THEN
    PRED_IGNORE_DATA=0
    IF(AGE>35.0) PRED_IGNORE_DATA=1
    IF( ID>10.AND.ID<18.OR.ID>60.AND.ID<70 ) PRED_IGNORE_DATA=1
    RETURN ;Assures no additional computation code in INFN is executed
    ENDIF
```

USAGE:

```
      SUBROUTINE INFN (ICALL,THETA,DATREC,INDXS,NEWIND)
      USE NMPRD_INT, ONLY: PRED_IGNORE_DATA,PRED_IGNORE_DATA_TEST
```

DISCUSSION:

The IGNORE=(list) and ACCEPT=(list) options of $DATA provide a limited means of filtering the input data set, which is performed by NMTRAN. To provide more elaborate filtering for excluding data, PRED can request that NONMEM filter out additional data records at the beginning of the run.

This is done by setting the reserved variable PRED_IGNORE_DATA to a non-zero value within $INFN, $PK, or $PRED, for each record to be ignored.

It may be useful to package PRED_IGNORE_DATA statements within

```
 IF(PRED_IGNORE_DATA_TEST==1) THEN
 ...
 RETURN
 ENDIF
```

structures to avoid unnecessary code execution.

If the PRED_IGNORE_DATA_TEST or PRED_IGNORE_DATA variables appear in abbreviated code, or the option

```
$DATA ... PRED_IGNORE_DATA
```

is used in the NM-TRAN control stream, then a PRED_IGNORE_DATA pass through the NONMEN data file with PRED_IGNORE_DATA_TEST=1 and ICALL=-1 occurs. Otherwise it does not. Therefore, existing code such as "IF (ICALL<=1) THEN ...ENDIF" does not need to be changed.

The following variables have properly defined values:

`ICALL`
Data record items in `DATREC`
`NEWIND`,`NEWL2`
`NPROB`,`IPROB`, `S1NUM`, `S2NUM`,
`S1NIT`,`S2NIT`, `S1IT`, `S2IT`

No other variables are properly defined when PRED_IGNORE_DATA_TEST=1. For example, the following should not be used:
`THETA`, `OMEGA`, `SIGMA`, `NREP`, `IREP`
`ETA` may be used but will be 0 there are no random variables in a pred_ignore_data block.

Typically the NONMEM file that is input to the pred_ignore_data block is FDATA. FDATA is unaffected by the pred_ignore_data block. However, with NONMEM 7.5 there is a new file, FDATA.csv, and records excluded by PRED_IGNORE_DATA will not be present in FDATA.csv.

In a pred_ignore_data block, the data record has been read by NONMEM and all data record items have numeric values. The (non-Fortran) operators .EQN. and .NEN. that can be used with the $DATA IGNORE and ACCEPT options are not needed and cannot be used in a pred_ignore_data block.

Any other functions of $INFN, such as DATA item modification (i.e., transgeneration of the data), RANDOM calls, etc. should be made with ICALL==1 or ICALL==0 IF blocks, as before.

It is possible to restrict PRED_IGNORE_DATA actions to a particular problem number:

```
IF(IPROB==2.AND.PRED_IGNORE_DATA_TEST==1) THEN
PRED_IGNORE_DATA=0
IF(AGE>35.0) PRED_IGNORE_DATA=1
IF( ID>10.AND.ID<18.OR.ID>60.AND.ID<70 ) PRED_IGNORE_DATA=1
RETURN
ENDIF
```

RETURN statements may be used.

EXIT statements may be used. They act like a RETURN but are otherwise ignored.

REFERENCES: Guide Introduction_7

PRED RES WRES

MEANING: PRED, RES, WRES
CONTEXT: NONMEM output

DISCUSSION:
NONMEM tables usually include special items: prediction, residual, and weighted residual, which are generated by NONMEM. The default labels for these items are PRED, RES, and WRES, respectively. Synonyms may be specified on the $TABLE record for one or more of these labels. With the NOAPPEND option on this record, the three items are not included in a table unless explicity listed on the $TABLE record. With the use of the user-routine SPTWO, the values of the RES and WRES items can be defined differently from the values described below.
(See **sptwo**).

These items may also be displayed in scatterplots. Synonyms may be specified on the $SCATTERPLOT record as well; synonyms specified on either the $TABLE or $SCATTERPLOT record also apply to the other record.

PRED

Prediction items are the predictions computed by the PRED subroutine. For population data, prediction items are always population predictions, i.e., they are computed at the mean value of eta (0).

RES

The residual is defined as DV - PRED; that is, the observed value minus the prediction item.

WRES

The weighted residuals for an individual are formed by transforming the individual's residuals so that under the model, assuming the true values of the parameters are given by the estimates of those parameters, all weighted residuals have mean 0, unit variance and are uncorrelated. For population data, the "weights" are computed at eta = 0.

For odd-type data, the prediction items are likelihoods (for population data, using the estimated values of the parameters, and computed at eta = 0). These may not be of much interest. The RES and WRES items are 0.

With a mixture model, each individual is classified into one of the subpopulations of the mixture according to a computation based on the individual's data and on the final parameter estimates. For a data record from the individual record, the prediction, residual, and weighted residual items in the corresponding row of a table (or point on a scatterplot) are based on the submodel defining the subpopulation into which the individual is classified.

If the Marginal data item (MRG_) is 1 or 2 for a given data record, then PRED is an expected prediction, rather than the prediction at the mean value of eta. When the Raw-data data item (RAW_) is 1, then DV is a raw-data average and RES is the difference between the PRED item and this average.
(See **displayed PRED-defined items**).

REFERENCES: Guide I, section C.3.5.3 , C.3.5.4
REFERENCES: Guide IV, section III.B.16 , III.B.17
REFERENCES: Guide V, section 9.5 , 10.7 , 11.4.4.2

PRIOR

MEANING: PRIOR subroutine
CONTEXT: User-supplied subroutine; replaces a NONMEM dummy routine

USAGE:

```
SUBROUTINE PRIOR (ICALL,CNT)
USE SIZES, ONLY: ISIZE,DPSIZE
INTEGER(KIND=ISIZE), INTENT(IN) :: ICALL
REAL(KIND=DPSIZE),   INTENT(IN OUT) :: CNT
```

DISCUSSION:
The user-written PRIOR subroutine allows a penalty function based on a frequency prior to be specified and added to the -2log likelihood function (Gisleskog et al, JPP, 2002, p. 473-505). This serves as a constraint on the *estimates* of THETA, OMEGA, and SIGMA and thus as a way for stable estimates to be obtained with insufficient data.

When the Simulation Step is implemented, THETA and OMEGA, and SIGMA may also be simulated (along with etas and epsilons, should any such appear in the model and have nonzero-variances) in the PRIOR subroutine (See **PRIOR_Simulation:_Parameters**). However, this need not be done (simply execute a return in PRIOR at ICALL=4), in which case the values for THETA, OMEGA, and SIGMA used in the simulation are the values given in the NONMEM control stream or input Model Specification File.

Input argument:

`ICALL`
: Similar to ICALL for PRED subroutine.
Possible values: 0, 1, 2, and 4.
In a multiple problem run, PRIOR is called with ICALL=1 only with the first problem.

Output argument:

`CNT`
: CNT is the penalty term.

Other Inputs:

Other inputs are available to PRIOR in NONMEM read-only modules. (See **NONMEM_modules**).

NONMEM Utility Routines:

PRIOR may call NONMEM utility routines, NWPRI and TNPRI. (See **nwpri**, **tnpri**). E.g.,

```
SUBROUTINE PRIOR (ICALL,CNT)
USE SIZES, ONLY: DPSIZE,ISIZE
REAL (KIND=DPSIZE) :: CNT
REAL (KIND=DPSIZE) :: PLEV
INTEGER (KIND=ISIZE) :: ICALL
NTHETA=3
NETA=3
NEPS=1
NTHP=2
NETP=2
NEPP=1
```

```
PLEV=0
CALL NWPRI (NTHETA,NETA,NEPS,NTHP,NETP,NEPP,NPEXP,ITYP,PLEV, &
     NSAM,ISS,CNT)
RETURN
END
```

NWPRI and TNPRI compute particular types of penalty functions. NWPRI computes a function based on a frequency prior that has a multivariate normal form for THETA, and also, in the case of population data, an inverse Wishart form for OMEGA (independent from the normal for THETA). When used during a Simulation Step, it produces a random value of THETA and a random value of OMEGA from the frequency prior. The prior information is entered manually into the control stream. This prior is especially useful for a subjective specification of a penalty function. (See **nwpri**, tnpri).

Model parameters may be constrained in various ways. Those parameters whose values are not fixed are transformed to new parameters whose values are not at all constrained a priori (the "unconstrained parameters", or UCP). TNPRI computes a penalty function based on a frequency prior that has a multivariate normal form for all UCP. When used during a Simulation Step, it produces a random value of the vector of all model parameters (whose values are not fixed). The prior information is found in an input model specification file, and it has been automatically stored there as part of a NONMEM analysis of a prior data set. (See **$msfi**, **model_specification_file**). This prior is especially useful for an empirical specification of a penalty function. (See **TNPRI**).

The $PRIOR statement can be used instead of a PRIOR subroutine.
(See **$prior.ctl**).

REFERENCES: Guide V, section 12.4.16

PRIOR SIMULATION: ICMAX

MEANING: NONMEM-PRED global variables
CONTEXT: PRIOR routine

USAGE:

```
USE NMPR_INT, ONLY: ICMAX=>IMAXSIM
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: IMAXSIM
```

DISCUSSION:

This variable allows the user to control the behavior of the NWPRI and TNPRI NONMEM utility routines. It is relevant during simulation when PRIOR sets ISS to a nonzero value. By default, NWPRI and TNPRI attempt at most 100 times to obtain a sample. After that number of attempts, they terminate with the message

```
MAXIMUM ATTEMPTS TO OBTAIN SAMPLE EXCEEDED
```

ICMAX

If a value is set in ICMAX, this overrides the default of 100. E.g., in a PRIOR subroutine:

```
   USE NMPR_INT, ONLY: ICMAX=>IMAXSIM
... other code ...
   ICMAX=500
```

Location prior to NONMEM 7: nmpr50

REFERENCES: None.

PRIOR SIMULATION: PARAMETERS

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPR_REAL, ONLY: THSIMP=>THET_P,OMSIMP=>OMEG_P,SGSIMP=>SIGM_P,THSIMPR
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: LTH,LVR,DPSIZE
REAL(KIND=DPSIZE) :: THET_P(LTH),OMEG_P(LVR,LVR),SIGM_P(LVR,LVR)
```

DISCUSSION:

Values of THETA, OMEGA and SIGMA that are produced during a Simulation Step using the user-supplied routine PRIOR must be stored in these variables. If one of the NONMEM utility routines NWPRI or TNPRI is used by PRIOR to produce THETA, OMEGA, and SIGMA values, these values will automatically be stored in THET_P,OMEG_P,SIGM_P. If $THETAR record is not used, THSIMPR contains the same values as THSIMP (THET_P). If $THETAR record, THSIMP contains the internal theta values, and THSIMPR contains the reported values (THSIMP values transformed by equations of the $THETAR record). The entire THETA, OMEGA and SIGMA arrays are stored by NPWRI and TNPRI, not just the the inital sub-vectors (prior-affected parts). (See **nwpri**, **tnpri**).

These variables may be used as right-hand quantities in $PK, $ERROR, $INFN and $PRED blocks. After being set during the Simulation Step, they remain available during problem finalization (i.e., ICALL=3).

Location prior to NONMEM 7: nmpr16

REFERENCES: None.

PRIORGET

MEANING: Transfer Results of an Analysis to NMTRAN Prior Information (NM75)
CONTEXT: NONMEM run

This utility extracts results from a raw output file (.ext) and a previous analysis into informative prior values for a subsequent analysis.

USAGE: `priorget root.ext root.cov myprior.ctl TBLN s/,/t`

where root.ext and root.cov are input the files from a previous analysis. TBLN is the table number, as these input files may have several tables to choose from. Delimiter can be , t for tab, or s for space. Optional if delimiter is space. The order of the parameters in the input files must be in the default TSOL (lower triangular) order.

EXAMPLE:

priorget example3.ext example3.cov prior3.ctl 5 s

extracts information from table 5 of the population mixture problem example3, and places it in NMTRAN $PRIOR control stream format in the output file prior3.ctl

See **INTRODUCTION TO NONMEM 7, priorget: Transfer Results of an Analysis to NMTRAN Prior Information**

You can then patch (or "include") this code into your control stream for the subsequent analysis. Any fixed thetas will have $THETAPV variances with value 0, so you will need to modify this as needed. The degrees of freedom for Omegas and Sigmas ($OMEGAPD, $SIGMAPD) are calculated according to the formula described in section "A Note on Setting up Prior Information."

Be aware that the degrees of freedom may fall below the block dimension, or be above the total number of subjects, and the user may wish to modify the results accordingly before using them in an analysis. Results from priorget should always be treated as informational first, and then inspected and modified according to your intended purpose.

REFERENCES: Guide Introduction_7

PROBABILITY DENSITY FUNCTIONS

MEANING: Functions that may be used in abbreviated code.
CONTEXT: Fortran coded function

A series of built in probability density functions are available with NONMEM 7.4.2. For a given probability density there is also a cumulative distribution function (densitycdf), and random number generating function (density_rng).

They are described in INTRODUCTION TO NONMEM 7.4.2 Section I.26.

The density and densityCDF functions have arguments that are compatible with the FUNC system, in which function provides derivatives (XD), and second derivatives (XDD) (see I.65.Expanded Syntax and Capacity for User-Defined Functions (FUNCA) (NM74)). Thus, even random (eta associated) variables may serve as arguments to the parameters of the density functions. The source code of these densities are in ..\source\DISTRIB.f90, DISTRIBCDF.f90, and DISTRIBRNG.f90. Note that multi-variate densities do not have a corresponding CDF routine.

Examples of their use are in the ..\examples\densities directory.

Here are the list of densities, which are modeled after the format from the Stan manual:

BERNOULLI
BERNOULLILOGIT
BINOMIAL
BINOMIALLOGIT
BETABINOMIAL
HYPERGEOMETRIC
CATEGORICAL
CATEGORICALLOGIT
ORDEREDLOGISTIC
NEGBINOMIAL
NEGBINOMIAL2
NEGBINOMIAL2LOG
POISSON
POISSONLOG
MULTINOMIAL
NORMAL
EXPMODNORMAL
SKEWNORMAL
STUDENTT
DOUBLEEXPONENTIAL
LOGISTIC
GUMBEL
LOGNORMAL
CHISQUARE
INVCHISQUARE
SCALEDINVCHISQUARE
EXPONENTIAL
GAMMA
INVGAMMA

WEIBULL
FRECHET
RAYLEIGH
PARETO
PARETO2
BETA
DIRICHLET
VON MISES

REFERENCES: none

PROBLEM ITERATION COUNTERS

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

USAGE:

```
USE NMPRD_INT, ONLY: NPROB,IPROB,S1NUM=>IDXSUP(1),S2NUM=>IDXSUP(2), &
                     S1NIT=>NITR_SUP(1),S2NIT=>NITR_SUP(2), &
                     S1IT=>NITSUP(1),S2IT=>NITSUP(2)
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: NPROB,IPROB,IDXSUP(2),NITR_SUP(2),NITSUP(2)
```

DISCUSSION:

`NPROB`
: The number of problems in the run.

`IPROB`
: The number of the current problem.

  `IPROB` is 1 unless there are multiple problems (e.g., multiple $PROBLEM records in the NM-TRAN control stream).

`S1NUM`
: The number of the active first-level superproblem.
  `S1NUM` is 0 if there is no active first-level superproblem.

`S2NUM`
: The number of the active second-level superproblem.
  `S2NUM` is 0 if there is no active second-level superproblem.

`S1NIT`
: The number of iterations for the active first-level superproblem.
  `S1NIT` is 0 if there is no active first-level superproblem.

`S2NIT`
: The number of iterations for the active second-level superproblem.
  `S2NIT` is 0 if there is no active second-level superproblem.

`S1IT`
: The number of the current iteration of the active first-level superproblem.
  `S1IT` is 0 if there is no active first-level superproblem.

`S2IT`
: The number of the current iteration of the active second-level superproblem.
  `S2IT` is 0 if there is no active second-level superproblem.

These variables may be used as right-hand quantities in abbreviated code for initialization/finalization blocks.

(See **$super**).

Location prior to NONMEM 7: rocm14

REFERENCES: None.

PROTECT FUNCTIONS

MEANING: PROTECT functions
CONTEXT: Source code

USAGE:

```
X=..
XLOG10=PLOG10(X)
```

DISCUSSION:
As of NONMEM 7.4, a series of routines are available that protect against domain violations, divide by zero, and floating point overflows. Each of these routines start with the letter P, followed by the name of the mathematical operation they are to perform. For example, PLOG is the protective code routine that performs the LOG operation. In addition, there are first derivative (such as PLOGD1), and second derivative (such as PLOGD2) companion routines available which NMTRAN uses for computing analytical derivatives. The source code of these routines are available in ..\source\PROTECT.f90. If you wish to modify their behavior, then copy PROTECT.f90 to your run directory as (say) myprotect.f90, modify it, then refer to this modified code with $SUBROUTINES OTHER=myprotect.f90 (The OTHER option is not needed if the P functions are coded explicitly or implicitly via $ABBR PROTECT.)

The following protective code routines are available:

For all routines, if X=not a number, X is converted to machine precision value, which is about 1.0E-15, before performing an operation on it. If X>INFNTY (where INFNTY is approximately 1.0E+154), then X is converted to INFNTY before an operation is performed on it.

PLOG(x): returns LOG of x. If x<SMALLZ, where SMALLZ is approximately 2.8E-103, then LOG(SMALLZ) is returned.

PLOG10(x): returns LOG10 of x. If x<SMALLZ, where SMALLZ is approximately 2.8E-103, then LOG10(SMALLZ) is returned.

PSQRT(x): returns SQRT of x. If x=0.0d+00, then 0 is returned.

PEXP(x): returns EXP of x. If x>40.0, then PEXP(100.0) is returned (avoids overflow).

PDZ(x): returns 1/x . Protects against divide by zero. If abs(x)<SMALLZ, then 1/SMALLZ is returned.

PZR(x): returns x. protects against zero. If abs(x)<SMALLZ, then SMALLZ is returned.

PNP(x): returns x. Protects against non-positive. If X<SMALLZ, then SMALLZ is returned.

PHE(x): returns x. Protects against high exponent. If X>100, then 100 is returned. Thus PEXP(x)=EXP(PHE(x)).

PNG(x): returns x. Protects against negative. If X<0.0, then 0.0 is returned.

PTAN(x): returns tan(x). Protects against returning infinity on inputs near pi/2.

PATAN(x): returns atan(x). Protects against large intputs.

PACOS(x), PASIN(x): returns acos(x), asin(x), respectively. If |X| is between 1.0 and 1+10**(-08), then x is submitted as 1 or -1. So, "dirty ones" are cleaned up, but

values clearly beyond 1 are allowed to trip up the function, so the user is aware of the logical error in the code, and fix the issue.

Instead of replacing various operations with protected code operations by hand, you can ask NMTRAN to automatically convert your code to protected code with the following statement:

$ABBR PROTECT

NMTRAN will automatically replace all LOG (or DLOG) with PLOG, EXP (or DEXP) with PEXP, SQRT (or DSQRT) with PSQRT, / operations with *PDZ(), and B**E operations with PEXP(E*PLOG(B)), and so on.

Note that the any P function name that is used explicitly or implicitly via $ABBR PROTECT may not be used for a user-defined variable name.

When you use $ABBR PROTECT, you will find a considerable improvement in estimation stability, regardless of estimation method used.

REFERENCES: Guide Introduction_7

PRRES

MEANING: PRRES subroutine
CONTEXT: NONMEM routine

DISCUSSION:
PRRES is a NONMEM routine. It is called with final parameter estimates.

An error message may indicate that NONMEM has experienced difficulties in PRRES with the final parameter estimates. (When the Estimation Step is not implemented, these are the same as the initial parameter estimates.)

PRRES may not be called by user-written code.

REFERENCES: None.

RANDMT,RANDMTU

MEANING: RANDMT and RANDMTU functions
CONTEXT: NONMEM utility routine

BACKGROUND:

The RANDOM subroutine may be used with all versions of NONMEM. The statement CALL RANDOM(K,R) obtains a random number R from random source K. It may be used in the Simulation Step (ICALL=4). The seed is described on the $SIMULATION record for the Kth source. It may also be called when data averages are being computed (ICALL=5). RANDOM can compute only from the NORMAL or UNIFORM distributions.

DISCUSSION:

Functions RANDMT and RANDMTU may be used with NONMEM 7.5 and later. They can be used to obtain random numbers during the Estimation or other steps, with seeds other than the ones used by NONMEM, or with other distributions.

Function RANDMT is simliar to the RANDOM subroutine, and allows the Student T distribution to be used in addition to normal and uniform distributions.

Function RANDMTU is independent of the RANDOM subroutine. It may be used with all NONMEM steps. The user has complete control and may specify starting seed values in the abbreviated code.

USAGE:

These two USE statements are required.

```
USE NM_INTERFACE,ONLY: RANDMT
USE NM_INTERFACE,ONLY: RANDMTU
```

They can be obtained with the statement `include nonmem_reserved_general.`

USAGE of RANDMT:

```
RVAL1=RANDMT(N)
```

Where
N: integer
0: get uniform random variable
N=1 or N>101: get normal random variable
N>1 and N<=100: get Student-t random variable with N-1 degrees of freedom

For more details about RANDMT,
See **INTRODUCTION TO NONMEM 7: RANDMT Function**

USAGE of RANDMTU:

```
RVAL2=RANDMTU(N,SOURCE,STARTSEED,RANM)
```

Where
N: integer (same as with RANDMT)
0: get uniform random variable
N=1 or N>101: get normal random variable
N>1 and N<=100: get Student-t random variable with N-1 degrees of freedom

SOURCE: Integer between 0 and 20, allowing the user to maintain up to 20 distinct seed sources. IF SOURCE=0, it uses the default source.

STARTSEED: Integer, which if not 0, causes initialization with the designated starting seed. STARTSEED should be set to 0 after this initialization. If SOURCE=0, this argument is ignored. If STARTSEED is <0, then the starting seed will be 10000*(seconds after midnight)+ABS(STARTSEED). (This is similar to the CLOCKSEED option of the $SIMULATION record, which allows a control stream to produce different stochastic results for automated replications without the need to modify the seed value in the control stream file in each replication.) The actual startseed is stored in RANDMTU_STARTSEED(SOURCE), module NM_BAYES_INT, accessible by using nonmem_reserved_general (see example below).

RANM: Integer, indicating the type of random number generator algorithm to use, as follows:
0: ran0 of reference [5], minimal standard generator
1: ran1 of reference [5], Bays and Durham.
2: ran2 of reference [5].
3: ran3 of reference [5], Knuth.
4: NONMEM's traditional random number generator, used as default in $SIMULATION

For more details bout RANDMTU,
See **INTRODUCTION TO NONMEM 7: RANDMTU Function**

EXAMPLE:

In the following, 2 random sources are used, with different starting seeds

```
$ERROR ; may also be used in other records
 ...
include nonmem_reserved_general
 ...
; Make sure NMTRAN is aware of RVAL1 and RVAL2, so it is accessible
; for $TABLE output
RAVL1=0.0
RVAL2=0.0
RVAL3=0.0
IF(NEWIND==0) THEN
; Initialize only at beginning of the data set.
; 3=t-distribution with DF=2
  RVAL1=RANDMTU(3,1,-1234,-1)
  write(*,*) RANDMTU_STARTSEED(1)
RVAL2=RANDMTU(3,2,5678,-1)
ENDIF
RVAL1=RANDMTU(3,1,0,-1)
RVAL2=RANDMTU(3,2,0,-1)
RVAL3=RANDMT(3)
```

Note that calling RANDMT(N) (or equivalantly RANDMTU(N,0,...)) did not require an initialization call (and would be ignored if you did), as this is under the control of $EST, $SIML, or $TABLE.

REFERENCES: Guide Introduction_7

RANDOM MODELS

MEANING: Models for random variables
CONTEXT: Abbreviated code

DISCUSSION:
The following are examples of commonly used models using random variables.

Models for CL in terms of TVCL ("typical value of clearance") and ETA are examples of models expressing population inter-individual variability.

Models for Y involving F ("prediction based on the pharmacokinetic parameters") and ERR are examples of models for intra-individual ("residual") variability. They are used with both population or single-subject data, in which case ERR stands for EPS or ETA, respectively.

Additive models

```
CL=TVCL+ETA(1)
Y=F+ERR(1)
```

Proportional (CCV; Constant Coefficient of Variation) models

```
CL=TVCL*(1+ETA(1))
Y=F*(1+ERR(1))
```

An equivalent way of coding the proportional model is:

```
CL=TVCL+TVCL*ETA(1)
Y=F+F*ERR(1)
```

Exponential models

```
CL=TVCL*EXP(ETA(1))
Y=F*EXP(ERR(1))
```

During estimation by the first-order method, the exponential model and proportional models give identical results, i.e., NONMEM cannot distinguish between them.

During estimation by a conditional estimation method, the exponential and proportional models for inter-individual variability give different results. During simulation, the two models give different results, in both the inter- and intra-individual cases.

Power Function model

```
Y=F+F**P*ERR(1)
```

The Power Function model has both the additive and the CCV error models as special cases, and smoothly interpolates between them in other cases.

Combined Additive and Proportional model (slope-intercept model)

```
Y =F*(1+ERR(1))+ERR(2)
```

Here is an alternative parameterization for the same model when there is no covariance between ERR(1) and ERR(2). Any theta may be used.

```
W=(1+THETA(5)*THETA(5)*F*F)**.5
Y=F+W*ERR(1)
```

REFERENCES: Guide V, section 3 , 4.1 , 7.5 , 8.3
REFERENCES: Guide V, section 8

REFERENCES: Guide VII, section I , III

RANDOM

MEANING: RANDOM subroutine
CONTEXT: NONMEM utility routine

USAGE:

```
USE SIZES, ONLY: ISIZE,DPSIZE
INTEGER(KIND=ISIZE) :: KR
REAL(KIND=DPSIZE) :: R
CALL RANDOM (KR,R)
```

DISCUSSION:
The NONMEM utility routine RANDOM may be called (by PRED, PK or ERROR) during the Simulation Step (ICALL=4) and when data averages are being computed (ICALL=5) to obtain numbers from different random sources.

Input argument:

`KR` An integer variable or integer constant. The index of a random source.

Output argument:

`R` A number from the KRth source. R is double precision with NONMEM 7, and is single precision with NONMEM VI. Each time RANDOM is called, a new number is output.

Random sources are defined using the $SIMULATION record. Typically, the first source is reserved by NONMEM to generate realizations of the eta and epsilon variables and/or to randomly mix individuals into different subpopulations according to the mixing parameter. Thus, typically, RANDOM must be called with KR > 1. However, there are cases when NONMEM itself needs no random sources, and all defined random sources can be used by PRED (see references).

A random source that is specified on the $SIMULATION record as NONPARAMETRIC should not be used to obtain a value of R.

EXAMPLES OF USAGE:

WT (an item in the data record, and a local variable with the same name) is generated with a simulated value having mean 70 and standard deviation 7. The second random source is assumed to be pseudo-normal, which is the default distribution for each random source.

An abbreviated code may use RANDOM explicitly in either a simulation or a data average block. When RANDOM is called in abbreviated code, `R` is reserved and is used for the random number. E.g.,

```
IF (ICALL.EQ.4.AND.NEWIND.NE.2) THEN
CALL RANDOM(2,R)
ENDIF
IF (ICALL.EQ.4) WT=70+7*R
```

REFERENCES: Guide IV, section III.B.13 , IV.I
REFERENCES: Guide V, section 12.4.8

RATE DATA ITEM

MEANING: Dose Rate (RATE) data item for PREDPP
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... RATE ...`

DISCUSSION:
`RATE` labels PREDPP's dose rate (RATE) data item. The dose rate data item is optional. In a dose or reset-dose event record it can take four kinds of values.

0 The dose is an instantaneous bolus dose.

>0 The dose is an infusion, and the value of RATE gives its rate. If this is not a steady-state infusion, the AMT data item must be positive also. If this is a steady-state dose event record, both AMT and RATE may be 0; (See **ss dose**).

-1 The dose is a zero-order bolus dose, i.e., an infusion, whose rate is modeled in the PK routine. As an example, suppose the compartment being dosed is numbered 1 (e.g., the CMT data item contains 1, or it contains 0 and compartment 1 is the default compartment for doses). The $PK abbreviated code must define a value for R1, the rate parameter for compartment 1, e.g., `R1=THETA(4)+ETA(4)`

-2 The dose is a zero-order bolus dose, i.e., an infusion, whose duration is modeled in the PK routine. As an example, suppose the compartment being dosed is numbered 1 as above. The $PK abbreviated code must define a value for D1, the duration parameter for compartment 1, e.g., `D1=THETA(4)+ETA(4)`

RATE must be 0 for observation, other-type, and reset event records.

(See **dose, steady_state_dose**).
REFERENCES: Guide VI, section V.E
REFERENCES: Guide V, section 6.8

RATE PARAMETER

MEANING: Rate (R) parameter for PREDPP
CONTEXT: Additional PK Parameters

USAGE:

```
$PK
 R1= ....
```

DISCUSSION:

Rate parameters are used with PREDPP. They are optional additional PK parameters. With NM-TRAN, they are symbolized in the $PK block by reserved variables `Rn`, where n is the compartment number to which the parameter applies.

There is one rate parameter associated with every possible dose compartment of the kinetic model (the output compartment is not a possible dose compartment) and the rate parameter used for a given dose is that one associated with the compartment into which the dose is given (the dose compartment).

A rate-modeled zero-order bolus dose is actually an infusion, but one whose rate is given by the rate parameter computed by the PK routine.

Rate parameters are optional in the sense that rate parameters associated with compartments never receiving rate-modeled zero-order bolus doses or rate-modeled steady-state infusions may be ignored. However, if the RATE data item on some dose event record contains the value -1, then a rate parameter for the dose compartment must be computed in the PK routine.

Rate parameters act continuously. PREDPP obtains the value of a rate parameter, holding over the state-interval (t_1, t_2), from a call to PK with the record associated with t_2, even if the dose event time occurs before t_2. Therefore, if there are state times (e.g. t_2) falling within the time interval over which a zero-order bolus dose appears in the system, there exists the possibility that the rate of drug input can change during the interval. For this to occur, the rate parameter would need to be modeled in terms of time varying covariates. As a result, a zero-order bolus dose where the rate is modeled might be better described as a piecewise zero-order process.

When additional doses are specified on a dose event record the rate parameter applies to the dose and to all the additional doses.

(See **cmt**, **rate**, **pk**, **$pk**, **default_compartment**).
(See **exogenous supplementation example**).

REFERENCES: Guide IV, section V.C.5
REFERENCES: Guide V, section 12.2.3
REFERENCES: Guide VI, section III.F.4-5

RAW DATA ITEM

MEANING: Raw-data (RAW_) data item
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... RAW_ ...`

DISCUSSION:
`RAW_` labels NONMEM's raw-data (RAW) data item. The raw-data data item is optional. With a nonobservation record (i.e. MDV=1) it controls the definition of the value of the DV item associated with that record that appears in tables or scatterplots (whereas without the raw-data item, this DV item is the one in the original data set). Basically, 0 means the displayed DV item is that in the original data set, and 1 means the DV item is the raw-data-average of original DV items.

Values are:

0 The data record is not a template record.

1 The data record serves as a template record, aiding in the definition of the (particular) raw-data-average. The DV and RES items become the raw-data-average and the difference between this average and the PRED item, respectively. The average is over all observations in all other data records with user-defined data items matching (i.e. equal to) those occurring in the template record. (A user-defined data item is a data item not recognized by NONMEM, i.e., not one of ID, L2, DV, MDV, RAW_, MRG_, REPL_. It may be a PREDPP data item. Specific user-defined data item types may be excluded from the match; see the $OMIT record.) The average is a two-stage average; it is the average of within-individual averages.

   Any PRED-defined item stored in the SAVE region and displayed in a table or scatterplot is handled as is the DV item; it too is replaced by an average of the corresponding PRED-defined items obtained with observation records. Thus, several different raw-data-averages can be defined (using the same template).

With RAW_=1, MDV must also be 1 or 101. This use of MDV=1 does not prevent, as would ordinarily happen, the DV or RES items from being plotted.

A plot of DV, RES (and of any PRED-defined item stored in the SAVE region) will only include points from records with RAW_=1, unless one explicitly partitions on RAW_.

If PREDPP is used and NM-TRAN generates MDV, MDV is set to 1 if RAW_ is 1. If PREDPP is used, NM-TRAN sets RAW_ to 0 (if it is not already 0) when EVID is not equal to 0.

Warning: If PREDPP is used and NM-TRAN generates EVID, it sets EVID=2 for records with MDV=1 and no dosing information. This causes RAW_ to be set to 0. When RAW_ is used, the data set should include MDV and EVID data items. The values should be MDV=1 and EVID=0 when RAW_>0.

(See **data average example**, **data average block**).

REFERENCES: none.

RAW OUTPUT FILE

MEANING: NONMEM Raw Output file
CONTEXT: NONMEM Output Files

DISCUSSION:

With NONMEM 7, a NONMEM run produces several additional output files. These provide a more efficient way of way of extracting numerical results from the analysis than by obtaining them from the NONMEM report file.
(See **additional_output_file**).

Names of the files start with "root", where root is the root name (not including extension) of the NM-TRAN control stream file given at the nmfe7 command line, or root="nmbayes" if the control stream file name is not passed as an argument to the NONMEM executable. E.g., in this example the root is myexample.

```
nmfe7 myexample.ctl myexample.res
```

The raw output file ("root.ext") will be named myexample.ext

A different name for this file may be specified using the FILE=filename option of the $ESTIMATION record. This filename is used for all subsequent problems, until a new FILE=filename option is coded. The raw output file is only generated if the $ESTIMATION record is present and accumulates information from each Estimation step.

The first two lines of each file is the same. First is a header line that begins with the word TABLE, such as:

```
TABLE NO.      n:
```

These tables are not related to the ones produced by the $TABLE record. The value of n is incremented each time the Estimation Step is implemented, i.e., once per $ESTIM record, when MAXEVAL is not 0 and $EST is not omitted. The $EST records may be in the same problem, or in subsequent problems. If $COV is present in a problem without $EST (e.g., when $MSFI is used), then n is also incremented.

Next on the header line is the analysis text (same as given on the #METH: line in the report file), e.g., "First Order".

In the raw output file, the analysis text is followed by the goal function text (same as given on the #OBJT: line in the report file), e.g.,

```
Goal Function=MINIMUM VALUE OF OBJECTIVE FUNCTION
```

At the end of the header line is:

```
Problem=1 Subproblem=0 Superproblem1=0 Iteration1=0
          Superproblem2=0 Iteration2=0
```

The values of Problem, Subproblem, Superproblem1, Iteration1, Superproblem2, Iteration2 are the same values as would be found in the following variables in modules during the run:
IPROB IREP S1NUM S1IT S2NUM S2IT
(See **Problem_Iteration_Counters**).

The second line of the raw output file contains the column headers for the table, all on one line.

Options NOTITLE and NOLABEL and FORMAT of the $ESTIMATION record can be used to supress the first and/or second lines, and to modify the format of the values in the table. They apply to raw and additional output files, but only for the current Estimation

Step. The options must be re-specified with each $EST record.

Values of the Parameters and of the objective function are printed to the raw output file every PRINT iterations. The default order is:
ITERATION THETA, SIGMA, OMEGA, OBJ (Objective function). Values of Sigma and Omega are in the original parameterization, as in the file INTER. The option ORDER of the $ESTIMATION record may be used to change the order of THETA, SIGMA, and OMEGA.
(See **order_option**).

The iteration number, which is the first value in every line, is typically positive, but also may be negative under the following conditions:

1) The burn-in iterations of the MCMC Bayesian analysis are given negative values, starting at -NBURN, the number of burn-in iterations requested by the user. These are followed by positive iterations of the stationary phase.

2) The stochastic iterations of the SAEM analysis are given negative values. These are followed by positive iterations of the accumulation phase.

3) Iteration -1000000000 (negative one billion) identifies the line that contains the final result (thetas, omegas, and sigmas, and objective function) of the particular analysis. These are the value in the NONMEM output file that are labelled FINAL PARAMETER ESTIMATE. The order is theta, sigma, omega.

4) Iteration -1000000001 identifies the line that contains the standard errors of the final population parameters. These are the value in the NONMEM output file that are labelled STANDARD ERROR OF ESTIMATE. The order is theta, sigma, omega. This line is present if the Covariance Step is successful, or with METHOD=BAYES.

5) Iteration -1000000002 identifies the line that contains the eigenvalues of the correlation matrix of the variances of the final parameters.

6) Iteration -1000000003 identifies the line that contains the condition number , lowest, highest, Eigenvalues of the correlation matrix of the variances of the final parameters.

7) Iteration -1000000004 identifies the line that contains the OMEGA and SIGMA elements in standard deviation/correlation format

8) Iteration -1000000005 identifies the line that contains the standard errors to the OMEGA and SIGMA elements in standard deviation/correlation format

9) Iteration -1000000006 identifies the line that indicates 1 if parameter was fixed in estimation, 0 otherwise.

10) Iteration -1000000007 lists termination status (first item) followed by termination codes.

In nm73, termination status catalogs the error status:
For traditional analyses, an error number is listed. If negative, the analysis was user-interrupted
For EM/Bayes analysis, error numbers map as follows:
0,4: optimization was completed
1,5: optimization not completed (ran out of iterations)
2,6: optimization was not tested for convergence
3,7: optimization was not tested for convergence and was user interrupted
8,12: objective function is infinite or all individual objective fuctions are zero. problem ended

16,20: All individual objective fuctions are zero. problem ended
4,5,6,7,12,20: reduced stochastic/stationary portion was not completed prior to user interrupt

11) Iteration -1000000008 lists the partial derivative of the likelihood (-1/2OFV) with respect to each estimated parameter. This may be useful for using tests like the Lagrange multiplier test.

See the description of root.xml in Guide INTRODUCTION TO NONMEM 7 for versions 7.4 and higher.

A file simliar to the raw output file is root.cnv. This file contains convergence information for the Monte Carlo/EM methods, if CTYPE>0:

1) Iteration -2000000000 identifies the line that contains the mean of last CITER values (mean objective function is that of second to last CITER values)

2) Iteration -2000000001 identifies the line that contains the standard deviation of last CITER values (for objective function, STD of second to last CITER values)

3) Iteration -2000000002 identifies the line that contains the linear regression p-value of last CITER values against iteration number. (for objective function, p-value of second to last CITER values)

4) Iteration -2000000003 identifies the line that contains the Alpha used to assess statistical significance (p-value<alpha)

Please note the following: The Sigma values are in their Cholesky format, as this is the form in which convergence of these values are tested. The Alpha are those based on ones actually used for convergence test of that parameter, or which would have been used on that parameter if CTYPE were of proper type. The alpha may be bonferoni corrected because of multiple comparisons, depending on number of parameters that were tested or would have been tested. Objective function alphas are not bonferoni corrected.

For importance sampling and iterative two stage, the average objective function listed in root.cnv could be used as an alternative to the final objective function for likelihood ratio tests.

REFERENCES: Guide Introduction_7

RECORD COUNTERS: NIREC,NDREC

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

USAGE:

```
USE ROCM_INT, ONLY: NIREC=>NINDREC,NDREC=>NDATINDR
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: NINDREC,NDATINDR
```

DISCUSSION:

`NIREC`
The number of the individual record at the current call.

`NDREC`
The number of the data record within the individual record at the current call.

NIREC and NDREC also change value in conjunction with calls to PASS, a NONMEM utility routine.

NIREC and NDREC may be used as right-hand quantities in $PRED, $PK, and $ERROR blocks, and in a $INFN block in conjunction with PASS.

NM-TRAN automatically provides the necessary USE statement in the generated subroutines.

See also additional_record_counters for other variables of interest such as FIRSTREC.

Location prior to NONMEM 7: rocm32

REFERENCES: None.

RECURSIVE PRED INDICATOR

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPRD_INT, ONLY: I_REC=>IRECRSIV
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: IRECRSIV
```

DISCUSSION:

A PRED routine may be recursive, i.e., for single-subject data, with a given data record, the PRED computation depends on the values of variables that PRED computes with *any* of the previous records. PREDPP is such a routine.

By default, PRED is taken to be nonrecursive. At ICALL=0 or 1, PRED can declare itself to be recursive, by including the variable I_REC and setting it to 1.

The setting of I_REC can be changed from problem to problem, but regardless of the setting, with a given problem the setting is always taken to be 0 if the data for that problem are population data.

If I_REC is set to 1 and the repetition feature is not used, the SPECIAL option need not appear on the $COVARIANCE record; the special computation is automatically used.

PREDPP sets I_REC to 1 at ICALL=1.

I_REC can be set explicitly in abbreviated code. With recursive $PRED abbreviated code and with single-subject data, if I_REC is not set to a value in $PRED, then NM-TRAN causes I_REC to be set to 1.

Location prior to NONMEM 7: nmprd8

REFERENCES: None.

REPETITION VARIABLES

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPR_INT, ONLY: RPTI=>NRPT_IN,RPTO=>NRPT_OUT, &
                    RPTON=>NRPT_ON,PRDFL=>IUSEPRD
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: NRPT_IN,NRPT_OUT,NRPT_ON,IUSEPRD
```

DISCUSSION:

These variables provide the information controlling the Repetition feature of NONMEM.

`RPTO`

RPTO is the "repetition output value". With each data record, it may be set by PRED, and it conveys information to NONMEM. RPTO may be used as a left-hand quantity in $PRED, $PK and $ERROR blocks.

`RPTO=0`

If RPTO is set to 0, this produces no effect.

`RPTO=n` (where n is between 1 and 5)

The current record is thus marked as a "repetition base with value n", i.e. as the first of a series of contiguous records of the current individual record (with single-subject data, contiguous records of the data set) which will be repeated. See the repeat data item (See **RPT_**).

`RPTO=-n` (where n is between 1 and 5)

The current record thus becomes a "repetition initiator with value n", i.e. before the next record is passed to PRED all preceding records of the individual record (with single-subject data, all records of the data set), starting with the last such record marked as a repetition base with value n, and up to and including the current record, will be once again passed to PRED (will be "repeated"). The value n is put onto the top of a stack - the "repetition stack".

The series of records from base to initiator is called the "repetition series for n". The series may be repeated any number of times (see RPTON). After all repetitions are complete, the value n is removed from the repetition stack.

While the series is being repeated, RPTO may be set again with any record R of the series. If the (unsigned) repetition output value is a value already on the repetition stack, this value is ignored. Otherwise, the output value is not ignored, and it may be set so as to mark R as a repetition base or as a repetition initiator. In the latter case the current value on the top of the stack (m) is pushed down to the next lower position, the value n is put onto the top of the stack, and a new repetition series is initiated. After the new series has been fully repeated, the value n is removed from the stack, the value m is again put onto the top of the stack, and repetition of the series for m is continued with the record following R.

`RPTO=-1`

This value is a special value that may be used as the repetition output value with a given record. If a previous record has been marked as a repetition base

with value 1, then the record in question becomes a repetition initiator with value 1 in the usual way. But if no previous record has been marked as a repetition base, then it is assumed that the first record of the individual record (with single-subject data, the first record of the data set) is a repetition base with value 1. That is, before the next record is passed to PRED all preceding records of the individual record (with single-subject data, all records of the data set), starting with the first such record and up to and including the current record, will be once again passed to PRED (will be "repeated"). The value 1 is put onto the top of the repetition stack.

In addition, for the repetition feature to work, it must be enabled at the outset. This will be done if the repeat data item appears in the data set, or if RPTO is set to a nonzero value at ICALL=0 or ICALL=1.

`RPTON`

With each data record, RPTON may be set by PRED, and it conveys information to NONMEM. When the repetition output value is nonnegative, RPTON is ignored. Otherwise, RPTON may be set to an integer that gives the number of times the repetition series initiated by the data record is to be repeated. With the value 0, the series is repeated once. RPTON may be used as a left-hand quantity in $PRED, $PK and $ERROR blocks.

`RPTI`

RPTI is the "repetition input value". With each data record, it is set by NONMEM, and it conveys information to PRED. RPTI may be used as a right-hand quantity in $PRED, $PK, $ERROR, $DES, and $AES blocks.

`RPTI=0`

The record being passed is not "being repeated".

`RPTI!=0`

The record being passed to PRED is "being repeated". The nonzero value is the length of the repetition stack (see above).

By default, with each pass through an individual record (with single-subject data, with each pass through the data set), and with any data record that is being passed for the first time and is other than a repetition initiator, the output from PRED is used by NONMEM. If the record is a repetition initiator, NONMEM uses the output from PRED only when the repetition output value n has appeared on the repetition stack for the first time (as a result of RPTO being set to -n with this record) and when the record is being passed for the last time before the output value is subsequently removed from the stack. Otherwise, NONMEM ignores all output from PRED, except for the values set for these variables.

For example, in the case where a record is a repetition initiator, as is a subsequent record, where both records set RPTO to -1, and where both records set RPTON to 0, the first record is passed (at least) four times, and NONMEM uses the output from the first record when it is passed for the second time.

`PRDFL`

PRDFL is the "PRED output control flag". With each data record, it may be set by PRED, and it conveys information to NONMEM. PRDFL may be used as a left-hand quantity in $PRED, $PK and $ERROR blocks.

The PRED output control flag is used with very advanced applications with the repetition feature. With it, some of the default behaviour for when NONMEM pays

attention to PRED output (see above) - may be overridden.

`PRDFL!=1`
: This signals that the output from PRED with a passed record (except for the values of the repetition variables) is to be ignored by NONMEM.

`PRDFL=1`
: This signals that the output from PRED with a passed record is to be used by NONMEM.

With the PRED output control flag, PRED specifies when it is that NONMEM is to use PRED's output. However, just as with the default behavior, where during a pass through the data, NONMEM uses the output from a given data record once and once only, with each data record, PRDFL must be set to 1 once and once only, either when the record is passed initially or when it is repeated. Moreover, as with the default behavior, PRDFL can be set to 1 with a given record only after PRDFL has been set to 1 with the previous data record of the individual record (for single-subject data, with the previous data record of the data set).

In addition, for the PRED output control flag to work, it must be enabled at the outset, i.e., PRDFL must be set to 1 at ICALL=0 or ICALL=1. It may be enabled only when the repetition feature has also been enabled.

Location prior to NONMEM 7: nmpr10

(See **repeti1**, **repeti2**)

Help file repeti1 discusses the following files in the examples directory:

repeat1.ctl
repeat1s.ctl
repeat1t.ctl
repeatf.ctl

REFERENCES: None.

REPL DATA ITEM (NM75)

MEANING: Replication (REPL_) data item
CONTEXT: $INPUT record and NONMEM data set

USAGE:
```
$INPUT ... REPL_ ...
```

DISCUSSION:
`REPL_` labels NONMEM's replication (REPL_) data item. The replication data item is optional. When the REPL_ data item is listed in $INPUT, the NONMEM data set is considered to be a template data set. NONMEM itself replicates subjects from the template data set at the start of the problem to create an expanded NONMEM data set.

(Note that the NONMEM data set is typically the file FDATA generated by NM-TRAN, unless there is nothing for NM-TRAN to change and the format is supplied on the $DATA record, in which case the file named on the $DATA record is the NONMEM data set. See NONMEM Users Guide, Part IV.)

The value of REPL_ is the replication number for that subject. If the value is fractional, it will be truncated to the nearest integer. Only the REPL_ value of the first record of each individual will be used to determine its replication number. If for some subject the value of REPL_ is 0, the subject is deleted from the NONMEM data set.

REPL_ may be used with the $DATA ... REPL option, which replicates the entire data set REPL=n times. If both are used the REPL_ data item applies first, and the REPL option applies second.
See REPL option of $DATA record.
The REPL option and REPL_ data item are meant to be used with $SIMULATION or $DESIGN.
For important information, see Guide Introduction_7 Section $DATA REPL

REFERENCES: Guide Introduction_7

RPT DATA ITEM

MEANING: REPEAT (RPT_) data item
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... RPT_ ...`

DISCUSSION:
`RPT_` labels NONMEM's repeat data item. The repeat data item is optional. It is used as an alternative way to mark a data record as a repetition base. (For another way of doing this, global "Repetition Variables" RPTI,RPTO,RPTON,PRDFL may be used.)

Permissible values of RPT_ are:

0 The data record is not a repetition base.

n (n between 1 and 5)
The data record is marked as a "repetition base with value n", i.e. as the first of a series of contiguous records of the current individual record (with single-subject data, contiguous records of the data set) which may be repeated.
See repeatf.ctl in the examples directory.

(See **Repetition_Variables**).

REFERENCES: none.

SCALING PARAMETER

MEANING: Scaling (S) parameter for PREDPP
CONTEXT: Additional PK Parameters

USAGE:

```
$PK
 S2= ....
```

DISCUSSION:

Scaling parameters are used with PREDPP. They are optional additional PK parameters. With NM-TRAN, they are symbolized in the $PK block by reserved variables `Sn`, where n is the compartment number to which the parameter applies.

For each observation (or at any time other than an observation time for which a prediction is needed), PREDPP computes a prediction available to the ERROR routine. Associated with each observation is an observation compartment. The amount A in the observation compartment at the time of observation, divided by the value of a parameter S, is used as the prediction. The parameter S is called a scaling parameter. There is one such parameter associated with every compartment of the kinetic model (including the output compartment).

Scaling parameters are optional in the sense that scaling parameters associated with compartments with which predictions are not needed may be ignored. The values of scaling parameters that are not computed in PK are always understood to be 1.

A scaling parameter is computed by the PK routine using, if needed, information in the event record.

(See **cmt**, **pcmt**, **pk**, **$pk**, **default_compartment**).

With all versions of PREDPP and all ADVAN routines, observations of urine concentration Cu, as well as plasma concentrations Cp, may be included in the data. The default output compartment of all ADVANs may be used to accumulate urine data. This is discussed in Guide V, the Introductory Guide. See

Chapter 6, Section 9, The Output Compartment: Urine Collections and Observations
Chapter 7, Secton 4.3.3. Scaling by a Data Item

REFERENCES: Guide IV, section V.C.5
REFERENCES: Guide V, section 3.2.2 , 7.4.3
REFERENCES: Guide V, section 6.9
REFERENCES: Guide V, section 7.4.1
REFERENCES: Guide VI, section III.F.1 , V.H

SDE

MEANING: SDE subroutine
CONTEXT: NONMEM utility routine

Utility subroutine sde.f90 numerically evaluates SDE (stochastic differential equations).

USAGE:

```
$SUBROUTINE ADVAN6 TOL=10 DP OTHER=sde.f90
```

See sde9.ctl, sde9b.ctl and sde.ctl. They, along with the data files, can be found in the examples directory of the NONMEM 7 distribution medium.

See **INTRODUCTION TO NONMEM 7, Repeated Observation Records**
See **INTRODUCTION TO NONMEM 7, Stochastic Differential Equation Plug-In**

REFERENCES: Guide Introduction_7

SHRINKAGE

MEANING: NONMEM's estimate of inter-subject variance shrinkage
CONTEXT: NONMEM output

DISCUSSION:

Shrinkage data are included in the results between #TERM and #TERE.

With NONMEM 7.3, these are
ETAshrink
EBVshrink
EPSshrink

For details, see the NONMEM 7 guide.

With NONMEM 7.4, these are
ETAshrinkSD (called ETAshrink in NM73)
ETAshrinkVR
EBVshrinkSD (called EBVshrink in NM73)
EBVshrinkVR
EPSshrinkSD (called EPSshrink in NM73)
EPSshrinkVR

For details, see the NONMEM 7.4 guide

With NM73 and later, option ETASTYPE=1 in the $EST record, will average shrinkage information only among individuals that provided a non-zero derivative of their data likelihood with respect to that eta, and will not include subjects with a non-influential eta, that is in which the derivative of the data likelihood is zero.

Furthermore, you may specify eta i of particular subjects to be excluded or included using reserved variable ETASXI(i). (ETASXI stands for eta shrinkage exclude/include).
(See **etasxi**).

Shrinkage data are also given in file root.shk File root.shm is a shrinkage map describing which etas were included or excluded in the eta shrinkage assessment.

REFERENCES: Guide Introduction_7

SIGNAL

MEANING: SIGNAL program
CONTEXT: NONMEM run

Program SIGNAL is a NONMEM utility program. It allows the user to send a signal to NONMEM while it is running.

USAGE:

```
signal X
```

The following signals may be sent:

Print toggle: J,R, or P
Paraprint toggle: B,A, PA, or PP
Next: K or N
Stop: E or S
Subject Print toggle: T, U, or SU

DISCUSSION:

Sometimes NONMEM does not respond to user input via the ctrl key. This may occur during a parallel distribution run using MPI, or if the user began NONMEM with the -background switch. The user may open another console window, copy the program signal.exe (signal in UNIX) from the NONMEM util directory to your run directory, then enter any one of these commands:

Print toggle (monitor estimation progress):
signal J
signal R
signal P

Paraprint toggle (monitor parallel processing traffic):
signal B
signal A
signal PA
signal PP

Next (move on to next estimation mode or next estimation): signal K signal N

Stop (end the present run cleanly):
signal E
signal S

Subject print toggle:
signal T
signal U
signal SU

These create one of these files in the current (run) directory:
sig.print
sig.paraprint
sig.next
sig.stop
sig.subject

The multiple signals are synonyms. For example, ctrl-T is the keyboard switch, so signal t should match this. Also, since it is a subject toggle, signal s or signal su are synonyms,

suggesting Subject.

The signal command creates the signal files, and the nonmem program detects the signal file, and immediately deletes it.

A second argument is permitted to specify the run directory, in case you are not executing the signal program from the run directory.
signal N ~/nonmem/

REFERENCES: Guide Introduction_7

SIGNIFICANT DIGITS FROM EST. STEP

MEANING: NONMEM read-only global variables
CONTEXT: PRED routine

USAGE:

```
USE ROCM_REAL, ONLY NSIG=>SIGD,SIG=>DIFA
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: LPAR,DPSIZE
REAL(KIND=DPSIZE) :: SIGD,DIFA(LPAR)
```

DISCUSSION:

`SIG(i)`

When the minimization in the Estimation Step terminates due to rounding errors, but the number of significant digits that is achieved is reported, the ith value in SIG gives the number of significant digits for the ith UCP element.

`NSIG`

NSIG gives the minimum of the values found in the vector `SIG`

The NSIG and SIG values should only be used with ICALL = 3.

Location prior to NONMEM 7: rocm16

REFERENCES: None.

SIMEPS

MEANING: SIMEPS subroutine
CONTEXT: NONMEM utility routine

USAGE:

```
USE SIZES, ONLY: DPSIZE,LVR
REAL(KIND=DPSIZE):: EPS(LVR)
CALL SIMEPS (EPS)
```

DISCUSSION:
The NONMEM utility routine SIMEPS can be called by PRED during the simulation step, to obtain simulated epsilon values. It may be called only when ICALL=4.

Output argument:

`EPS`

An array into which SIMEPS stores simulated epsilon values EPS(1), EPS(2), ... The dimension of the array may be smaller than the maximum, e.g., it may equal the number of epsilons in the problem.

Simulated epsilon values arise from a multivariate normal pseudo-random distribution with mean 0 and variance-covariance as specified for SIGMA. With different calls to SIMEPS with different level-two records, new and different simulated epsilon values are obtained.
(A level-two record has the same value of L2 and may also be called an L2 record.)
(See **L2**).
(When the L2 data item is not defined, all data records are level-two records, and so different data records are different level-two records.) By default, with different calls to SIMEPS with the same level-two record, the same simulated epsilon values are obtained - those obtained at the first call with the record. (There is an advanced feature whereby records are "repeated"
(See **Repetition_Variables**),
and when records of a level-two record are being repeated, with different calls to SIMEPS with the same level-two record, the values obtained are the last values stored in these variables when the record was previously passed to PRED.)

If, though, the NEW option is used with the first random source on the $SIMULATION record, then each time SIMEPS is called (with the same or different level-two record), new and different values are obtained. With any particular call to SIMEPS, the effect of the NEW option can, though, be overridden;
(See **Simulation:_IETAOL_IEPSOL**)

So that simple simulation can be easily implemented with abbreviated code, values of epsilon are obtained by calls to SIMEPS occurring in the generated subroutine. When the data are population data and the Simulation Step is implemented, SIMEPS is called once with every call to PRED (or once at every call to ERROR if PREDPP is used). These calls are implemented so that even if, initially, the Simulation Step is not implemented, the NONMEM executable resulting from using an abbreviated code for PRED (or for ERROR if PREDPP is used) can be reused with a run implementing the Simulation Step.

Additional calls to SIMEPS may appear in simulation blocks of $PRED and $ERROR abbreviated code.
There is an analogous routine SIMETA. (See **simeta**).

REFERENCES: Guide IV, section III.B.13
REFERENCES: Guide V, section 12.4.8
REFERENCES: Guide VI, section IV.B.1 , IV.B.2 , Figure 10

SIMETA

MEANING: SIMETA subroutine
CONTEXT: NONMEM utility routine

USAGE:

```
USE SIZES, ONLY: DPSIZE,LVR
REAL(KIND=DPSIZE):: ETA(LVR)
CALL SIMETA (ETA)
```

DISCUSSION:
The NONMEM utility routine SIMETA can be called by PRED during the simulation step, to obtain simulated eta values. It may be called only when ICALL=4.

Output argument:

`ETA`

An array into which SIMETA stores simulated eta values ETA(1), ETA(2), ... The dimension of the array may be smaller than the maximum, e.g., it may equal the number of etas in the problem.

Simulated eta values arise from a multivariate normal pseudo-random distribution with mean 0 and variance-covariance as specified for OMEGA. With different calls to SIMETA with different individual records, new and different simulated eta values are obtained.
(With population data, an individual record has the same value of ID and may also be called a level-one "L1" record.) (See **ID**).
By default, with different calls to SIMETA with the same individual record, the same simulated eta values are obtained - those obtained at the first call with the record. (There is an advanced feature whereby records are "repeated"
(See **Repetition_Variables**),
and when records of an individual record are being repeated, with different calls to SIMETA with the same individual record, the values obtained are the last values stored in these variables when the record was previously passed to PRED.)

If, though, the NEW option is used with the first random source on the $SIMULATION record, then each time SIMETA is called (with the same or different individual record), new and different values are obtained. Thus, for example, when PRED is called with the first data record of an individual record, PRED can in turn call SIMETA multiple times until values are obtained such that, for example, ETA(2) is not larger than 5 in absolute value; that is, values can be obtained from a truncated distribution (see below). With any particular call to SIMETA, the effect of the NEW option can, though, be overridden;
(See **Simulation:_IETAOL_IEPSOL**)
So that simple simulation can be easily implemented with abbreviated code, values of etas are obtained by calls to SIMETA occurring in the generated subroutine. When the data are population data, SIMETA is called once per individual record by PRED (or PK if PREDPP is used). (In the case of PK, the array of etas is stored in a common and is available to the ERROR subroutine as well.) When the data are single-subject data, SIMETA is called once at every call to PRED (once at every call to ERROR if PREDPP is used).

These calls are implemented so that even if, initially, the Simulation Step is not implemented, the NONMEM executable resulting from using an abbreviated code for PRED (for PK or ERROR if PREDPP is used) can be reused with a run implementing the

Simulation Step.

EXAMPLES OF USAGE:

In this example, the value of ETA(2) used by PRED will be less than 5 in absolute value. For this code to have the desired effect, the option NEW must be used in the $SIMULATION record.

```
IF (ICALL.EQ.4.AND.NEWIND.NE.2) THEN
  DO WHILE (ABS(ETA(2)).GT.5)
  CALL SIMETA (ETA)
  ENDDO
ENDIF
```

Suppose there are two etas to be selected in this manner. Each one needs its own CALL SIMETA loop, because each CALL SIMETA replaces all the etas.

```
IF (ICALL.EQ.4.AND.NEWIND.NE.2) THEN
  DO WHILE (ABS(ETA(2)).GT.5)
  CALL SIMETA (ETA)
  ENDDO
  ETA2=ETA(2)
  DO WHILE (ABS(ETA(1)).GT.0.52)
  CALL SIMETA (ETA)
  ENDDO
  ETA1=ETA(1)
ENDIF
```

Another way this can be implemented is as follows:

```
 IF (ICALL.EQ.4.AND.NEWIND.NE.2) THEN
   DO WHILE(ABS(ETA(2)).GT.5.OR.ABS(ETA(1)).GT.0.52)
   CALL SIMETA (ETA)
   ENDDO
   ETA2=ETA(2)
   ETA1=ETA(1)
 ENDIF
```

There is an analogous routine SIMEPS. (See **simeps**).

REFERENCES: Guide IV, section III.B.13 , IV.I
REFERENCES: Guide V, section 12.4.8
REFERENCES: Guide VI, section III.E.2 , IV.B.2

SIMULATION BLOCK

MEANING: Abbreviated code for simulation
CONTEXT: Abbreviated code

SAMPLE:

```
$PK
 IF (ICALL.EQ.4) CL=THETA(1)+ETA(1)
```

DISCUSSION:
A "simulation block" is a block of abbreviated code that is only executed when ICALL=4 (during simulation). Such blocks may be present in $PK, $ERROR, and $PRED, and may be implemented by means of generated FORTRAN subroutines. E.g.,

```
IF (ICALL.EQ.4) THEN
 ... simulation block ...
ENDIF
```

Special rules apply to such blocks.

1) No eta derivatives are computed in a simulation block.

2) Transgeneration is permitted. NM-TRAN allows a data item label to appear on the left of an assignment statement. NM-TRAN generates assignment statements changing first the data item in the event or data record, and then the local variable having that label. E.g., suppose WT is listed in $INPUT:

   ```
   IF (ICALL.EQ.4) WT=70+70*ETA(3)
   ```

   The generated code is:

   ```
   IF(ICALL.EQ.4.D0)THEN
   EVTREC(NVNT,8 )=70.D0+70.D0*ETA(03)
   WT=EVTREC(NVNT,08)
   ENDIF
   ```

   NONMEM and PREDPP reserved data items should not be modified during simulation. Transgeneration is permitted with simulation with subproblems. With all versions of NONMEM, the data set for each subproblem after the first is the same data set used by the previous subproblem, and includes any changes (transgeneration) made by the previous subproblem.

3) Calls to certain NONMEM routines are permitted:

   ```
   CALL SIMETA(ETA)
   CALL SIMEPS(EPS)
   CALL RANDOM(n,R)
   ```

   where n is an integer 1-10. If `CALL RANDOM` is present, R becomes a reserved variable used for the random number.

   Note that NM-TRAN provides the necessary calls to SIMETA and SIMEPS in generated routines. Explicit calls are used in abbreviated code only to obtain different values of `ETA` and `EPS`.

4) A RETURN statement may be used. If in $ERROR or $PRED, and the RETURN occurs in a simulation block, then Y may be assigned a value prior to the return. If so, then F is set (F=Y); otherwise F is not set.

5) Loops are permitted. The syntax is as follows.

```
DO WHILE (condition)
 .. statements ..
END DO
```

Here are some examples. For a truncated normal distribution that only requires testing the eta value directly, this code can be used:

```
$PK
 ...
DO WHILE (ETA(1).GT.5)
CALL SIMETA(ETA)
ENDDO
```

Another example:

```
IF (ICALL.EQ.4.AND.NEWIND.NE.2) THEN
  DO WHILE (ETA(1).GT..5.OR.ETA(1).LT.-.5)
    CALL SIMETA(ETA)
  ENDDO
ENDIF
IF (ICALL.EQ.4) WT=70+70*ETA(1)
```

(With these two examples, the first random seed of the $SIMULATION record must have the NEW option. Note also that, because of the previous automatic call to SIMETA, ETA(1) requires no initialization, but that R in the next example does.)

```
IF (ICALL.EQ.4.AND.NEWIND.NE.2) THEN
  R=1
  DO WHILE (R.GT..5.OR.R.LT.-.5)
    CALL RANDOM(2,R)
  ENDDO
ENDIF
IF (ICALL.EQ.4) WT=70+70*R
```

This example illustrates how a categorical variable with equal-likely probabilities can be generated from a random number R, uniformly distributed between 0 and 1. In this example, the categorical variable BIN takes values 1 through 5.

```
IF (ICALL.EQ.4) THEN
  CALL RANDOM(2,R)
  BIN=INT(R*5)+1
ENDIF
```

The number 5 can be replaced with any other positive integer n to obtain an n-valued categorical variable. Here INT is the function that transforms a nonnegative number x into the greatest integer not exceeding x. The effect of this simulation code is to perform the transformation:

BIN=1 if R < .2
BIN=2 if R < .4 and R >= .2
BIN=3 if R < .6 and R >= .4
BIN=4 if R < .8 and R >= .6
BIN=5 if R < 1 and R >= .8

6) The "EXIT n k" statement may be used. The value of n may be 0, 1 or 2. The value of k is referred to as the PRED EXIT CODE.
If it is desired that the simulation be immediately terminated, then use an EXIT 2 code:

```
IF(ICALL==4.and.IPRED<0.1 .and. TIME>20.0) EXIT 2
```

With versions of NONMEM prior to 7.2, the "EXIT 1" statement in the Simulation step also caused NONMEM to abort. As of NONMEM 7.2, if an error occurs in PREDPP during simulation such as

```
PK PARAMETER FOR KA IS NON-POSITIVE
```

or a user-implemented EXIT 1 is issued during simulation, then PRED will be called with a new ETA and EPS. This feature is referred to as Simulation Error Forgiveness. NONMEM describes this as PRED SIMULATION REDO in the NONMEM report file. It writes to the NONMEM report file a description of the data record and THETA and ETA values, for example

```
 PRED SIMULATION REDO PRED EXIT CODE = 1 INDIVIDUAL NO.        1
 ID= 1.00000000000000E+00   (WITHIN-INDIVIDUAL) DATA REC NO.    1
 THETA=
  3.00E+00   8.00E-02   4.00E-02
 ETA=
  4.66E-01   2.91E-03   9.95E-01
 MESSAGE ISSUED FROM SIMULATION STEP
```

If ten such errors occur in the same subject, then it is supposed that the cause of the simulation error is not due to an occasional bad random sample, but is caused by a systematic error in the control stream file. The simulation step is terminated with the message

```
PRED ERROR OCCURRED TOO OFTEN ON SIMULATION
```

instead of a message

```
SIMULATION STEP PERFORMED
```

With NONMEM 7.5, the PRED EXIT CODE k may be in the range 1000-9999. For example,

```
IF(ICALL==4.and.IPRED<0.01 .and. TIME>20.0) EXIT 1 2300
```

This can only occur with user's EXIT code; PREDPP will not generate this kind of EXIT. NONMEM will try PRED SIMULATION REDO up to 10000 times. The message "PRED SIMULATION REDO" itself is written to PRDERR up to 30 times. After that, the following message is written to PRDERR:

```
 SUBSEQUENT PRED SIMULATION REDO ERROR MESSSAGES SUPPRESSED
```

NONMEM continues trying new ETA and EPS. Be careful that the condition does not occur too often (causing wasteful computation). After 10000 tries, the simulation is terminated as a protection against an infinite loop. The following message is written to PRDERR:

```
 TOO MANY CONSECUTIVE PRED ERRORS (>10000) OCCURRED ON SIMULATION
```

(See **PRED Exit Code**).
(See **abbreviated**).
(See **INTRODUCTION TO NONMEM 7, Simulation Error Forgiveness (NM72)**).
(See **INTRODUCTION TO NONMEM 7, Extensions to Simulation Error Forgiveness (NM75)**).

REFERENCES: Guide IV, section III.B.13 , IV.I
REFERENCES: Guide V, section 12.4.8
REFERENCES: Guide VI, section III.E.2 , IV.B.2

SIMULATION: ETA,EPS

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPRD_REAL, ONLY: ETA,EPS
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: LVR,DPSIZE
REAL(KIND=DPSIZE) :: ETA(LVR),EPS(LVR)
```

DISCUSSION:

When the ONLYSIMULATION option on the $SIMULATION record is used, the displayed eta values are those that PRED stores in ETA during the Simulation Step (i.e., at ICALL=4).

In generated subroutines, eta values are automatically stored in ETA, and thus automatically, the simulated etas are displayable.

When using the PRED repetition feature, epsilon values generated in PRED for simulation purposes (with or without the use of the ONLYSIMULATION option) should be stored in EPS. Then when records are repeated, these same epsilons values will be available in EPS as input. In particular, with every repeated record, the values that were stored in EPS the last time the record was passed, are made available as input to PRED.

In generated subroutines, epsilon values are automatically stored in EPS, and thus automatically, the simulated epsilons are available as input with repeated records.

When using the repetition feature with single-subject data, eta values generated in PRED for simulation purposes should be stored in ETA. What happens in this case is analogous to what happens in the case of population data with epsilon values.

In generated subroutines, eta values are automatically stored in ETA, and thus automatically, the simulated etas are available as input with repeated records.

Location prior to NONMEM 7: nmprd7

REFERENCES: None.

SIMULATION: IETAOL IEPSOL

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine
USAGE:

```
USE NMPR_INT, ONLY: IETAOL,IEPSOL
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: IETAOL,IEPSOL
```

DISCUSSION:

Values are stored by PRED for use with SIMETA and SIMEPS. They allow the effect of the NEW option on the $SIMULATION record to be overridden, which may be useful when the PRED repetition feature is used. They also allow calls to SIMETA and SIMEPS to be ignored. This may be useful when e.g. a simulation was undertaken with user code where SIMEPS is called with every data record (as happens automatically with NM-TRAN generated codes), and the exact same simulation is now to be repeated, but with a data set obtained from the earlier one by the addition of new nonobservation records (with which SIMEPS output is not needed). If calls to SIMEPS with the new records are not ignored, SIMEPS output will be generated with all the records, and in particular, SIMEPS output will be generated with the original records, which will differ from what it was earlier, thus resulting in the simulation of a different set of observations.

`IETAOL`

IETAOL=-1: Next call to SIMETA will be ignored

IETAOL=0: Next call to SIMETA will behave as usual.

IETAOL=1: Next call to SIMETA will behave as though NEW had not been specified. If SIMETA has been called previously with the individual record, SIMETA will produce the previous eta values.

`IEPSOL`

IEPSOL=-1: Next call to SIMEPS will be ignored.

IEPSOL=0: Next call to SIMEPS will behave as usual.

IEPSOL=1: Next call to SIMEPS will behave as though NEW had not been specified. If SIMEPS has been called previously with the level-two record, SIMEPS will produce the previous epsilon values.

Values must be stored before PRED calls SIMETA (SIMEPS). NM-TRAN generated or Library code has a call to SIMETA (SIMEPS) in its second section (see Guide IV). If this call is to be affected, values must be stored using verbatim code in the FIRST block.

Location prior to NONMEM 7: nmpr11.for

REFERENCES: None.

SIMULATION: NREP,IREP

MEANING: NONMEM read-only global variables

CONTEXT: User-supplied routines

USAGE:

```
USE ROCM_INT, ONLY: NREP,IREP=>NCREP
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: NREP,NCREP
```

DISCUSSION:

`NREP`

The number of replications in the Simulation Step, given by the NSUBS option of the $SIMULATION record.

`IREP`

The number of the current replication.

These variables may be used as right-hand quantities in abbreviated code for initialization/finalization blocks.

Location prior to NONMEM 7: rocm10

REFERENCES: None.

SIMULATION: SIMEPS ERROR CODE

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine
USAGE:

```
USE NMPRD_INT, ONLY: IERSQ
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: IERSQ
```

DISCUSSION:

Value is stored by SIMEPS for use with PRED and ERROR. Is relevant when correlations are stored in CORRL2. Is set to 0 before SIMEPS is called.
(See **Correlation_Across_L2_Records**).

`IERSQ`

The SIMEPS error return code.
IERSQ=0: Normal return
IERSQ=1: Correlation matrix obtained using the correlations stored in CORRL2 is not positive definite.

PRED may attempt corrective action; if it does so successfully, it should reset IERSQ to 0.

(With versions of NONMEM through 7.2, C was used rather than CORRL2.)

With NONMEM 7.3 (and all earlier versions), the value of IERSQ is not checked in generated code.

Location prior to NONMEM 7: nmprd6

REFERENCES: None.

SIZE OF INDIVIDUAL RECORD

MEANING: NONMEM read-only globoal variable
CONTEXT: PRED routine

USAGE:

```
USE ROCM_INT, ONLY: LIREC=>NDATPASS
```

GLOBAL DECLARATION:

```
INTEGER(KIND=ISIZE) :: NDATPASS
```

DISCUSSION:
LIREC is The number of data records contained in the individual record at the current call to PRED.

LIREC also changes value appropriately in conjunction with calls to PASS.

LIREC may be used as a right-hand quantity in $PRED, $PK, and $ERROR blocks, and in a $INFN block in conjunction with PASS.

Location prior to NONMEM 7: rocm48

REFERENCES: None.

SIZES FSIZES PRSIZES

MEANING: MODULE for NONMEM and its components.
CONTEXT: Source code

DISCUSSION:
File SIZES is supplied on the NONMEM distribution media and is copied to the resources directory. It contains FORTRAN PARAMETER statements giving the default values of symbolic parameters used in source code. Constants from SIZES may be incorporated into the source code by means of the FORTRAN USE statement. Some of these constants describe array sizes.

With NONMEM 7.2 and higher, there are several changes vs. previous releases.

There is only one version of SIZES (i.e., there is no longer a SIZES_reg or SIZES_big).

Many NONMEM and NM-TRAN arrays are allocated dynamically at run time.

NM-TRAN creates a subroutine FSIZESR for NONMEM. FSIZESR contains values for some parameters (e.g., LVR) that are exactly what is needed for the current problem, and contains 0's for parameters that NM-TRAN cannot assess. Default values for parameters that are 0 in FSIZESR are obtained by NONMEM from the file SIZES.f90.

Note that file FSIZES is a convenient reference for users to view, but is not used. SUBROUTINE FSIZESR in FSUBS is what is actually used during the NONMEM run.

NM-TRAN creates a file prsizes.f90 for PREDPP. PREDPP arrays are allocated statically but may be re-compiled at run time using parameters defined in prsizes.f90. Some constants in prsizes are assessed for the current problem; others are copied from default values in SIZES.f90.

The user may override many of the parameter values in FSIZES and prsizes with the $SIZES record. Any value specified by $SIZES will override both the default in SIZES.f90 and the value that NM-TRAN would have specified.
(See **$sizes**)
As of NONMEM 7.3, NMTRAN determines the maximum number of observation records (MDV=0) that occur in any subject, among all data files used in the entire control stream file. If this value is greater than the NO value listed in SIZES.f90, it will set NO to this larger size. Thus, users no longer have to be conscientious of sizing the NO parameter. However, there is no guarantee that NMTRAN will correctly assess NO for the entire scope of the control stream file for all types of problems. Should this occur, NONMEM may issue an error, and the user will need to set the NO value with a $SIZES record.

Constants that may be changed with $SIZES record:

LTH LVR LVR2 LPAR MMX MAXIDS NO PD MAXOMEG MAXPTHETA

Consants that are also in prsizes:

PC PCT PIR PD PAL MAXFCN

Constants that may not be changed using $SIZES record

LSTEXT MAXXNAME MAXVRESWRES MAXVRESN MXNAME DIMQ PPR PW SD SCO SDF MAX_EXTRA

The following is the descriptive comment and value of each parameter in the version of SIZES that is supplied on the NONMEM 7 distribution medium. More information including descriptions of the buffers may be found in the INTRODUCTION TO NONMEM 7.

NLUSER=100
: Maximum number of logical I/O units user may use.

LTH=100
: MAX. NO. OF THETA'S. Dynamically sized, or set by user via $SIZES record

LVR=30
: MAX. NO. OF ETA'S + EPS'S. Dynamically sized, or set by user via $SIZES record

LVR2=20
: MAX. NO. OF ETA'S PERMITTED WHEN LAPLACIAN METHOD IS USED. Value may be over-ridden by user via $SIZES record Value may be over-ridden by user via $SIZES record

NO=250
: MAX NO. OF OBSERVATION RECORDS / INDIVIDUAL RECORD

MMX=10
: MAX NO. OF MIXTURE SUBPOPULATIONS. Dynamically sized, or set by user via $SIZES record

LNP4=4000
: SIZE OF COMMON NMPRD4. Value may be over-ridden by user via $SIZES record

LSUPP=4050
: MAX. NO. OF POINTS OF SUPPORT WITH NONPARAMETRIC ESTIMATE. Value may be over-ridden by user via $SIZES record

LIM7=2
: SIZE OF BUFFER 7. DO NOT GO LOWER THAN 2. Value may be over-ridden by user via $SIZES record

LWS3=9000
: SIZE OF WORKING SPACE 3 AT LEAST AS LARGE AS: NS*NOETAS**2, WHERE NS IS THE NO. OF DIRECTIONS USED WITH THE STIELTJES METHOD, AND NOETAS IS THE NUMBER OF ETA'S. Value may be over-ridden by user via $SIZES record

MAXIDS=10000
: MAX. NO. OF INDIVIDUALS IN DATA SET. Dynamically sized, or set by user via $SIZES record

LIM1=10000
: SIZE OF BUFFER 1. Value may be over-ridden by user via $SIZES record

LIM2=100000
: SIZE OF BUFFER 2. Value may be over-ridden by user via $SIZES record

LIM3=10000
: SIZE OF BUFFER 3. DO NOT GO LOWER THAN 2. Value may be over-ridden by user via $SIZES record

LIM4=1000
: SIZE OF BUFFER 4: LIM4=NUMBER OF SUBJECTS. Value may be over-ridden

by user via $SIZES record

LIM5=200
SIZE OF BUFFER 5. Value may be over-ridden by user via $SIZES record

LIM6=400
SIZE OF BUFFER 6. Value may be over-ridden by user via $SIZES record

LIM8=200
SIZE OF BUFFER 8. Value may be over-ridden by user via $SIZES record

LIM10=100000
SIZE OF BUFFER 10. Value may be over-ridden by user via $SIZES record

LIM11=25
SIZE OF BUFFER 11. Value may be over-ridden by user via $SIZES record

LIM13=1000
SIZE OF BUFFER 13: LIM13=NUMBER OF SUBJECTS. Value may be over-ridden by user via $SIZES record

LIM15=1000
SIZE OF BUFFER 15: LIM15=NUMBER OF SUBJECTS. Value may be over-ridden by user via $SIZES record

LIM16=400
SIZE OF BUFFER 16. Value may be over-ridden by user via $SIZES record

LSTEXT=70000
AT LEAST (MAXOMEG*(MAXOMEG+1)/2+MAXPHETA+5)*NUMTEXT. LSTEXT is maximun number of characters to a single line of the raw output file specified by $EST FILE=. where NUMTEXT is number of characters needed to represent a number and its delimiter. For example, FORMAT=,1PE12.5 takes up NUMTEXT=13 characters.

LSFORM=2048
LSFORM is the character length of FORMAT TFORMATL,RFORMATL, pertaining to the full length of LFORMAT, RFORMAT for the $TABLE record.

MAXRECID=200
Maximum number of records in any individual record. For use by PREDPP. Value may be over-ridden by user via $SIZES record

PC=30
MAX. NO. OF COMPARTMENTS MAXIMUM IS 99. Value may be over-ridden by user via $SIZES record

PCT=30
MAX. NO. OF MODEL EVENT TIMES. Value may be over-ridden by user via $SIZES record

MAXNRDS=PC
MAXNRDS: MAX. NO. OF DELAY COMPARTMENTS FOR ADVAN16, ADVAN17, and ADVAN18. YOU MAY WANT TO MAKE IT LOWER THAN PC AND SAVE MEMORY Value may be over-ridden by user via $SIZES record

PAST_SIZE=4000
RESOLUTION (NUMBER OF DETAILED POINTS) FOR DELAY COMPARTMENTS STORAGE FOR ADVAN16 AND ADVAN17. Value may be over-ridden by user via $SIZES record

PCT_BIG=10000

MAX. NO. OF MODEL EVENT TIMES THAT NMTRAN CAN PROCESS.^M ! PG: SIZE OF GG; MAX. NO. OF BASIC+ADDITIONAL PK PARAMS.^M ! (MAXIMUM IS PCT+99)^M

PIR=700

SIZE OF COMPACT DA/DP/DT ARRAYS. Value may be over-ridden by user via $SIZES record

PD=50

CHANGED TO INCREASE DATA ITEMS PER DATA RECORD FROM 20 TO 50 IT IS ALSO SIZE OF VDATREC DATA ARRAY. THIS IS DONE TO SEPARATE VDATREC AND VRESWRES VARIABLES/ Dynamically sized, or set by user via $SIZES record.

MAXIC=90

MAXIC: MAXIMUM NUMBER OF ACTIVE INFUSIONS FOR PREDPP. Value may be over-ridden by user via $SIZES record

PDT=500

MAXIMUM NUMBER OF TABLE ITEMS/PRED-DEFINED ITEMS. Value may be over-ridden by user via $SIZES record

PAL=50

NO. OF ADDITIONAL AND LAGGED DOSES. Value may be over-ridden by user via $SIZES record

MAXFCN=1000000

MAX. NO. OF CALLS IN GENERAL NON-LINEAR MODELS IMAX IN MODULE PRCOM_INT OVERRIDES. Value may be over-ridden by user via $SIZES record

STIELTJ_SIZE=101

The next four are for internal use, pertaining to various additional weighted residual diagnostics

MAXXNAME=40

MAXVRESWRES=39

MAXVRESN=9

MXNAME=40

DIMTMP=500

RELATED TO THE NUMBER OF USER-DEFINED VARIABLES. Value may be over-ridden by user via $SIZES record

DIMCNS=500

RELATED TO THE TOTAL NUMBER OF CONSTANTS. Value may be over-ridden by user via $SIZES record

DIMNEW=1000

RELATED TO THE TOTAL NUMBER OF INTERMEDIATE VARIABLES. Value may be over-ridden by user via $SIZES record

DIMQ=99999

ARRAY SIZE FOR LOGICAL CONDITIONS

FL=49

LOGICAL UNIT NUMBER FOR FLIB

DIMVRB=1000
MAX. NO. OF LINES OF VERBATIM CODE

PL=10
MAXIMUM DEPTH OF NESTED IF STATEMENTS

NFUNCX=100
MAXIMUM NUMBER OF USER FUNCTIONS.

NVECX=100
MAXIMUM NUMBER OF USER VECTORS.

MAXOTHER=1000
Maximum number of filenames listed on $SUBROUTINE OTHER=filename

SD=30
LENGTH OF DATA LABEL

FSD=67000
LENGTH OF CONTROL STREAM FILE STRING (ALSO, MULTIPLE LINES CONCATENATED WITH & MAY NOT EXCEED FSD)

FSD1=67001
LENGTH OF CONTROL STREAM FILE STRING

SCO=30
STRING LENGTH OF NUMBER IN $THETA, $OMEGA, $SGIGMA RECORDS.

SDF=24
LENGTH OF DATA ITEM

NFSIZES=50
NUMBER OF ITEMS LISTED IN FSIZES FILE

NPRSIZES=14
NUMBER OF PREDPP ITEMS IN PRSIZES.F90

CONSTANTS FOR Monte Carlo, EM methods

MAX_EXTRA=20
Numer of $EST statements allowed per problem

NPOPMIXMAX=10
Now dynamically sized to MAXIMUM NUMBER OF SUB-POPULATIONS FOR MIXTURE MODELS MMX. Or, set by user via $SIZES record

MAXOMEG=70
Now dynamically sizes to LVR, OR MAXIMUM OMEGA DIMENSION ETAS. Or, set by user via $SIZES record (but should be left alone)

MAXPTHETA=90
Now dynamically sized to LTH, MAXIMUM NUMBER OF THETA, PLUS LOWER TRIANGLE OF SIGMA. THUS, IF NUMBER OF THETAS IS N, AND DIMENSION OF SIGMAS IS M, THEN NEED MAXPTHETA=N + M*(M+1)/2. OR, may be set by user via $SIZES record (but should be left alone).

MAXITER=210
Maximum number of previous iterations to incorporate into Monte Carlo convergence tests Effective CITER is <=MAXITER

ISAMPLEMAX=10
FOR SAEM METHOD, EFFECTIVE ISAMPLE<=ISAMPLEMAX

MAXSIDS=100
MAXIMUM NUMER OF SUPER ID ITEMS IN DATA SET

MAXSIDL=0
MAXIMUM LEVEL OF SUPER IDS. MAXSIDL=0 MEANS NO SUPER IDS. Default 0, until $LEVEL is used, in which case it will be dynamically sized.

MAXFTEXT=100
Maximum of PRDERR message lines

PNM_MAXNODES=100
MAXIMUM NUMBER OF PARALLELIZATION NODES

PNM_BUFFER_SIZE=100000
INTERNAL BUFFER SIZE, FOR EFFICIENT PACKAGING AND SENDING BETWEEN PROCESSES

REFERENCES: Guide III, section V.3.0
REFERENCES: Guide Introduction_7

SKIP

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:
```
USE NMPR_INT, ONLY SKIP_=>SKIP
```

GLOBAL DECLARATION:
```
INTEGER(KIND=ISIZE) :: SKIP
```

DISCUSSION:

The SKIP_ variable controls premature termination of a problem (with subproblems), superproblem or superproblem iteration. It may be set as follows during (sub)problem-finalization.

`SKIP_=1`
: With a subproblem: terminate the entire problem and proceed to the next problem, if this exists.

`SKIP_=2`
: With a problem during an iteration of a second-level superproblem A nested within a first-level superproblem B: terminate the iteration and proceed to the next iteration, if this exists, and if this does not exist, to the following (second-level super)problem of B, if this exists, and if this does not exist, to the next iteration of B.

`SKIP_=3`
: With a problem during an iteration of a second-level superproblem A nested within a first-level superproblem B: terminate the entire superproblem A and proceed to the following (second-level super)problem of B, if this exists, and if this does not exist, to the next iteration of B.

`SKIP_=4`
: With a problem during an iteration of a first-level superproblem: terminate the iteration and proceed to the next iteration, if this exists, and if this does not exist, to the following (first-level super)problem, if this exists.

`SKIP_=5`
: With a problem during an iteration of a first-level superproblem: terminate the entire superproblem and proceed to the following (first-level super)problem, if this exists.

With NM-TRAN, in a finalization block of abbreviated code one may set `SKIP_` and/or use the following phrases:

```
END PROBLEM                                (same as SKIP_=1)
END SECOND-LEVEL SUPERPROBLEM              (same as SKIP_=3)
END FIRST-LEVEL SUPERPROBLEM               (same as SKIP_=5)
END SECOND-LEVEL SUPERPROBLEM ITERATION    (same as SKIP_=2)
END FIRST-LEVEL SUPERPROBLEM ITERATION     (same as SKIP_=4)
```

E.g. The following are all equivalent:

A.

```
SKIP_=2
```

B.

```
IF (...) THEN
   END SECOND-LEVEL SUPERPROBLEM ITERATION
ENDIF
```

C.

```
IF (...) THEN
   ENDSECONDLEVELSUPERPROBLEMITERATION
ENDIF
```

Location prior to NONMEM 7: nmpr15

REFERENCES: None.

SPTWO

MEANING: SPTWO subroutine
CONTEXT: User-supplied subroutine; replaces a NONMEM dummy routine

This feature is not fully documented. The interested user may be able to obtain more information by studying the appropriate sections of NONMEM code and previous examples that may be available from advanced users.

USAGE:

```
SUBROUTINE SPTWO (ICALL,I1,I2,NROB,D,IER)
USE SIZES, ONLY: ISIZE,DPSIZE,NO
INTEGER(KIND=ISIZE), INTENT(IN)     :: NROB,ICALL
INTEGER(KIND=ISIZE), INTENT(IN OUT) :: I1,I2,IER
REAL(KIND=DPSIZE),   INTENT(IN OUT) :: D(NO,2)
```

DISCUSSION:
SPTWO is used to redefine the meaning of the RES and WRES items for observation records within an individual record. It is called with each individual record. The labels 'RES' and 'WRES' can be changed, as usual.

Input argument:

`ICALL`
: Similar to ICALL for PRED subroutine.
Possible values: 0, 1, 2

`NROB`
: Number of observation records in the individual record.

Output argument:

`I1,I2`
: When SPTWO is used, by default, `I1` and `I2` are 0, meaning zero lines are not to be generated through RES or WRES values on scatterplots. If zero lines through RES or WRES values on scatterplots are desired, SPTWO should set `I1` or `I2`, respectively, to 1.

`D(J,1)`
: Value of RES for Jth observation record, J=1,...,NROB.

`D(J,2)`
: Value of WRES for Jth observation record, J=1,...,NROB

`IER`
: Error indicator.
0 - Normal return.
non-zero - NONMEM should stop.

Other Inputs:

NONMEM read-only global variables.
(See **MIX:_DATA**)
(See **CONTR:_III,DIM**)
(See **CONTR:_F,G,H**)
(See **Non-active_ETA_list_for_PRED**)
(See **Parameters_OMEGA_SIGMA:_Current**)
(See **Record_Counters:_NIREC,NDREC**)

REFERENCES: None.

SS DATA ITEM

MEANING: Steady-State (SS) data item for PREDPP
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... SS ...`

DISCUSSION:
`SS` labels PREDPP's steady-state (SS) data item. The steady-state data item is optional. It can take one of three values in any event record.

0 indicates that the dose is not a steady state dose.

1 indicates that the dose is a steady state dose, and that the compartment amounts are to be reset to the steady-state amounts resulting from the given dose. Compartment amounts resulting from prior dose event records are "zeroed out," and infusions in progress or pending additional doses are cancelled. The system is not totally reset: the on/off status of the compartments remains as it was at the time of the prior event record (if any), and the value of time must be greater than or equal to its value on the prior event record (if any).

2 indicates that the dose is a steady state dose and that the compartment amounts are to be set to the sum of the steady-state amounts resulting from the given dose plus whatever those amounts would have been at the event time were the steady-state dose not given. I.e., letting t be the time on the event record, then the amounts in the compartments are updated to amounts valid for time t, and next, these amounts are added to the steady-state amounts. This is meaningful when kinetics are linear and the superposition principle holds.

3 indicates that the dose is a steady state dose. SS = 3 is identical to SS = 1 with one exception: with Steady State routines SS6 and SS9, the existing state vector (compartment amounts and eta derivatives) is used as the initial estimate in the computation of the steady-state amounts. The user supplies the initial estimate with some combination of prior event records, e.g., reset, transient dose, and other-type event records.

When the SS data item is used, one or more of the data items AMT, RATE, and II must be present in the event record to specify the steady state dosing pattern.

REFERENCES: Guide VI, section V.F
REFERENCES: Guide V, section 6.8

SS DOSE EVENT RECORD

MEANING: Specification of steady-state dose for PREDPP
CONTEXT: $INPUT record and NONMEM data set

DISCUSSION:
PREDPP recognizes two varieties of doses, transient and steady-state. Transient doses are described separately (See **dose_event**). Steady-state doses are described here.

(With NM75 there is a new way of computing SS, the Empirical steady state method, in which there is no SS data item, and a negative value of ADDL requests the computation. This is described separately.)
(See **empirical_SS**).
(See Guide Introduction_7)

A steady-state dose is a dose that is imagined to be the last of a series of implied doses, each exactly like the dose in question, given at a regular interval specified by the II data item and leading to steady-state by the time the steady-state dose is given.

For steady-state doses, the SS data item must be defined and positive (See **ss_data_item**).

The ADDL data item may be used on the SS dose event record to specify additional transient doses after the steady-state dose, given at the interval II.

The CMT data item (See **cmt**) applies to steady-state doses as to transient doses.

PK parameters absorption lag (`ALAGn`), bioavailability (`Fn`), modeled rates (`Rn`) and modeled durations (`Dn`) apply to steady-state doses as they do to transient doses, except as noted.

When absorption lag applies, it is understood to apply to all implied doses. Steady-state is computed as if there were no ALAG, and then there is an advance to the appropriate point in the steady-state cycle (II-ALAG).
See **Guide VI, Section V.F.4, Note 4**

MTIME parameters do not apply to steady-state doses.
See **Guide VI, Section V.F.4, Note 3**

When bioavailability applies, it is understood to apply to all implied doses.

Values of AMT, RATE, II data items:

AMT	RATE	II	
>0	0	>0	Steady-state with multiple bolus doses.
>0	>0	>0	Steady-state with multiple infusions. The final infusion is started at the event time of the SS dose record and continues beyond its event time. (More than one infusion continues when the duration of the steady-state infusion is greater than II.)
>0	-1,-2	>0	Steady-state with multiple zero-order bolus doses. The final infusion is started at the event time of the SS dose record and continues beyond its event time. (More than one infusion continues when the duration of the steady-state infusion is greater than II.)

0	0,>0,-1	0	Steady-state with constant infusion. The infusion terminates at the event time of the SS dose record. The ADDL data item cannot be used to specify additional transient doses. Doses with rate 0 are useful when the differential equations coded by the user explicitly provide for endogenous drug production. (The Initial Steady State feature (I_SS) may be specified in the model instead of using a steady-state dose record with rate 0.)

REFERENCES: Guide VI, section V.F
REFERENCES: Guide V, section 6.8

SS OPTION

MEANING: Choice of Steady-State Routine for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES ... [SS=]name
```

SAMPLE:
```
$SUBROUTINE ADVAN=ADVAN6 SS=SS6
```

DISCUSSION:
The SS option tells NM-TRAN that a SS routine should be included in the NONMEM/PREDPP executable. It is optional.

PREDPP contains a library of routines, including SS routines to compute steady-state kinetics. The SS routines all have the same entry name, SS. Most analytic ADVAN routines have a corresponding specific SS routine except for ADVAN8 and ADVAN10, for which SS6 is used.

If the SS data item is present in the $INPUT record, then an SS routine is required in the NONMEM/PREDPP executable. NM-TRAN will supply the SS option of the $SUBROUTINES record if it is not present. The default pairing of ADVAN and SS routines is:

ADVAN1-SS1 ; ADVAN2-SS2 ; ADVAN3-SS3 ; ADVAN4-SS4
ADVAN5-SS5 ; ADVAN6-SS6 ; ADVAN7-SS7 ; ADVAN8-SS6
ADVAN9-SS9 ; ADVAN10-SS6 ADVAN11-SS11 ; ADVAN12-SS12
ADVAN13-SS6; ADVAN14-SS6; ADVAN15-SS9; ADVAN16-SS6
ADVAN17-SS9; ADVAN18-SS6

(With versions prior to NONMEM 7.4, SS13 was used with ADVAN13. See ADVAN13 help entry for other changes to ADVAN13.)

The SS option can be used to request creation of a NONMEM/PREDPP executable which contains an SS routine even though the current data set does not contain the SS data item.

The only real effect of the SS option is to list the name of the desired SS routine in NM-TRAN's FREPORT file. The actual construction of the executable is the users responsibility.

When the SS option is not included on the $SUBROUTINES record, and the SS data item does not appear on the $DATA record, then NM-TRAN lists the null SSS routine (SSS0) in FREPORT and no SS routine is listed. Otherwise, NM-TRAN lists the (non-null) SSS routine ("Supervisor of Steady-State") and the appropriate SS routine.

The nmfe and nmfe.bat commands automatically include the listed SSS and SS routines in the NONMEM executable.

REFERENCES: Guide VI, section VII

SS6 SS9

MEANING: Choice of Steady-State Routine for General Nonlinear Kinetics in PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [SS=]SS6
```

SAMPLE:
```
$SUBROUTINE SS6
```

DISCUSSION:
The SS6 subroutine computes steady-state kinetics in a very general way and may be used with any choice of ADVAN (other than ADVAN9, ADVAN15, or ADVAN17, with which only SS9 can be used). With NONMEM 7.4, SS6 is the only steady-state routine that can be used with ADVAN6, ADVAN8, ADVAN10, ADVAN13, ADVAN14, ADVAN16, ADVAN18 (With earlier versions of NONMEM 7, SS13 was used with ADVAN13.)

When SS6 or SS9 is used, a DES subroutine (or $DES block) is required to evaluate differential equations.

TOL is required; (See **$subroutines**).

With NONMEM 7.4, values of TOL and ATOL may be specified for evaluations of Steady State amounts.
See options SSATOL and SSATOLC of the $SUBROUTINE and $TOL records, and the TOL routine.

Required PREDPP Library subroutines:

With SS6: ZSPOW1, FCN1, FCN2

With SS9: ZSPOW1, FCN2, FCN4

REFERENCES: Guide VI, section VI.C , VII.B , VII.C.6 , VI.C.9
REFERENCES: Guide IV, section V.C.3 , V.C.10

STANDARD ERROR OF ESTIMATE

MEANING: NONMEM's estimate of the precision of its parameter estimates
CONTEXT: NONMEM output

DISCUSSION:
Asymptotic statistical theory applied to extended least-squares estimation (as used in NONMEM) says that the distribution of the parameter estimators is multivariate normal, with variance-covariance matrix that can be estimated from the data. NONMEM supplies such an estimate (See **covariance matrix of estimate**). The square root of the ith diagonal element of the matrix is the standard error of the ith parameter estimate. NONMEM output presents standard errors as in this example.

```
****** STANDARD ERROR OF ESTIMATE ********************

THETA - VECTOR OF FIXED EFFECTS PARAMETERS  **********
           TH 1      TH 2
         6.27E+00  1.92E+01

OMEGA - COV MATRIX FOR RANDOM EFFECTS - ETAS  ********

            ETA1      ETA2
ETA1     1.71E-02
ETA2    .........  1.12E-01

SIGMA - COV MATRIX FOR RANDOM EFFECTS - EPSILONS  ****

            EPS1
 EPS1    5.57E-03
```

Note that standard errors are given for all types of population parameters: THETA, the vector of fixed effects parameters, and OMEGA and SIGMA, the matrices of random effects parameters. In this example, the 2x2 matrix, OMEGA, was constrained to be diagonal; the omitted entry (.....) indicates that the element omega(2,1) is not estimated, and consequently has no standard error.

REFERENCES: Guide I, section C.3.5.2
REFERENCES: Guide V, section 5.4.2.1

STANDARD ERRORS

MEANING: NONMEM read-only global variables
CONTEXT: PRED routine

USAGE:

```
USE ROCM_REAL, ONLY: SETHET=>SETH,SEOMEG=>SEOM,SESIGM=>SESIG,
    SETHETR=>SETHR
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: LTH,LVR,DPSIZE
REAL(KIND=DPSIZES) :: SETH(LTH),SEOM(LVR,LVR),SESIG(LVR,LVR)
```

DISCUSSION:

`SETHET`
: SETHET(i) = the standard error of the estimate of theta(i).

`SETHETR`
: SETHETR(i) = the standard error of the estimate of reported theta(i). If record $THETAR is not used, SETHET and SETHETR are equal. If record $THETAR is used, then SETHET is the standard error of the internal theta as used in $PK/$PRED, and SETHETR is the standard error of the theta reported in the report file.

`SEOMEG`
: SEOMEG(i,j) = the standard error of the estimate of omega(i,j).

`SESIGM`
: SESIGM(i,j) = the standard error of the estimate of sigma(i,j).

These values should only be used with ICALL = 3.

Location prior to NONMEM 7: rocm7

REFERENCES: Guide I, section C.3.5.2

REFERENCES: Guide V, section 5.4.2.1

STATE VECTOR TIME: TSTATE

MEANING: PREDPP read-only global variables
CONTEXT: User-supplied PK and ERROR routine

USAGE:

```
USE PROCM_REAL, ONLY: TSTATE
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: DPSIZE
REAL(KIND=DPSIZE) :: TSTATE
```

DISCUSSION:

`TSTATE`

> `TSTATE` = the time at which the state-vector (See **State Vector: A**) was last computed. It is the previous event time (i.e. the time on the previous event record passed to PK), or if at a later time, but before the time for which PK is being called, a lagged or additional dose was given, or a regular infusion was terminated, or a modeled event occurred, then TSTATE is the latest such time.

NM-TRAN includes this global variable in the PK and ERROR routines when $PK or $ERROR abbreviated code includes references to variables `A(n)` or when verbatim code is present.

Location prior to NONMEM 7: procm9

REFERENCES: None.

STATE VECTOR: A

MEANING: PREDPP read-only global variables
CONTEXT: User-supplied PK and ERROR routines

USAGE:

```
USE PROCM_REAL, ONLY A=>AMNT,DAETA,D2AETA
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: PC,LVR,DPSIZE
REAL(KIND=DPSIZE) :: AMNT(PC),DAETA(PC,LVR),D2AETA(PC,LVR,LVR)
```

DISCUSSION:

`A` A is the state vector of compartment amounts.
`A(n)` = the amount in compartment n.

`DAETA`
`DAETA(n,i)` = the derivative of `A(n)` wrt eta(i).

`D2AETA`
`D2AETA(n,i,j)` = the second derivative of `A(n)` wrt eta(i), eta(j) (lower-triangular; j=1, ..., i).

The A(n) can be used as right-hand quantities in $ERROR and $PK abbreviated code. These amounts are the latest ones computed. With $ERROR, the A(n) are computed at the event time on the event record passed to ERROR. With $PK, the A(n) are computed at the event time on the previous event record, or possibly at a later time. This time - the latest time at which the amounts are computed - is given in the variable TSTATE, which may also be used in $PK abbreviated code.
(See **State_Vector_TIME:_TSTATE**).

Location prior to NONMEM 7: procm4

REFERENCES: Guide VI, section IV.D , Figure 14

STUBS

MEANING: Type of NONMEM, PREDPP or User subroutine and function
CONTEXT: Replaced by a user-written subroutine or function

DISCUSSION:
"Stubs," or "dummy" routines, are subroutines or functions called by NONMEM, PREDPP or user code. The versions of these routines that are distributed with NONMEM are essentially empty, i.e., they do nothing. They can be replaced by user-written code in advanced applications.

Each stub and utility subroutine or function has its own help entry.

1) Stubs for NONMEM

CRIT Modifies the computation of the default objective function.

CONTR Specifies the contribution to the objective function of a level 1 ("L1") record.

CCONTR Specifies the contribution to the objective function of a level 2 ("L2") record.

MIX Describes the mixing parameter of a mixture model.

PRIOR Allows a Bayesian penalty to be included in the objective function.

SPTWO SPTWO can be used to redefine the RES and WRES items for observation records.

THETAI THETAI is used to transform initial thetas.

THETAR THETAR is used to transform final thetas for reporting.

2) stubs for PREDPP

INFN Initialization/Finalization routine.

3) Stubs for User functions

FUNCA, FUNCB, FUNCC, FUNCD, FUNCE, FUNCF, FUNCG, FUNCH, FUNCI
Defines a function for abbreviated code.

Code to replace a function stub should be placed in a file, e.g., funcacode. It should be listed on the $SUBROUTINES record. E.g.,

```
$SUBROUTINES ... OTHER=funcacode
```

REFERENCES: (CRIT) Guide II, section C.6
REFERENCES: (INFN) Guide VI, section VI.A , Figure 37

SUBROUTINE

MEANING: Kind of subroutines in NONMEM/PREDPP Executable
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:

```
$SUBROUTINES [SUBROUTINES=]kind
```

SAMPLE:

```
$SUBROUTINE SUBROUTINES=DP
```

DISCUSSION:
The SUBROUTINES option tells NM-TRAN how to implement the abbreviated code. If there is no abbreviated code, the SUBROUTINES option is ignored. Possible values for kind are:

`DP` This is the default. NM-TRAN generates FORTRAN subroutines from abbreviated code which compute with double-precision arithmetic. These subroutines are written into NM-TRAN output file FSUBS. DP may also be coded DOUBLE or D. Double-precision versions of NONMEM routines (and of PREDPP, if used) should be included when the NONMEM executable is constructed.

NM-TRAN also places into file FREPORT a line describing what kind of subroutines were specified.

```
GENERATED DP SUBROUTINES:
```

REFERENCES: Guide IV, section III.B.6
REFERENCES: Guide III, section V.5.0, V.7.0, V.8.0

SUPER PROBLEM PRINT CONTROL

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

USAGE:
```
USE NMPRD_INT, ONLY IPRNV
```

GLOBAL DECLARATION:
```
INTEGER(KIND=ISIZE) :: IPRNV(2)
```

DISCUSSION:

`IPRNV(i)`

0 indicates no printing of NONMEM input information with iterations 2, 3, etc. of active ith level superproblem.

1 indicates printing of NONMEM input information with iterations 2, 3, etc. of active ith level superproblem.

Status with 2nd level superproblem takes precedence.

(See **$super**).

Location prior to NONMEM 7: rocm28

REFERENCES: None.

SUPP

MEANING: SUPP subroutine
CONTEXT: NONMEM utility routine

USAGE:

```
USE SIZES, ONLY:: ISIZE
INTEGER(KIND=ISIZE):: IE,IC
 ...
CALL SUPP (IE,IC)
```

DISCUSSION:
The NONMEM utility routine SUPP is used to suppress portions of the NONMEM output report. SUPP may be called only when ICALL is 0, 1, or 3.

Input argument:

- `IE` An integer variable or integer constant. If the value is 1, the output from the Estimation Step is not included in the output report.
- `IC` An integer variable or integer constant. If the value is 1, the output from the Covariance Step is not included in the output report.

SUPP is typically used during simulation with numerous sub-problems. A user-supplied INFN routine, or similar code in a user-supplied PRED routine, presumably calculates and prints summary statistics based on the final parameter estimates, etc. Printed values from each sub-problem are of little interest and may be suppressed to reduce the size of the report file.

When IE=1, pages with these headings are suppressed:

```
MINIMUM VALUE OF OBJECTIVE FUNCTION
FINAL PARAMETER ESTIMATE
```

When IC=1, pages with these headings are suppressed:

```
STANDARD ERROR OF ESTIMATE
COVARIANCE MATRIX OF ESTIMATE
CORRELATION MATRIX OF ESTIMATE
INVERSE COVARIANCE MATRIX OF ESTIMATE
```

The option PRINT=0 of the $ESTIMATION record is used to suppress the Intermediate printout from the Estimation Step ("MONITORING OF SEARCH:...")

EXAMPLES OF USAGE:

```
CALL SUPP(1,1)
```

This suppress both Estimation and Covariance Step output.

REFERENCES: None.

TABLE COMPARE

MEANING: TABLE_COMPARE program
CONTEXT: NONMEM run

The utility program table_compare compares the numerical values between two table files produced by the NONMEM $TABLE record. The user may specify the tolerance for the comparison.

USAGE:

```
table_compare table1.tab table2.tab , precision.xtl >differences.txt
```

where delimiter is {, t s} for {comma tab space}, and precision.xtl is a precision specification or control file. Default delimiter is space and default control file is table_compare.xtl. It is useful to redirect difference results to a file, in this example differences.txt.

EXAMPLE:

For example, the user may desire that only relative differences greater than 0.001 be reported. A very simple control file could be:

```
$PRECISION
ALL=0.01,0.003
```

stating that all columns be compared with a relative difference of 0.01, and absolute difference of 0.003. Precision criteria for specific columns in the tables may also be given:

```
$PRECISION
ALL=0.01,0.003 WRES=0.1,0.2
CL=0.05,0.02
```

The equation for comparison is
ABS(X-Y)>R*MAX(ABS(X),ABS(Y))+A

then the difference is reported, where R is relative difference tolerance, and A is absolute difference tolerance.

REFERENCES: Guide Introduction_7

TABLE QUANT

MEANING: TABLE_QUANT program
CONTEXT: NONMEM run

The utility program table_quant transforms importance sampling (SIR) data in the NONMEM raw output file. Such data are present when the option SIRSAMPLE of the $COVARIANCE record is used.
(See Guide Introduction_7, "Importance Sampling of the Variance-Covariance of the Parameter Estimates").
Table_quant produces a table file with frequencies and cumulative values.

USAGE:

```
table_quant inputfile outputfile delimiter start end
```

Inputfile name should be the NONMEM file (root.ext).
Requred

The results are written to outputfile, which may be called root.qnt. (This name is needed if R script quantplot.R, or Splus script, quantplot.ssc, available in Pdx-Pop 5.2, is used to view histograms and cdf plots from these result files.) The output file contains, for each item, the sorted value, its frequency (freq), and its quantile position or cumulative probability (cum).

Required.

Delimiter is that used in the input file (s for space, t for tab, comma default)

start and end are the range of iterations to be quantized (default is all non-negative iterations).

Table_quant uses the WEIGHT values record by NONMEM in the file. Bayes results, which do not have the WEIGHT column, can be processed with this utility, but the weight is then assumed constant among all samples.

The output file will contain for each item, the sorted value, its frequency (freq), and its quantile position or cumulative probability (cum). Also, make sure the input file contains just one table of information.

See also table_resample (tabresamp) program.

EXAMPLE:

REFERENCES: Guide Introduction_7

TABLE RESAMPLE

MEANING: TABLE_RESAMPLE program
CONTEXT: NONMEM run

The utility program table_resample performs a resampling of importance sampling (SIR) data in the NONMEM raw output file. Such data are present when the option SIRSAMPLE of the $COVARIANCE record is used.
(See Guide Introduction_7, "Importance Sampling of the Variance-Covariance of the Parameter Estimates").
Table_resample produces a table file. Samples are weighted according to the WEIGHT column. If no WEIGHT column is present, WEIGHT is assumed to be equal among all samples.

USAGE:

```
table_resample inputfile outputfile delimiter newsize seed start end
```

Inputfile name
: should be the raw NONMEM file (root.ext).

outputfile
: The results are written to outputfile, which is required.

Delimiter
: same as the delimiters that is used in the input file (s for space, t for tab, comma default)

start and end
: The range of iterations to be resampled (default is all non-negative iterations).

newsize
: Size of the new samples, when seed > 0.

seed=0
: Non-randomized expansion of the samples, based on WEIGHT column

seed>0
: Randomized starting at seed, with repeated samples allowed If the user chooses seed>0, then newsize samples will be generated randomly and with replacement, in proportion to the WEIGHT column.

seed<0
: Randomized starting at abs(seed), with repeated samples not allowed.

If the user chooses seed=0, then newsize samples will be generated, each line of the original root.ext file being repeated in proportion to its WEIGHT value, and these repeated samples will be placed in root_new.ext. Thus the weight of each sample is physically expressed in the manner of repeated rows of that sample. To assure that integer truncation does not render the smaller weighted samples to be not at all expressed, newsize should be something like 10000, or even 100000. The resulting file, root_new.ext, will have the same structure as a BAYES result file, without the WEIGHT column, and the R script bayesplot.R or Splus script bayesplot.ssc, available in Pdx-Pop 5.1, may be used to view histograms, quantile plots, and quantile tables.

The seed<0 should be used only if newsize<<min(oldsize of original file, end-

start+1) that is, you just want a to pick a few samples.

See also table_quant (tabquant) program.

EXAMPLE:

REFERENCES: Guide Introduction_7

TABLE TO XML

MEANING: TABLE_TO_XML program
CONTEXT: NONMEM run

Program TABLE_TO_XML is a NONMEM utility program in the ..\util directory. It converts additional output table files produced by NONMEM to XML Formatted files.

USAGE:

```
table_to_xml inputfile outputfile ,
```

where the delimiter is {, t s} for {comma tab space}. Default delimiter is space. The rules (schema, document type definition) by which the xml file is constructed are given in table.xsd and table.dtd in the ..\util directory.

EXAMPLE:

```
table_to_xml my_results.cov my_results_cov.xml
```

REFERENCES: Guide Introduction_7

TIME DATA ITEM

MEANING: Time (TIME) data item for PREDPP
CONTEXT: $INPUT record and NONMEM data set

USAGE:
`$INPUT ... TIME ...`

DISCUSSION:
`TIME` labels PREDPP's time (TIME) data item. The time data item is required with PREDPP. (An exception occurs with ADVAN9, ADVAN15, and ADVAN17 when there are only equilibrium compartments, in which case the time data item is optional.) However, even when PREDPP is not used, a time data item has a special meaning to NM-TRAN; see below.

With PREDPP, the time data item gives the time of the event, the "event time." With NONMEM 7.4, time data items may be negative; with earlier versions, time data items must be non-negative. Event records must be ordered within an individual record so that time does not decrease, i.e., so that time on any record other than the first is greater than or equal to time on the prior record. Exceptions are the event times of reset and reset-and-dose events, where the time may be any number.

With or without PREDPP, if any time data item in the NM-TRAN data set contains a colon (:), then all time data items are assumed to be clock times (hh:min), and NM-TRAN translates all times to relative times (hh.fr) starting at time 0. Specifically, the time of the first data record of an individual record (if the data are population data) or the first data record of the data set (if the data are single-subject data) is set to 0. Subsequent times in the NONMEM data set are calculated relative to that time. (See **date data item**).

Example:

NM-TRAN data set TIME	NONMEM data set TIME
----	----
`8:00`	`0.`
`14:45`	`6.75`
`14.75`	`6.75`

NM-TRAN permits negative clock times.

The $DATA record may include the option `TRANSLATE=(TIME/24)`, in which case all relative times (obtained after clock time conversion) are divided by 24. (See **$data**).
REFERENCES: Guide VI, section V.C
REFERENCES: Guide IV, section II.C.2 , III.B.2

TIME SCALE PARAMETER

MEANING: Time scale (TSCALE) parameter for PREDPP
CONTEXT: Additional PK Parameters

USAGE:

```
$PK
 TSCALE= ....
```

DISCUSSION:

The time scale parameter is used with PREDPP. It is an optional additional PK parameter. With NM-TRAN, it is symbolized in the $PK block by either of the reserved variables `TSCALE` or `XSCALE`.

If a time scale parameter is modeled by PK, then time itself may be scaled differently between individuals. There is a single time scale parameter that multiplies all rate constants. The parameter acts continuously (and could therefore theoretically itself vary with time measured on an external clock). It can only be used with linear kinetic models. If it is not used, it can be ignored. If the value of the time scale parameter is not computed in PK, it is always understood to be 1.

The time scale parameter does not scale the duration parameter D of a duration-modeled zero-order bolus dose.

PREDPP ignores the time scale parameter with general non-linear models (ADVAN6, 8, 9, 13,14,15,16,17,18).

(See **pk**, **$pk**).

REFERENCES: Guide IV, section V.C.5
REFERENCES: Guide VI, section III.F.8

TNPRI

MEANING: TNPRI subroutine
CONTEXT: NONMEM utility routine

USAGE:

```
USE SIZES, ONLY: ISIZE,DPSIZE
INTEGER (KIND=ISIZE) :: IFND,MODE,ITYP,NSAM,ISS,IVAR
REAL (KIND=DPSIZE) :: PLEV
REAL (KIND=DPSIZE) :: CNT
REAL (KIND=ISIZE) :: ICALL
CALL TNPRI (IFND,MODE,ITYP,PLEV,NSAM,ISS,IVAR,CNT)
```

DISCUSSION:

The model parameters THETA, OMEGA and SIGMA parameters may be constrained in various ways. Those parameters whose values are not fixed are transformed in a one-to-one manner to parameters whose values are not at all constrained a priori (the "unconstrained parameters", or UCP). The user-written PRIOR subroutine allows a penalty function based on a frequency prior to be specified and added to the -2log likelihood function (Gisleskog et al, JPP, 2002, p. 473-505). This function serves as a constraint on the *estimates* of THETA, OMEGA and SIGMA and thus as a way for stable estimates to be obtained with insufficient data. TNPRI may be called by PRIOR. (See **prior**). It computes a penalty function based on a frequency prior that has a multivariate normal form for all the UCP. The one-to-one transformation between THETA, OMEGA and SIGMA and the UCP induces a frequency prior for THETA, OMEGA and SIGMA itself. This latter form is called the "transformed normal" form.

Both forms have the same parameters, which are called "hyperparameters". The values of the hyperparameters are produced with a prior NONMEM problem (called the "prior problem") used to analyze a prior data set. Thus the frequency prior has an empirical nature. The prior problem should have implemented the Covariance Step (the Estimation Step could have been implemented in yet an earlier problem). Whenever a problem implements the Covariance Step and a model specification file has been output, this file *automatically* contains information - referred to as the "prior information", and including values for the hyperparameters - which can be used when TNPRI is used in a subsequent problem.

The abilty to use TNPRI suggests that with any serious analysis undertaken in the future, though it may not be necessary to save the data that were analyzed, a model specification file should be produced and safely stored along with a control stream showing the particulars that gave rise to this file. There is a special way to input the prior information from the model specification file for its use with a subsequent application of TNPRI. (Alternatively, a NONMEM control stream may consist of multiple problems, where one problem (A) uses TNPRI, and the prior problem appears as an earlier problem (B) in the same control stream. The prior information from problem B can be available to problem A without the need to use a model specification file.)

When TNPRI is used during a Simulation Step, it produces a random value of the vector of all model parameters (whose values are not fixed in the parameter records) from the frequency prior. (See **Simulation example**). (See **tnpri example**). If TNPRI is used during a given problem at ICALL=2, it should not be used during the same problem at ICALL=4, and vice-versa.

TNPRI may be used at ICALL=0 or ICALL=1, in which case only the values of its input arguments are checked. If it is not used at ICALL=0 or ICALL=1, then the first time it is used at ICALL=2 or ICALL=4, checking will occur.

Do not use TNPRI with the new $ESTIMATION methods of NONMEM 7.

Input argument:

IFND

0: Indicates that a prior problem has been specified either earlier in the current control stream or in a previously run control stream, and that this problem has output a model specification file. The prior information will have automatically been stored in the model specification file. The prior problem should have implemented the Covariance Step (which should have computed either the default variance-covariance matrix or two times the inverse of the R matrix). The current control stream should consist of at least two problems: the problem that uses TNPRI and a preceeding problem that serves only to input the prior information from the model specification file, such as the following:

```
$PROB   READ THE MODEL SPECIFICATION FILE
$DATA data
$INPUT ID TIME AMT DV TYPE SS II
$MSFI msf1  ONLYREAD
```

This simple problem (B) should be the last problem in the current control stream that appears before the problem using TNPRI (problem (A)) and inputs a model specification file. The problem specification should not include any task records, but it may include e.g. a $SUBROUTINES record and/or abbreviated code, if problem B is the first problem in the control stream, and these elements will be needed for a subsequent problem. If problem A inputs a model specification file msf2, any prior information in msf2 will be ignored. If problem A outputs a model specification file msf3 and implements the Covariance Step, the prior information stored in msf3 is based not only on the data being analyzed, but also on the prior information in msf1.

1: Indicates that along with the current problem (with which TNPRI is being used), a prior problem has been specified earlier in the same control stream. The prior problem is taken to be the last problem in the control stream that appears before the current one and that implements the Covariance Step (which should have computed either the default variance-covariance matrix or two times the inverse of the R matrix).

MODE

Discussion: In any NONMEM problem specification, all of the last contiguously listed parameters on the $THETA record, whose values on this record are fixed, and assuming the value of the very last parameter itself is fixed, are called the "terminal fixed THETA parameters". There may be no such parameters. E.g. if the $THETA record is

```
$THETA 3.1 FIX 6.3 .01 400 FIX 7 FIX 90 FIX
```

then THETA parameters 4-6 are the terminal fixed THETA parameters. If the record is

```
$THETA 3.1 FIX 6.3 .01 400 FIX 7 FIX 90
```

then there are no terminal fixed THETA parameters, because the value 90 of the very last parameter is not fixed. Similarly, all the last contiguous fixed parameters in OMEGA (SIGMA), assuming the values in the very last block set are fixed, are called the "terminal fixed OMEGA (SIGMA) parameters". (Recall that in a diagonal $OMEGA record, all the values form separate block sets.)

There are some rules concerning the parameter records of a NONMEM control stream that relate the parameters of the prior problem to those of the current problem:

I. With the exception of the terminal fixed THETA parameters in the prior problem, which are altogether ignored in the current problem, the THETA parameters of the prior problem must be the first THETA parameters listed in the $THETA record of the current problem, whether or not their values are fixed in the prior problem.

II. The order of these first THETA parameters on the $THETA record must be the same as that in the $THETA record of the prior problem. The initial estimates of these parameters need not be the same, but any upper or lower bounds must be the same.

III. Among the first THETA parameters, values that are fixed on the $THETA record of the prior problem must also be fixed on the $THETA record of the current problem, although the values need not be the same, but not vice-versa; see discussion below concerning parameter interpretation.

IV. THETA parameters peculiar to the current problem may be listed at the end of the $THETA record of the current problem, after the parameters that are shared between the prior and current problems.

E.g. if the $THETA record with the prior problem is

```
$THETA 3.1 FIX 6.3 (0,.01) 400 FIX 7 FIX 90 FIX
```

then the $THETA record with the current problem may be

```
$THETA 10  FIX 7.0 (0,.03) 80.2 100.8
```

in which case the values 80.2 and 100.8 are the values of the THETA parameters that are peculiar to the current problem. The 2nd and 3rd parameters are used in both prior and current problems, but have different values in the two problems. The first parameter may be used in the current problem, in which case its value is fixed to 10.

The same rules apply to the $OMEGA and $SIGMA records of the current problem.

A model specification file from a previous instance of the current problem may be used, as usual, providing that if with the previous instance, parameter records were used, the above rules were followed.

Parameter Interpretation: The prior information concerns all parameters appearing in the prior problem whose values are not fixed in that problem. However, a parameter with a nonfixed value in the prior problem may have a fixed value in the current problem, and this gives rise to some ambiguity. Such a parameter may have an interpretation in the prior problem that remains unchanged in the current problem. Then it needs to be identified as a "shared parameter", in which case, because the value of the parameter is fixed with the current model, the frequency prior needs to be

adjusted in order to approximate the -2 log likelihood function for the prior data when the parameter value is regarded as being fixed in the prior model. (or when TNPRI is being called in the Simulated Step, in order to use a *conditional* prior distribution, given the parameter value is the fixed value). A shared parameter may not actually be used in the current problem, but the salient point is that with the current problem, the interpretation of this parameter remains unchanged. Alternatively, the parameter's interpretation may change in the current problem. Then it needs to be identified as a "prior-specific" parameter, in which case if it is actually used in the current model, its value will be the fixed value, but for the purpose of applying the prior information, TNPRI will regard this parameter as one specific to the prior model. The MODE argument concerns this distinction.

0: All parameters with a nonfixed value in the prior problem but a fixed value in the current problem are identified as being prior-specific parameters. However, the final estimates of these parameters are their fixed values.

1: All parameters with a nonfixed value in the prior problem but a fixed value in the current problem are identified as being prior-specific parameters. If the Estimation Step is implemented, the final estimate for such a parameter is the same one that would result from not fixing the value of the parameter in the current problem and letting it be estimated, *assuming* that the parameter is not at all used in the current model. This value will be similar to the final estimate of the parameter in the prior problem. (A difference from this prior estimate may result, reflecting the fact that if there is a sequence of models, each a submodel of the one succeeding it, then the data used to fit the last of the models may bear on a parameter peculiar to the first model, if only slightly.) If indeed the parameter's value were not fixed, but the parameter's estimate obtained along with the estimates of the non-prior-specific parameters, then the latter estimates and the minimum value of the objective function would be unchanged, but the search for the parameter's estimate would require additional computer time. With the value MODE=1, in fact, the parameter's estimate is obtainable posthoc, after the search, and this additional computer time is saved. The output from the Covariance Step includes all the usual type of information about the estimators of the prior-specific parameters along with that about the estimators of the non-prior-specific parameters. However, note that an output model specification file will not contain any information about the estimates of prior-specific parameters.

2: All parameters with a nonfixed value in the prior problem but a fixed value in the current problem are identified as being shared parameters (i.e. the frequency prior is adjusted). The final estimates of these parameters are their fixed values.

`ITYP`

Relevant only if TNPRI is called during a Simulation Step. Values are:

0: The value of the UCP (unconstrained parameter) vector is obtained from simple random sampling.

1: Within the given problem, TNPRI is to be called a specified number (NSAM) of times to obtain this number of different values of the UCP vector. These values are obtained by generating a Latin sample of size NSAM from equiprobable partitions of an ellipsoid in UCP space (hyper-ellipsoidal sampling), followed by sampling a

point "uniformly" from each partition. This scheme may be used, for example, when the problem has NSAM subproblems, in which case, TNPRI would be called NSAM times, once each time during the problem when ICALL=4, and at each of these calls, a different random value of the UCP vector will be produced.

2: Just as with value 1, but the NSAM values are obtained by generating a Latin sample of size NSAM from equiprobable partitions of an ellipsoid in UCP space (hyper-ellipsoidal sampling), followed by taking the "center point" of each partition.

In all three cases, a value of the UCP vector is transformed into THETA-OMEGA-SIGMA space.

After each call to NWPRI, the simulated values for THETA, OMEGA and SIGMA may be found in global variables and thus they are communicated directly to NONMEM. (See **PRIOR_Simulation:_Parameters**).

`PLEV`

When TNPRI is being used at ICALL=0 or 1, but TNPRI will not be used at ICALL=4 (i.e. during the Simulation Step), `PLEV` should be set to 0. When it is being used at ICALL=2, `PLEV` should also be set to 0. When TNPRI is being used at ICALL=0 or 1 and will also be used at ICALL=4, or when it is being used at ICALL=4, then `PLEV` must be set to a fraction strictly less than 1, e.g. 0.999. PLEV is double precision with NONMEM 7, and is single precision with NONMEM VI.

A UCP value will actually be obtained using a truncated multivariate normal distribution, i.e. from an ellipsoidal region R1 over which only a fraction of mass of the normal occurs. This fraction is given by PLEV. Simple random sampling occurs in R1. Latin sampled partitions are partitions of R1.

`NSAM`

Relevant only if TNPRI is called during a Simulation Step. Consider two cases. a) Latin hyper-ellipsoid sampling is used with ITYP=1, or b) simple random sampling along with the adjustment for small sample correlation effect is used (see next input argument). In case a) NSAM should equal the exact total number of different values of the parameter vector that must eventually be produced over the entire NONMEM problem. In case b) NSAM should be no less than this number.

`ISS`

Relevant only if TNPRI is called during a Simulation Step. A value of the UCP vector is obtained by first sampling a value from the *standard* multivariate normal distribution - called here "the standard value" and then transforming this value to one from the appropriate multivariate normal. The correlation matrix of the standard normal is the identity matrix. When NSAM is small, the estimated correlation matrix from the sampled standard values might not be quite close to the identity matrix - this is here called "the small sample correlation effect".

1: An adjustment is made for the small sample correlation effect, by first transforming the NSAM standard values *altogether* into new values which are very nearly standard multivariate normal values and such that the sample correlation matrix of these new values is exactly the identity matrix.

0: No adjustment is made for the small sample correlation effect.

IVAR

Relevant only if TNPRI is called during a Simulation Step. When TNPRI simulates a parameter value, this value is that of an estimate of the parameter from possible data under the prior model, and this simulation is based on asymptotic statistical theory. The partial derivatives comprising the R and S matrices referred to below are taken with respect to the UCP (See **covariance**). The possible values of IVAR are:

0: The variance-covariance matrix of the multivariate normal on the UCP is taken to be two times the inverse R matrix obtained from the prior problem. This could be appropriate if the estimated asymptotic variance-covariance matrix from the prior problem is also based only on the R matrix.

1: The variance-covariance matrix of the multivariate normal on the UCP is taken to be Rinv*S*Rinv matrix from the prior problem.

2: The variance-covariance matrix of the multivariate normal on the UCP is taken to be S matrix from the prior problem. (NM75)

Briefly:
IVAR=0: Uses Rinv of a former problem.
IVAR=1: Uses Rinv*S*Rinv of a former problem
IVAR=2: Uses S of a former problem.

Output argument:

CNT

Relevant only if TNPRI is called at ICALL=2. CNT is the penalty.

REFERENCES: Guide II, section D.2.5

TOL

MEANING: TOL subroutine
CONTEXT: User-supplied subroutine; for use with PREDPP

USAGE:

Versions before NONMEM 7.4:

```
SUBROUTINE TOL(NRD)
USE SIZES,     ONLY: ISIZE
INTEGER(KIND=ISIZE) :: NRD
DIMENSION :: NRD(*)
```

With NONMEM 7.4:

```
SUBROUTINE TOL(NRD,ANRD,NRDC,ANRDC)
USE SIZES,     ONLY: ISIZE
INTEGER(KIND=ISIZE) :: NRD(0:*), ANRD(0:*), NRDC(0:*), ANRDC(0:*)
```

Optional declarations with NONMEM 7.4:

```
USE NMPRD_INT, ONLY: IPROB
USE NM_BAYES_INT, ONLY: NM_STEP,BASE_STEP,EST_STEP,COV_STEP, &
 TABLE_STEP,SIML_STEP,INE_STEP, NONP_STEP
```

DISCUSSION:
The TOL subroutine is called by PREDPP when ADVAN 6, 8, 9, 10, 13, 14, 15, 16, 17, or 18 is used. It is also called when SS6 or SS9 is used. With NONMEM 7.4, there are multiple calls during the run, for each NONMEM step. With earlier version, TOL is called only once at the start of a run.

Output argument:

`NRD(I)`

The number of digits that are required to be accurate in the computation of the drug amount in compartment I, i.e., the relative tolerance. ADVAN 9, 13, 16, 17, and 18 have the capability of using
different values of NRD for different compartments (ADVAN 14 and 15 only allow different absolute tolerances for each compartment). For compartments not specified, the tolerance of the last compartment specified will be used.

However, all the other ADVAN routines requiring TOL take the relative tolerance to be the same for all compartments; NRD(I), $I > 1$, is ignored, and only NRD(1) is used. With NONMEM 7.4, NRD(0) is the relative tolerance for the Steady State computations. If NRD(0) is not specified, NRD(1) is used.

The value of NRD(1) can also be specified using $SUBROUTINES option TOL.
The value of NRD(0) can also be specified using $SUBROUTINES option SSTOL.

`ANRD(I)` (NM74)

The absolute tolerance in the computation of the drug amount in compartment I. The default is 12 (that is, accuracy is 10**(-12)). Used by ADVAN 9, 13, 14, 15, 16, 17, and 18, which have the capability of using
different values of ANRD for different compartments. For compartments not specified, the tolerance of the last compartment specified will be used.

ANRD(0) is the absolute tolerance for the Steady State computations. If ANRD(0) is not specified, ANRD(1) is used.

The value of ANRD(1) can also be specified using $SUBROUTINES option ATOL. The value of ANRD(0) can also be specified using $SUBROUTINES option SSATOL.

`NRDC(I)` (NM74)

Same as NRD(I), but used for the FOCE/LAPLACE covariance step. Used with ADVAN 9, 13, 14, and 15, 16, 17, and 18. If not set, NRDC defaults to the value of NRD. NRDC(0) is used for Steady State computations during the FOCE/LAPLACE covariance step.

The value of NRDC(1) can also be specified using $SUBROUTINES option TOLC. The value of NRDC(0) can also be specified using $SUBROUTINES option SSTOLC.

`ANRDC(I)` (NM74)

Same as ANRD(I), but used for the FOCE/LAPLACE covariance step. Used with ADVAN 9, 13, 14, and 15, 16, 17, and 18. If not set, ANRDC defaults to the value of ANRD. ANRDC(0) is used for Steady State computations during the FOCE/LAPLACE covariance step.

The value of ANRDC(1) can also be specified using $SUBROUTINES option ATOLC.
The value of ANRDC(0) can also be specified using $SUBROUTINES option SSATOLC.

When NM-TRAN is used, this information may be supplied by either the TOL option of the $SUBROUTINES record or the $TOL record.

Finally, you may supply a TOL routine that assigns values of NRD and ANRD specifically for the initial (base) setting and each NONMEM step (estimation, covariance, simulation, table/scatter step, simulation, initial parameters estimate, nonparametric). For example, create a toluser.f90 file,

```
SUBROUTINE TOL(NRD,ANRD,NRDC,ANRDC)
USE NMPRD_INT, ONLY: IPROB
USE NM_BAYES_INT, ONLY: NM_STEP,BASE_STEP,EST_STEP,COV_STEP, &
 TABLE_STEP,SIML_STEP,INE_STEP, NONP_STEP
IMPLICIT NONE
INTEGER :: NRD(0:*), ANRD(0:*), NRDC(0:*), ANRDC(0:*)
IF(NM_STEP==EST_STEP) THEN
NRD(1)=6
ANRD(1)=10
ELSE IF (NM_STEP==COV_STEP) THEN
NRD(1)=7
ANRD(1)=8
ELSE IF (NM_STEP==TABLE_STEP) THEN
NRD(1)=8
ANRD(1)=7
ELSE
NRD(1)=9
ANRD(1)=12
ENDIF
IF(IPROB>1) THEN
NRD(1)=NRD(1)+1
ANRD(1)=ANRD(1)+1
```

```
ENDIF
RETURN
END
```

and incorporate using $SUBR, e.g.,

```
$SUBROUTINES ADVAN13 TRANS1 TOL=toluser.f90
```

REFERENCES: Guide IV, section V.C.4
REFERENCES: Guide VI, section VI.D , Figure 41
REFERENCES: Guide VI, Appendix II

TRANS (OPTION)

MEANING: Choice of Translator Routine for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [TRANS=]TRANSn
```

SAMPLE:
```
$SUBROUTINE ADVAN1 TRANS1
```

DISCUSSION:
PREDPP contains a library of routines, called translator routines, which perform re-parameterization of the basic PK parameters computed by the PK routine. Exactly one TRANS routine must be selected for each NONMEM/PREDPP run. They are:

- TRANS1 Dummy, or null, translator; may be used with all ADVAN routines. It is the default.
- TRANS2 Used with ADVAN1 and ADVAN2.
- TRANS3 Used with ADVAN3 and ADVAN4.
- TRANS4 Used with ADVAN3, ADVAN4, ADVAN11, ADVAN12
- TRANS5 Used with ADVAN3 and ADVAN4.
- TRANS6 Used with ADVAN3, ADVAN4, ADVAN11, ADVAN12

Each has its own entry.

REFERENCES: Guide VI, section III.A , III.M

TRANS (SUBROUTINE)

MEANING: TRANS subroutine
CONTEXT: User-supplied subroutine; for use with PREDPP

USAGE:

```
SUBROUTINE TRANS (ITRANS,IRGG,GG,NETAS)
USE SIZES, ONLY: ISIZE,DPSIZE,LVR
INTEGER(KIND=ISIZE) :: IRGG,NPETAS,ITRANS
REAL(KIND=DPSIZE)   :: GG(IRGG,LVR+1,LVR+1)
```

DISCUSSION:
The TRANS subroutine translates (or transforms) the values for a set of basic PK parameters modeled in PK to a set of values for the internal parameters required by the ADVAN. The PREDPP Library has a number of TRANS subroutines, representing different possible parameterizations in PK, from which the user may choose. If a suitable translator is not found in the Library, the user may write his own.

Input/Output argument:

`ITRANS`

ITRANS=1: TRANS has been called for initialization at the beginning of a NONMEM problem; one such call per problem. ITRANS must be reset by TRANS to a number in the range 1-8999. This number appears on NONMEM output, allowing the user to identify the TRANS routine being used.

ITRANS=2: On input the GG array has stored in it the values computed by PK (except that were any PK parameter modeled in its logarithm form in PK, PREDPP would have already exponentiated its typical/subject-specific value and multiplied its eta derivatives by its exponentiated typical/subject-specific value). On output the GG array should have stored in it the values that would be computed by PK were the internal parameters of the ADVAN modeled directly in PK (and none in their logarithmic form).

ITRANS=4: TRANS has been called during the Simulation Step. Only the first column of the GG array need be computed as with ITRANS=2. Other columns need not be computed.

`GG` The array of PK parameters and their eta derivatives.
The array is described elsewhere; (See **pk**).

Input argument:

`NETAS`

The number of population etas in the problem.

Certain variables are available in modules. Their use is optional.

Variables in read/write modules:

`IERPRD NETEXT ETEXT`
(See **PRED Exit Code**).
(See **PRED Error Message**).

Variables in read-only commons:

`NOETAS, SECOND`
(See **Partial Derivative Indicators**).

When SECOND is true, PREDPP requires second-partial derivatives of etas.

REFERENCES: Guide VI, section III.M

TRANS1

MEANING: Choice of Translator Routine for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [TRANS=]TRANS1
```

SAMPLE:
```
$SUBROUTINE ADVAN1 TRANS1
```

DISCUSSION:
TRANS1 is a translator routine in PREDPP's library which can be used with any ADVAN routine. It is a "dummy" translator that does not perform any reparameterization of the basic PK parameters. It is the default translator when none is specified on the $SUBROUTINES record.

REFERENCES: Guide VI, section III.A , III.M

TRANS2

MEANING: Choice of Translator Routine for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [TRANS=]TRANS2
```

SAMPLE:
```
$SUBROUTINE ADVAN1 TRANS2
```

DISCUSSION:
TRANS2 is a routine in PREDPP's library which can be used with ADVAN1 or ADVAN2. It performs a reparameterization of the basic PK parameters to the internal parameters K and KA.

Basic PK parameters for ADVAN1, TRANS2:

`CL` clearance
`V` volume of distribution

Relationship: K=CL/V

Basic PK parameters for ADVAN2, TRANS2:

`CL` clearance
`V` volume of distribution
`KA` rate constant of absorption

Relationship: K=CL/V
KA=KA

REFERENCES: Guide VI, section III.A , VII.C.1

TRANS3

MEANING: Choice of Translator Routine for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [TRANS=]TRANS3
```

SAMPLE:
```
$SUBROUTINE ADVAN3 TRANS3
```

DISCUSSION:
TRANS3 is a routine in PREDPP's library which can be used with ADVAN3 or ADVAN4. It performs a reparameterization of the basic PK parameters to the internal parameters K, K12, K21, and KA.

Basic PK parameters for ADVAN3, TRANS3:

`CL`	clearance
`V`	central volume
`Q`	intercompartmental clearance
`VSS`	volume of distribution at steady-state

Relationship:
K=CL/V
K12=Q/V
K21=Q/(VSS-V)

Basic PK parameters for ADVAN4, TRANS3:

`CL`	clearance
`V`	central volume
`Q`	intercompartmental clearance
`VSS`	volume of distribution at steady-state
`KA`	absorption rate constant

Relationship:
K=CL/V
K23=Q/V
K32=Q/(VSS-V)
KA=KA

REFERENCES: Guide VI, section III.A , VII.C.3

TRANS4

MEANING: Choice of Translator Routine for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [TRANS=]TRANS4
```

SAMPLE:
```
$SUBROUTINE ADVAN3 TRANS4
```

DISCUSSION:
TRANS4 is a routine in PREDPP's library which can be used with ADVAN3, ADVAN4, ADVAN11, or ADVAN12. It performs a reparameterization of the basic PK parameters to the internal parameters K, K12, K21, and KA.

Basic PK parameters for ADVAN3, TRANS4:

`CL`	clearance
`V1`	central volume
`Q`	intercompartmental clearance
`V2`	peripheral volume

Relationship:
K=CL/V1
K12=Q/V1
K21=Q/V2

Basic PK parameters for ADVAN4, TRANS4:

`CL`	clearance
`V2`	central volume
`Q`	intercompartmental clearance
`V3`	peripheral volume
`KA`	absorption rate constant

Relationship:
K=CL/V2
K23=Q/V2
K32=Q/V3
KA is unchanged

Basic PK parameters for ADVAN11, TRANS4:

`CL`	clearance
`V1`	central volume
`Q2`	intercompartmental clearance (central and periph. 1)
`V2`	peripheral 1 volume
`Q3`	intercompartmental clearance (central and periph. 2)
`V3`	peripheral 2 volume

Relationship:
K=CL/V1
K12=Q2/V1
K21=Q2/V2
K13=Q3/V1
K31=Q3/V3

Basic PK parameters for ADVAN12, TRANS4:

`CL`	clearance
`V2`	central volume
`Q3`	intercompartmental clearance (central and periph 1)
`V3`	peripheral 1 volume
`Q4`	intercompartmental clearance (central and periph 2)
`V4`	peripheral 2 volume
`KA`	absorption rate constant

Relationship:
K=CL/V2
K23=Q3/V2
K32=Q3/V3
K24=Q4/V2
K42=Q4/V4
KA=KA

REFERENCES: Guide VI, section VII.C.3 , VII.C.4

TRANS5

MEANING: Choice of Translator Routine for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [TRANS=]TRANS5
```

SAMPLE:
```
$SUBROUTINE ADVAN3 TRANS5
```

DISCUSSION:
TRANS5 is a routine in PREDPP's library which can be used with ADVAN3 or ADVAN4. It performs a reparameterization of the basic PK parameters to the internal parameters K, K12, K21, and KA.

Basic PK parameters for ADVAN3, TRANS5:

`AOB`	A/B
`ALPHA`	alpha
`BETA`	beta

Relationship:
K21=(AOB*BETA+ALPHA)/(AOB+1)
K=(ALPHA*BETA)/K21
K12=ALPHA+BETA-K21-K

Basic PK parameters for ADVAN4, TRANS5:

`AOB`	A/B
`ALPHA`	alpha
`BETA`	beta
`KA`	absorption rate constant

Relationship:
K32=(AOB*BETA+ALPHA)/(AOB+1)
K=(ALPHA*BETA)/K32
K23=ALPHA+BETA-K32-K
KA=KA

REFERENCES: Guide VI, section VII.C.3 , VII.C.4

TRANS6

MEANING: Choice of Translator Routine for PREDPP
CONTEXT: Option of NM-TRAN $SUBROUTINES record

USAGE:
```
$SUBROUTINES [TRANS=]TRANS6
```

SAMPLE:
```
$SUBROUTINE ADVAN3 TRANS6
```

DISCUSSION:
TRANS6 is a routine in PREDPP's library which can be used with ADVAN3, ADVAN4, ADVAN11, and ADVAN12. It performs a reparameterization of the basic PK parameters.

Basic PK parameters for ADVAN3, TRANS6:

`ALPHA`	alpha
`BETA`	beta
`K21`	rate constant (peripheral to central)

Relationship:
K=ALPHA*BETA/K21
K12=ALPHA+BETA-K21-K

Constraint:

Assuming that ALPHA < BETA, then ALPHA < K21 < BETA.
The roles of ALPHA and BETA are exchangeable.

Basic PK parameters for ADVAN4, TRANS6:

`ALPHA`	alpha
`BETA`	beta
`K32`	rate constant (peripheral to central)
`KA`	absorption rate constant

Relationship:
K=ALPHA*BETA/K32
K23=ALPHA+BETA-K32-K
KA is unchanged

Constraint:

Assuming that ALPHA < BETA, then ALPHA < K32 < BETA.
The roles of ALPHA and BETA are exchangeable.

Basic PK parameters for ADVAN11, TRANS6:

`ALPHA`	alpha
`BETA`	beta
`GAMMA`	gamma
`K21`	rate constant (peripheral 1 to central)
`K31`	rate constant (peripheral 2 to central)

Relationship:
K=ALPHA*BETA*GAMMA/(K21*K31)
K13=(P+K31*K31-K31*S-K*K21)/(K21-K31)
K12=S-K-K13-K21-K31

where
S=ALPHA+BETA+GAMMA
P=ALPHA*BETA+ALPHA*GAMMA+BETA*GAMMA

Constraint:

Assuming that ALPHA<BETA<GAMMA, then
ALPHA<K21<BETA<K31<GAMMA
or
ALPHA<K31<BETA<K21<GAMMA.
The roles of ALPHA, BETA, GAMMA, K21, K31, K12, and K13 are symmetric and are exchangeable.

Basic PK parameters for ADVAN12, TRANS6:

`ALPHA`	alpha
`BETA`	beta
`GAMMA`	gamma
`K32`	rate constant (peripheral 1 to central)
`K42`	rate constant (peripheral 2 to central)
`KA`	absorption rate constant

Relationship:
K=ALPHA*BETA*GAMMA/(K32*K42)
K24=(P+K42*K42-K42*S-K*K32)/(K32-K42)
K23=S-K-K24-K32-K42
where
S=ALPHA+BETA+GAMMA
P=ALPHA*BETA+ALPHA*GAMMA+BETA*GAMMA
KA is unchanged

Constraint:

Assuming that ALPHA<BETA<GAMMA, then
ALPHA<K32<BETA<K42<GAMMA
or
ALPHA<K42<BETA<K32<GAMMA.
The roles of ALPHA, BETA, GAMMA, K32, K42, K23, and K24 are symmetric and are exchangeable.

REFERENCES: None.

USERS GUIDE

MEANING: NONMEM Users Guide
CONTEXT: Documentation

The following documents are collectively called the NONMEM Users Guide.

Part I - Users Basic Guide

Part II - Users Supplemental Guide

Part III - NONMEM Installation Guide

Part IV - NM-TRAN Guide

Part V - NONMEM-PREDPP Introductory Guide

Part VI - PREDPP Guide

Part VII - Conditional Estimation Methods

Part VIII - Help Guide

NONMEM V Supplemental Guide

Introduction to Version VI

Introduction to NONMEM 7

They are distributed as PDF files on the NONMEM distribution media. They are also present in HTML format as part of the HTML version of On-line help (Hyper-NMhelp).

REFERENCES: Guide I
REFERENCES: Guide II
REFERENCES: Guide III
REFERENCES: Guide IV
REFERENCES: Guide V
REFERENCES: Guide VI
REFERENCES: Guide VII
REFERENCES: Guide VIII
REFERENCES: Guide Supplemental_V
REFERENCES: Guide Introduction_VI
REFERENCES: Guide Introduction_7

VARIABLES IN MODULES

MEANING: Variables in Modules
CONTEXT: Abbreviated code

DISCUSSION:
Variables in certain FORTRAN modules may be used in abbreviated code. With NONMEM 7.2 and higher, either upper or lower case may be used.

Variables in modules whose names start `ROCM` or `PROCM` may be used only on the right.

Variables in modules whose names start `NMPR` or `PRDPK` may always be used on the right and may sometimes be used on the left.

Names of modules are reserved and may not be used in any block of abbreviated code.

Provided is a list of blocks of abbreviated code and the modules and variables that may be used. Second is a list of modules, the variables in the modules, and the blocks of abbreviated code in which these variables may be used.

```
$AES
NMPRD_INT:  COMACT,COMSAV
NMPRD_INT:  NPROB,IPROB
NMPRD_INT:  S1IT,S1NIT,S1NUM,S2IT,S2NIT,S2NUM
NMPRD_INT:  NTHES_,NETAS_,NEPSS_
NMPR_INT:   RPTI,RPTO,RPTON,PRDFL
PKERR_REAL: MTIME
PRMOD_INT:  ICALL,IDEFD,IDEFA
PRMOD_INT:  MTDIFF
PROCM_INT:  ISFINL
PROCM_INT:  MNOW,MPAST,MNEXT
PROCM_INT:  NVNT
PROCM_REAL: DOSREC
PROCM_REAL: DOSTIM
PROCM_REAL: EVTREC
PROCM_REAL: THETA
ROCM_INT:   MIXNUM,MIXEST
ROCM_INT:   NIREC,NDREC
ROCM_INT:   NREP,IREP
ROCM_REAL:  MIXP

$DES
NMPRD_INT:  COMACT,COMSAV
NMPRD_INT:  MSEC,MFIRST,IFIRSTEM
NMPRD_INT:  NPROB,IPROB
NMPRD_INT:  S1IT,S1NIT,S1NUM,S2IT,S2NIT,S2NUM
NMPRD_INT:  NTHES_,NETAS_,NEPSS_
NMPR_INT:   RPTI,RPTO,RPTON,PRDFL
PKERR_REAL: MTIME
PRMOD_INT:  ICALL,IDEFD,IDEFA
PRMOD_INT:  MTDIFF
PROCM_INT:  ISFINL
PROCM_INT:  MNOW,MPAST,MNEXT
```

```
PROCM_INT:  NVNT
PROCM_REAL: DOSREC
PROCM_REAL: DOSTIM
PROCM_REAL: EVTREC
PROCM_REAL: THETA
ROCM_INT:   MIXNUM,MIXEST
ROCM_INT:   NIREC,NDREC
ROCM_INT:   NREP,IREP
ROCM_REAL:  MIXP

$ERROR
CMNM1_INT:  NIND_7
NMPRD_INT:  COMACT,COMSAV
NMPRD_INT:  MSEC,MFIRST,IFIRSTEM
NMPRD_INT:  NEWL2
NMPRD_INT:  NPROB,IPROB
NMPRD_INT:  S1IT,S1NIT,S1NUM,S2IT,S2NIT,S2NUM
NMPRD_INT:  NTHES_,NETAS_,NEPSS_
NMPRD_INT:  MDVRES
NMPRD_INT:  NPDE_MODE
NMPRD_INT:  ETASXI
NMPRD_INT:  NOFIRSTDERCODE
NMPRD_REAL: CORRL2
NMPRD_REAL: ETA,EPS
NMPR_INT:   F_FLAG
NMPR_INT:   RPTI,RPTO,RPTON,PRDFL
NMPR_REAL:  CTLO
NMPR_REAL:  CTUP
NMPR_REAL:  YLO,YUP
NMPR_REAL:  THSIMP,OMSIMP,SGSIMP,THSIMPR
PKERR_REAL: MTIME
PRMOD_INT:  MTDIFF
PROCM_INT:  MNOW,MPAST,MNEXT
PROCM_INT:  NEWIND
PROCM_REAL: A
ROCM_INT:   LIREC
ROCM_INT:   MIXNUM,MIXEST
ROCM_INT:   NINDR,INDR1,INDR2
ROCM_INT:   NIREC,NDREC
ROCM_INT:   NREP,IREP
ROCM_REAL:  CNTID
ROCM_REAL:  CDEN_,DEN_
ROCM_REAL:  IIDX
ROCM_REAL:  MIXP
ROCM_REAL:  OMEGA
ROCM_REAL:  PR_CT
ROCM_REAL:  PR_Y
ROCM_REAL:  TEMPLT

$INFN
```

```
CMNM1_INT:  NIND_7
NMPRD_INT:  NEWL2
NMPRD_INT:  NPROB,IPROB
NMPRD_INT:  S1IT,S1NIT,S1NUM,S2IT,S2NIT,S2NUM
NMPRD_INT:  NWIND
NMPRD_INT:  PRED_IGNORE_DATA,PRED_IGNORE_DATA_TEST
NMPRD_REAL: CORRL2
NMPRD_REAL: ETA,EPS
NMPRD_REAL: PASSRC
NMPR_INT:   F_FLAG
NMPR_INT:   RPTI,RPTO,RPTON,PRDFL
NMPR_INT:   SKIP_
NMPR_REAL:  THSIMP,OMSIMP,SGSIMP,THSIMPR
PRMOD_INT:  I_SS,ISSMOD
ROCM_INT:   IERE,IERC
ROCM_INT:   LIREC
ROCM_INT:   MIXNUM,MIXEST
ROCM_INT:   NINDR,INDR1,INDR2
ROCM_INT:   NIREC,NDREC
ROCM_INT:   NREP,IREP
ROCM_REAL:  CDEN_,DEN_
ROCM_REAL:  CNTID
ROCM_REAL:  IIDX
ROCM_REAL:  MIXP
ROCM_REAL:  OBJECT
ROCM_REAL:  PR_CT
ROCM_REAL:  PR_Y
ROCM_REAL:  SETHET,SEOMEG,SESIGM,SETHETR
ROCM_REAL:  THETAF,OMEGAF,SIGMAF,THETAFR
ROCM_REAL:  PRED_,RES_,WRES_,NPRED_,NRES_,NWRES_
ROCM_REAL:  CPRED_,CRES_,CWRES_,PREDI_,RESI_,WRESI_,
ROCM_REAL:  CPREDI_,CRESI_,CWRESI_,EPRED_,ERES_,EWRES_
ROCM_REAL:  NPDE_,ECWRES_

$MIX
NMPRD_INT:  NPROB,IPROB
NMPRD_INT:  S1IT,S1NIT,S1NUM,S2IT,S2NIT,S2NUM
NMPRD_INT:  NTHES_,NETAS_,NEPSS_
ROCM_INT:   NINDR,INDR1,INDR2
ROCM_INT:   NIREC,NDREC
ROCM_INT:   NREP,IREP
ROCM_REAL:  TEMPLT
ROCM_REAL:  THETA,DATA

$PK
CMNM1_INT:  NIND_7
NMPRD_INT:  COMACT,COMSAV
NMPRD_INT:  MSEC,MFIRST,IFIRSTEM
NMPRD_INT:  NPROB,IPROB
NMPRD_INT:  S1IT,S1NIT,S1NUM,S2IT,S2NIT,S2NUM
```

```
NMPRD_INT:  NTHES_,NETAS_,NEPSS_
NMPRD_INT:  MDVRES
NMPRD_INT:  NPDE_MODE
NMPRD_INT:  ETASXI
NMPRD_INT:  NOFIRSTDERCODE
NMPRD_REAL: CORRL2
NMPRD_REAL: ETA,EPS
NMPR_INT:   RPTI,RPTO,RPTON,PRDFL
NMPR_REAL:  THSIMP,OMSIMP,SGSIMP,THSIMPR
PKERR_REAL: MTIME
PRMOD_INT:  I_SS,ISSMOD
PRMOD_INT:  MTDIFF
PRMOD_REAL: A_0
PROCM_INT:  A_0FLG
PROCM_INT:  MNOW,MPAST,MNEXT
PROCM_INT:  NEWIND
PROCM_REAL: A
PROCM_REAL: DOSREC
PROCM_REAL: DOSTIM
PROCM_REAL: TSTATE
ROCM_INT:   LIREC
ROCM_INT:   MIXNUM,MIXEST
ROCM_INT:   NINDR,INDR1,INDR2
ROCM_INT:   NIREC,NDREC
ROCM_INT:   NREP,IREP
ROCM_REAL:  CNTID
ROCM_REAL:  CDEN_,DEN_
ROCM_REAL:  IIDX
ROCM_REAL:  MIXP
ROCM_REAL:  OMEGA
ROCM_REAL:  PR_CT
ROCM_REAL:  PR_Y

$PRED
CMNM1_INT:  NIND_7
NMPRD_INT:  COMACT,COMSAV
NMPRD_INT:  I_REC
NMPRD_INT:  MSEC,MFIRST,IFIRSTEM
NMPRD_INT:  NEWL2
NMPRD_INT:  NPROB,IPROB
NMPRD_INT:  S1IT,S1NIT,S1NUM,S2IT,S2NIT,S2NUM
NMPRD_INT:  NTHES_,NETAS_,NEPSS_
NMPRD_INT:  MDVRES
NMPRD_INT:  NPDE_MODE
NMPRD_INT:  ETASXI
NMPRD_INT:  NOFIRSTDERCODE
NMPRD_INT:  PRED_IGNORE_DATA,PRED_IGNORE_DATA_TEST
NMPRD_REAL: CORRL2
NMPRD_REAL: ETA,EPS
NMPR_INT:   F_FLAG
```

```
NMPR_INT:   RPTI,RPTO,RPTON,PRDFL
NMPR_INT:   SKIP_
NMPR_REAL:  CTLO
NMPR_REAL:  CTUP
NMPR_REAL:  THSIMP,OMSIMP,SGSIMP,THSIMPR
NMPR_REAL:  YLO,YUP
ROCM_INT:   IERE,IERC
ROCM_INT:   LIREC
ROCM_INT:   MIXNUM,MIXEST
ROCM_INT:   NINDR,INDR1,INDR2
ROCM_INT:   NIREC,NDREC
ROCM_INT:   NREP,IREP
ROCM_REAL:  CNTID
ROCM_REAL:  CDEN_,DEN_
ROCM_REAL:  IIDX
ROCM_REAL:  MIXP
ROCM_REAL:  OBJECT
ROCM_REAL:  OMEGA
ROCM_REAL:  PR_CT
ROCM_REAL:  PR_Y
ROCM_REAL:  SETHET,SEOMEG,SESIGM,SETHETR
ROCM_REAL:  TEMPLT
ROCM_REAL:  THETAF,OMEGAF,SIGMAF,THETAFR
ROCM_REAL:  PRED_,RES_,WRES_,NPRED_,NRES_,NWRES_
ROCM_REAL:  CPRED_,CRES_,CWRES_,PREDI_,RESI_,WRESI_,
ROCM_REAL:  CPREDI_,CRESI_,CWRESI_,EPRED_,ERES_,EWRES_
ROCM_REAL:  NPDE_,ECWRES_
```

MODULES: VARIABLES
BLOCKS

```
CMNM1_INT:  NIND_7
            $ERROR $PK $PRED $INFN
NMPRD_INT:  COMACT,COMSAV
            $DES $AES $ERROR $PK $PRED
NMPRD_INT:  I_REC
            $PRED
NMPRD_INT:  MSEC,MFIRST,IFIRSTEM
            $DES $ERROR $PK $PRED
NMPRD_INT:  NEWL2
            $ERROR $PRED $INFN
NMPRD_INT:  NPROB,IPROB
            $DES $AES $ERROR $MIX $PK $PRED $INFN
NMPRD_INT:  NTHES_,NETAS_,NEPSS_
            $DES $AES $ERROR $MIX $PK $PRED
NMPRD_INT:  NWIND
            $INFN
NMPRD_INT:  S1IT,S1NIT,S1NUM,S2IT,S2NIT,S2NUM
            $DES $AES $ERROR $MIX $PK $PRED $INFN
NMPRD_REAL: CORRL2
            $ERROR $PK $PRED $INFN
NMPRD_REAL: ETA,EPS
```

```
            $ERROR $PK $PRED $INFN
NMPRD_REAL: PASSRC
            $INFN
NMPR_INT:   F_FLAG
            $ERROR $PRED $INFN
NMPR_INT:   RPTI,RPTO,RPTON,PRDFL
            $DES $AES $ERROR $PK $PRED $INFN
NMPR_INT:   SKIP_
            $PRED $INFN
NMPR_REAL:  CTLO
            $ERROR $PRED $ERROR $PRED
NMPR_REAL:  THSIMP,OMSIMP,SGSIMP,THSIMPR
            $PK $PRED $INFN
NMPR_REAL:  YLO,YUP
            $ERROR $PRED
NMPR_REAL:  THSIMP,OMSIMP,SGSIMP,THSIMPR
            $ERROR
PKERR_REAL: MTIME
            $DES $AES $ERROR $PK
PRMOD_INT:  ICALL,IDEFD,IDEFA
            $DES $AES
PRMOD_INT:  I_SS,ISSMOD
            $PK $INFN
PRMOD_INT:  MTDIFF
            $DES $AES $ERROR $PK
PRMOD_REAL: A_0
            $PK
PROCM_INT:  A_0FLG
            $PK
PROCM_INT:  ISFINL
            $DES $AES
PROCM_INT:  MNOW,MPAST,MNEXT
            $DES $AES $ERROR $PK
PROCM_INT:  NEWIND
            $ERROR $PK
PROCM_INT:  NVNT
            $DES $AES
PROCM_REAL: A
            $ERROR $PK
PROCM_REAL: DOSREC
            $DES $AES $PK
PROCM_REAL: DOSTIM
            $DES $AES $PK
PROCM_REAL: EVTREC
            $DES $AES
PROCM_REAL: THETA
            $DES $AES
PROCM_REAL: TSTATE
            $PK
ROCM_INT:   IERE,IERC
```

```
            $PRED $INFN
ROCM_INT:   LIREC
            $ERROR $PK $PRED $INFN
ROCM_INT:   MIXNUM,MIXEST
            $DES $AES $ERROR $PK $PRED $INFN
ROCM_INT:   NINDR,INDR1,INDR2
            $ERROR $MIX $PK $PRED $INFN
ROCM_INT:   NIREC,NDREC
            $DES $AES $ERROR $MIX $PK $PRED $INFN
ROCM_INT:   NREP,IREP
            $DES $AES $ERROR $MIX $PK $PRED $INFN
ROCM_REAL:  CDEN_,DEN_
            $ERROR $PK $PRED $INFN
ROCM_REAL:  CNTID
            $ERROR $PK $PRED $INFN
ROCM_REAL:  CPREDI_,CRESI_,CWRESI_,EPRED_,ERES_,EWRES_
            $PRED $INFN
ROCM_REAL:  CPRED_,CRES_,CWRES_,PREDI_,RESI_,WRESI_
            $PRED $INFN
ROCM_REAL:  IIDX
            $ERROR $PK $PRED $INFN
ROCM_REAL:  MIXP
            $DES $AES $ERROR $PK $PRED $INFN
ROCM_REAL:  NPDE_,ECWRES_
            $PRED $INFN
ROCM_REAL:  OBJECT
            $PRED $INFN
ROCM_REAL:  OMEGA
            $ERROR $PK $PRED
ROCM_REAL:  PRED_,RES_,WRES_,NPRED_,NRES_,NWRES_
            $PRED $INFN
ROCM_REAL:  PR_CT
            $ERROR $PK $PRED $INFN
ROCM_REAL:  PR_Y
            $ERROR $PK $PRED $INFN
ROCM_REAL:  SETHET,SEOMEG,SESIGM,SETHETR
            $PRED $INFN
ROCM_REAL:  TEMPLT
            $ERROR $MIX $PRED
ROCM_REAL:  THETA,DATA
            $MIX $PRED $INFN
```

NONMEM RESERVED VARIABLES

There are many other variables that are generally internal to NONMEM, and often are not needed by users except occasionally, which are not explicitly recognized by NMTRAN. Examples are ITER_REPORT,BAYES_EXTRA_REQUEST,BAYES_EXTRA in module NMBAYES_INT.

Such variables can be used in abbreviated code if they are listed in include files whose names start with NONMEM_RESERVED. See NONMEM_RESERVED_GENERAL in directory ..\util. It needs to be copied from util to the present run directory so that NM-

TRAN has access to it.

(See **abbreviated code**).
(See **Bayes Example 8**).

REFERENCES: Guide Introduction_7

VARIANCE-COVARIANCE

MEANING: Variance-covariance matrix
CONTEXT: NONMEM output

DISCUSSION:
NONMEM output refers to "VARIANCE-COVARIANCE" (or "COVARIANCE") matrices in three contexts:

OMEGA and SIGMA

OMEGA is the variance-covariance matrix for the first level random effects ETA. SIGMA is the variance-covariance matrix for the second-level random effects EPSILON.

Error messages referring to "VARIANCE-COVARIANCE COMPONENTS" arise from difficulties with the initial estimates of OMEGA and/or SIGMA, either those supplied by the user, or when no estimates are supplied, with those obtained in NONMEM's Initial Estimates Step. Initial estimates of both OMEGA and SIGMA must be positive definite.

With NONMEM 7.3 and higher, if the initial estimate of a block is not positive definite because of rounding errors, a value will be added to the diagonal elements to make it positive definite. A message in the NONMEM report file will indicate that this was done. E.g.,

```
DIAGONAL SHIFT OF  1.1000E-03 WAS IMPOSED TO ENSURE POS-
ITIVE DEFINITENESS
```

VAR-COV

Error messages referring to "VAR-COV" (in particular, "ESTIMATED TO BE SINGULAR" or "ESTIMATED TO BE 0") arise when the variance-covariance matrix for an individual's data is non-positive definite. For example, with the error model

```
Y=F+F*EPS(1)
```

predicted values for some observations (i.e. values of F) may be zero or close to 0. Then variances for these observations (which are proportional to F**2) are also zero, and this gives rise to such an error message.

COVARIANCE MATRIX OF ESTIMATE

This variance-covariance matrix refers to an estimate of the variability and covariability of the parameter estimates. It is computed in the Covariance Step, from the R and S matrices. An error message from the Covariance Step, stating that one of these two matrices is non-positive definite, indicates that the minimization procedure did not find a true or unique minimum (See **covariance**).

REFERENCES: Guide I, section C.3.5.2
REFERENCES: Guide IV, section III.B.10 , III.B.11
REFERENCES: Guide V, section 5.4 , 13.4.3

VERBATIM CODE

MEANING: FORTRAN statements
CONTEXT: Abbreviated code

USAGE:

```
"       verbatim line
```

DISCUSSION:
Verbatim code is FORTRAN code within a block of abbreviated code that is to be copied by NM-TRAN to the generated FORTRAN subroutine. It is not itself regarded as abbreviated code. Such code is marked by having the character " as the first non-blank character. The " is dropped and the characters to the right are copied as-is to columns 1-72 of the generated subroutine. Lines of verbatim code can include any FORTRAN statements: comment, declaration (e.g., INTEGER, COMMON, USE), I/O, CALL, assignment, continuation, DO, GOTO, etc.). Lower-case characters can be used freely, if the FORTRAN compiler permits them.

NM-TRAN does make some modifications to verbatim code, as follows:

Placement of code
: The verbatim code is adjusted as necessary to conform to FORTRAN column conventions. Alphabetic text that follows the initial ", or that follows a statement number, is moved to column 7, as required by FORTRAN, except for:

Comment lines
: If the character that immediately follows the initial " is !, this conforms to the FORTRAN 90 syntax for comment lines. The line is copied unchanged. If the character that immediately follows the initial " is C or c or " or *, this conforms to the FORTRAN 77 syntax for comment lines. The line is copied unchanged, but C or c or " or * is replaced by !. Example:

```
"! this is a comment
```

Continuation lines
: Fortran 77 continuation lines (non-blank in position 6) are not permitted with NONMEM 7. Instead, the line to be continued should end with character &. Example:

```
"       X=A &
"       +D/E
```

Replacement Rule
: In $PK, $ERROR, $DES, $AES, $INFN, and $PRED, in a line of abbreviated code labels of items in the data record are replaced by direct references to the data record itself (either DATREC or EVTREC, as appropriate). Thus, for example, if such a label occurs as the left-hand side of an assignment statement, then without this rule, when the statement is executed, the value of a local variable having the label is modified, whereas with the rule, an item in the data record is modified.

The character @, immediately before a label, can be used as an "escape" to prevent the label from being replaced. If the character @ immediately follows the initial ", none of the labels occuring in the line are replaced.

Labels occurring in common statements or as subroutine arguments are never replaced.

Low-value Character
: Tab characters (and other characters that are smaller than blank in the computer's

collating sequence, such as carriage return ^M) are permitted in verbatim code. With NONMEM 7, the last non-blank character on the line is replaced by a space if it is a low-value character. This permits DOS-type line endings ^M.

By default, all verbatim code goes in the main section of code and is called MAIN verbatim code. (The main section follows declarations and initial executable code inserted by NM-TRAN.) Within the main section, verbatim and abbreviated code may be freely mixed. Each line of verbatim code is positioned in the generated code after all code generated from the preceding line of abbreviated code. The user may explicitly specify a different location as follows.

FIRST verbatim code

Verbatim lines which must be positioned immediately after the declarations which are part of the normal subroutine header, and prior to the FIRST executable statement of the subroutine, must precede the first line of abbreviated code and must start with the line "FIRST.

MAIN verbatim code

FIRST verbatim code is normally terminated by the first line of abbreviated code. If there is both FIRST and MAIN verbatim code, and/or the main section is to start with verbatim code, the line "MAIN may be used to separate the FIRST and MAIN verbatim code.

LAST verbatim code

Verbatim lines which are to immediately precede the RETURN statement from the generated subroutine must follow the last line of abbreviated code and must be preceded by the line "LAST.

Example:

```
$ERROR
Y=F*(1+ERR(1))
"LAST
" PRINT *,HH(1,1)
```

This displays values after they have been assigned, immediately prior to the return.

If any lines of verbatim code are present in a block of abbreviated code, NM-TRAN generates USE statements appropriate to the kind of abbreviated code and allows variables undefined in abbreviated code to be used as right-hand quantities in abbreviated code.

Verbatim code is meant for use by expert users of NONMEM who are able to understand the generated FORTRAN subroutine and check that the verbatim code has the desired effect.

REFERENCES: Guide IV, section IV.I

WRITE FORMATS

MEANING: PREDPP read-only global variables

CONTEXT: User-supplied routines

USAGE:

```
USE PROCM_CHAR, ONLY: FMT
```

GLOBAL DECLARATION:

```
CHARACTER*80 FMT(3)
```

DISCUSSION:

`FMT(1)`
: A suitable format statement for `WRITE` statements when the full omega array is written.

`FMT(2)`
: A suitable format statement for `WRITE` statements when the full sigma array is written.

`FMT(3)`
: A suitable format statement for `WRITE` statements when the full theta array is written.

Location prior to NONMEM 7: procm8

REFERENCES: None.

WRITE PRINT

MEANING: FORTRAN statements
CONTEXT: Abbreviated code

SAMPLE:

```
WRITE (50,*) 99,ID,CL,THETA(1),ETA(1)

IF (ICALL.EQ.3) THEN
  WRITE (55,2) BIAS
ENDIF
```

DISCUSSION:

Certain forms of FORTRAN `WRITE`, `PRINT`, `OPEN`, `CLOSE`, and `REWIND` statements may be used anywhere in $PRED, $INFN, $PK, $ERROR, $DES, $AES, $AESINITIAL blocks.

`PRINT` list ...
`PRINT` *,list ...
`WRITE` (unit,format) list ...
`WRITE` (*,format) list ...
`WRITE` (unit,*) list ...
`WRITE` (*,*) list ...
`OPEN`(unit)
`OPEN`(unit,`FILE`=filename)
`CLOSE`(unit)
`REWIND`(unit)

Single or double quotes around the filename are optional. However, if the filename contains commas, semicolons, or parentheses, then it must be surrounded by single quotes ' or double quotes ". Filename may also contain equal signs if it is enclosed in quotes.

If the file is opened by NM-TRAN, filename may contain embedded spaces if it is enclosed in quotes, and may contain at most 80 characters. If the file is opened by NONMEM, filename may not contain embedded spaces, and may contain at most 71 characters.

unit

6, n or *.

6 indicates the unit connected to the NONMEM output file
n indicates the number of an alternative unit (40<n<2000).
* indicates a FORTRAN system-dependent output (with most systems, this is equivalent to using unit 6, but see the FORTRAN documentation) With OPEN, CLOSE and REWIND, the unit may not be 6 or *.

format

*, 1, 2, 991, or 992.

* indicates list-directed output. (FORTRAN library routines will select a format appropriate for the current run-time value of the quantity when displayed.)

1 or 991 requests the built-in format specification:
991 FORMAT (35F14.4)

2 or 992 requests the built-in format specification:
992 FORMAT (35E15.7)

(Format specification numbers 991 and 992 are used in generated code to avoid conflict with statement numbers arising from DO WHILE statements. The one digit versions are perhaps easier to remember.)

(No integer format specification is provided because constants are stored as floating point variables in generated and library routines.)

list A list of one or more right-hand quantities, i.e., variables or constants that may appear on the right in abbreviated code. May not include expressions or names of abbreviated functions. When subscripts are appropriate, these must be integer constants, declared integer variables, or expressions involving such constants and variables.

With a PRINT statement, a list may also include character constants. A character constant is delimited by single or double quotes (' or "). As with any Fortran character constant, a pair of adjacent delimiters within the constant represents a single character. E.g., PRINT *,'A isn''t B' will appear in the output as:
A isn't B

In an initialization or finalization block, a list may include elements of the OMEGA and SIGMA arrays. In a finalization block, a list may include elements of the standard error arrays (SETHET,SETHETR,SEOMEG and SESIGM).

Any array whose elements may be listed individually in a WRITE or PRINT statement may also be written in its entirety, by listing it without any subscripts. Specifically, one or more of the following may be included in a list:

```
THETA THETAFR SETHET SETHETR
 THSIMP ETA THSIMPR
OMEGA(BLOCK) OMEGA(DIAG) SEOMEG(BLOCK) SEOMEG(DIAG)
OMSIMP(BLOCK) OMSIMP(DIAG)
SIGMA(BLOCK) SIGMA(DIAG) SESIGM(BLOCK) SESIGM(DIAG)
SGSIMP(BLOCK) SGSIMP(DIAG)
IIDX CNTID IIDX,CNTID
```

Note that IIDX and CNTID are arrays in a module;
(See **Objective Function Value Individual**).
When "IIDX" or "CNTID" is listed, then only that array is written. When "IIDX,CNTID" is specified, then pairs of values are written, one pair per line, one pair for each individual record.

The option BLOCK requests that the entire array be written in full symmetric form. The option DIAG requests that only the diagonal elements be written. DIAG may also be coded DIAGONAL. Arrays of possibly different sizes (e.g., OMEGA and SIGMA) may not be listed together in the same WRITE statement. If a WRITE statement requests that an entire array be written, then the only other items that may be listed with the statement are other entire arrays.

When entire arrays are written, the format specification with a WRITE statement must be *. With either a WRITE or PRINT statement, an appropriate format is created by the NONMEM system. With these formats, elements of the standard error arrays that do not exist have the value 1E10 and are printed as 0.1000000E+11. (Such elements appear in NONMEM output as dots (.....).) If the covariance step fails or is not requested, all elements of the standard error arrays are 0 and are printed as 0.0000000E+00.

Example: To write the omega matrix and its standard errors in symmetric form code the following. (Only as many elements will be written as are appropriate for the dimension

of omega in the problem.)

```
IF (ICALL.EQ.3) WRITE (99,*) OMEGA(BLOCK),SEOMEG(BLOCK)
```

A list may also include a vector element or the entire vector (by listing the vector without a subscript).

The `OPEN`, `CLOSE`, and `REWIND` statements are part of the FORTRAN language. For details (i.e., the relationship of external files to units), see the Language Reference Manual and the Users Guide for your compiler.

REFERENCES: None.

XML COMPARE

MEANING: XML_COMPARE program
CONTEXT: NONMEM run

The utility program xml_compare compares the contents of two NONMEM report XML files that are produced by NONMEM.

USAGE:

```
xml_compare myresult1.xml myresult2.xml myprecision.xtl >mydifferences.txt
```

where delimiter is {, t s} for {comma tab space}, and myprecision.xtl is a precision specification or control file. Default delimiter is space and default control file is xml_compare.xtl. It is useful to redirect difference results to a file, in this example mydifferences.txt.

The control file can be quite elaborate, but it allows specification of various precision values for the many different types of values in the NONMEM report XML file, and to ignore certain entries as well. An example xml_compare.xtl file is in the util directory, and has the following contents:

```
$IGNORE
monitor
elapsed_time
datetime
covariance_status
termination_status
nonmem(version)
$PRECISION
GENERAL=0.2,0.2 OBJ_BAYES=2.0,0.0 OBJ_SAEM=0,100.0
OBJ_ITS=0,5.0 OBJ_IMP=0,10.0 OBJ_F=0,5.0
DIAG=0.3,0 OFFDIAG=0,0.5 COR=0.0,0.3 VAR=0.3,0.1 COV=-1.0
EIGENVALUES=2.0,0 OBJ_DIRECT=0,100.0
correlation_o=-1.0 INVCOVARIANCE_O=-1 INVCOVARIANCE_D=-1
etashrink=0,20 epsshrink=0,10
METHOD=DIRECT ALL=-1
METHOD=SAEM epsshrink=0,20
```

The $IGNORE record will ignore all elements that are listed, or just a specific attribute of an element, such as nonmem(version).

Under the $PRECISION record, a

```
GENERAL=R,A
```

can be given for most items, where relative is the relative tolerance, and absolute is the absolute tolerance. Following the GENERAL specification, tolerances may be specified for other items.

Two items of identical element and attributes are compared between the two files, where the equation for comparison is, between value X of xml file 1 and value Y of xml file 2,

ABS(X-Y)>R*MAX(ABS(X),ABS(Y))+A

The OBJ_BAYES is given a special test, as it has a standard deviation with it:

STD(X,Y)=SQRT(STD(X)2+ STD(Y)2)
ABS(X-Y)>R*STD(X,Y)+A

In the above example OBJ_BAYES=(2,0) means that if the Bayes objective functions in the two files differ by more than 2 standard deviations, then the difference is noted.

To ignore an item for comparison, specify -1. To specify an exact comparison, use 0,0. To refer to a particular optimization method, then enter METHOD=SAEM for example, and thereafter, all entries of items pertain to that estimation method, until METHOD is changed.

The METHOD attribute may have one of the following settings:

FOCE, ITS, IMP, SAEM, DIRECT, BAYES

For the total list of items, and their scope:

See **INTRODUCTION TO NONMEM 7, xml_compare Utility Program and its Use for Installation Qualification**

Because of the versatility of selecting which items are to be compared and with what precision, the xml_compare program can be used for batch processing installation qualification procedures, in comparing NONMEM results of a test run against a reference run. All results given in the standard NONMEM output file are also reported in the XML file.

REFERENCES: Guide Introduction_7

YLO YUP

MEANING: NONMEM-PRED global variables
CONTEXT: PRED routine

USAGE:

```
USE NMPR_REAL, ONLY: YLO,YUP
```

GLOBAL DECLARATION:

```
USE SIZES, ONLY: DPSIZE
REAL(KIND=DPSIZE) :: YLO,YUP
```

DISCUSSION:
If an observation is assumed to be normally distributed, with mean and variance specified for NONMEM as usual, then the likelihood for the observation is conditioned on the observation being in (or outside) an interval defined by the values YLO and YUP. If YLO is less than YUP, then the likelihood of the observation is conditioned on the observation being in the interval with lower bound YLO and upper bound YUP. If YLO is greater than YUP, then the likelihood of the observation is conditioned on the observation being outside the interval with lower bound YUP and upper bound YLO. These values may be set with the data record containing the observation. They may be set in $PRED and $ERROR abbreviated codes. If YLO or YUP is not set, it is assumed to be minus infinity or plus infinity, respectively. When YLO or YUP is set, the Laplacian estimation method must be used.

PR_Y is the estimated probability that an observation will fall within (or outside) the interval.

(See **pr_y**, **YLO_YUP_Probability:_PR_Y**).

Location prior to NONMEM 7: nmpr12

REFERENCES: None.

YLO YUP PROBABILITY: PR Y

MEANING: NONMEM read-only global variables
CONTEXT: User-supplied routines

USAGE:
```
USE ROCM_REAL, ONLY: PR_Y
```

GLOBAL DECLARATION:
```
USE SIZES, ONLY: DPSIZE
REAL(KIND=DPSIZE) :: PR_Y
```

DISCUSSION:

When with a given data record, either of the limits YLO or YUP are set so that during the analysis an interval is defined in which (or outside of which) an observation is conditioned to exist, then during a copying pass (and during ICALL=5 and 6), PR_Y is the estimated probability that an observation will fall within (or outside) the interval.
(See **YLO YUP**)
(See **copying_block**, **expectation**)
(See **data average**)

For the purpose of computing this probability, an actual observation need not exist in the record. If the mean and variance of the intra-individual model for a potential observation are specified, and if the limits YLO or YUP are set, a value of PR_Y will be computed, whether the record has MDV=0 or MDV=1. If neither YLO nor YUP are set, the value PR_Y will be 1.

`PR_Y` may be used as a right-hand quantity in abbreviated code for $PRED, $PK, $ERROR, and $INFN blocks.

Location prior to NONMEM 7: rocm44

REFERENCES: None.

EXAMPLES

ABBREVIATED FUNCTION EXAMPLES

Here is an example of the use and construction of a function of both time and the first-order elimination rate constant of a simple PK one-compartment model, to compute the amount in the system at a given time.
(See **abbreviated_function**).

```
$PRED
   ...
   VECTRA(1)=THETA(1)*EXP(ETA(1))
   VECTRA(2)=TIME
   ...
   A=DOSE*FUNCA(VECTRA)
   ...
```

With the $ABBREVIATED REPLACE feature, the function name can be changed to one that is perhaps easier to understand.

```
$ABBR REPLACE ONE_COMPARTMENT_LINEAR=FUNCA
   ...
$PRED
   ...
   VECTRA(1)=THETA(1)*EXP(ETA(1))
   VECTRA(2)=TIME
   ...
   A=DOSE*ONE_COMPARTMENT_LINEAR(VECTRA)
   ...
```

Either way, one needs to supply the function.

```
      FUNCTION FUNCA(X,X1,X2)
! Implements the One Compartment Linear model
      USE SIZES, ONLY: DPSIZE
      REAL(KIND=DPSIZE), INTENT(IN)     :: X
      REAL(KIND=DPSIZE), INTENT(IN OUT) :: X1,X2
      REAL(KIND=DPSIZE) :: FUNCA
      DIMENSION :: X(9),X1(9),X2(9,9)
      REAL(KIND=DPSIZE) :: EXPT
! THE FUNCTION ITSELF
      EXPT=EXP(-X(1)*X(2))
      FUNCA=EXPT
! 1ST. PARTIALS
      X1(1)=-EXPT*X(2)
      X1(2)=-EXPT*X(1)
! 2ND. PARTIALS
      X2(1,1)=EXPT*X(2)*X(2)
      X2(1,2)=EXPT*X(1)*X(2)
      X2(2,1)=EXPT*X(1)*X(2)
      X2(2,2)=EXPT*X(1)*X(1)
      RETURN
      END
```

Suppose the above code is in file ONE_COMPARTMENT_LINEAR.f90.
Then the control stream should also include

```
$SUBROUTINES OTHER=ONE_COMPARTMENT_LINEAR.f90
```

Multiple functions can be used with the $ABBREVIATED REPLACE feature. Suppose two functions are needed, called (for example) RED and GREEN. The control stream might contain the following fragment:

```
$SUBR OTHER=red.f90 OTHER=green.f90
 ...
$ABBR REPLACE RED=FUNCA
$ABBR REPLACE GREEN=FUNCB
 ...
$PRED
 ...
   VECTRA(1)=THETA(1)*EXP(ETA(1))
   VECTRA(2)=TIME
   A=DOSE*RED(VECTRA)
   B=DOSE*GREEN(VECTRA)
 ...
```

The file red.f90 contains code for FUNCA, and the file green.f90 contains code for FUNCB.

The $ABBREVIATED REPLACE feature could also be used to provide more meaningful names for reserved vectors. For example, suppose the arguments of the functions are different vectors.

```
$ABBR REPLACE RED=FUNCA, REDARG=VECTRA
$ABBR REPLACE GREEN=FUNCB, GREENARG=VECTRB
 ...
$PRED
 ...
   REDARG(1)=...
   REDARG(2)=...
   GREENARG(1)=...
   GREENARG(2)=...
   A=DOSE*RED(REDARG)
   B=DOSE*GREEN(GREENARG)
 ...
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 1

This is example1.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory.

```
;Model Desc: Two compartment Model, Using ADVAN3, TRANS4
;Project Name: nm7examples
;Project ID: NO PROJECT DESCRIPTION

$PROB RUN# Example 1 (from samp5l)
$INPUT C SET ID JID TIME  DV=CONC AMT=DOSE RATE EVID MDV CMT CLX
       V1X QX V2X SDIX SDSX
$DATA example1.csv IGNORE=C

$SUBROUTINES ADVAN3 TRANS4

$PK
; The thetas are MU modeled.
; Best that there is a linear relationship between THETAs and Mus
; The linear MU modeling of THETAS allows them to be efficiently
; Gibbs sampled.

MU_1=THETA(1)
MU_2=THETA(2)
MU_3=THETA(3)
MU_4=THETA(4)
CL=DEXP(MU_1+ETA(1))
V1=DEXP(MU_2+ETA(2))
Q=DEXP(MU_3+ETA(3))
V2=DEXP(MU_4+ETA(4))
S1=V1

$ERROR
Y = F + F*EPS(1)

; Initial values of THETA
$THETA
(0.001, 2.0) ;[LN(CL)]
(0.001, 2.0) ;[LN(V1)]
(0.001, 2.0) ;[LN(Q)]
(0.001, 2.0) ;[LN(V2)]

;INITIAL values of OMEGA
$OMEGA BLOCK(4)
0.15   ;[P]
0.01  ;[F]
0.15   ;[P]
0.01  ;[F]
0.01  ;[F]
0.15   ;[P]
```

```
0.01  ;[F]
0.01  ;[F]
0.01  ;[F]
0.15   ;[P]

;Initial value of SIGMA
$SIGMA
(0.6 )   ;[P]

;Prior information is important for MCMC Bayesian analysis,
;not necessary for maximization methods
;Note the syntax used for defining priors that is available
;as of NONMEM 7.3
$PRIOR NWPRI

; Prior information of THETAS
$THETAP (2.0 FIX)X4

; Variance to prior information of THETAS.
; Because variances are very large, this means that the prior
; information to the THETAS is highly uninformative.
$THETAPV BLOCK(4) FIX VALUES(10000,0.0)

; Prior information to the OMEGAS.
$OMEGAP BLOCK(4) FIX VALUES(0.2,0.0)
; Degrees of freedom to prior OMEGA matrix.
; Because degrees of freedom is very low, equal to the
; the dimension of the prior OMEGA, this means that the
; prior information to the OMEGAS is highly uninformative
$OMEGAPD (4 FIX)

; Prior information to the SIGMAS
$SIGMAP 0.06 FIX
; Degrees of freedom to prior SIGMA matrix.
; Because degrees of freedom is very low, equal to the
; the dimension of the prior SIGMA, this means that the
; prior information to the SIGMA is highly uninformative
$SIGMAPD (1 FIX)

; The first analysis is iterative two-stage,
; maximum of 500 iterations (NITER), iteration results
; are printed every 5 iterations, gradient precision (SIGL) is 4.
; Termination is tested on all of
; the population parameters (CTYPE=3),
; and for less then 2 significant digits change (NSIG).
; Prior information is not necessary for ITS, so NOPRIOR=1.
; The intermediate and final results of the ITS method will be
; recoded in row/column format in example1.ext

$EST METHOD=ITS MAPITER=0 INTERACTION FILE=example1.ext NITER=500
```

```
     PRINT=5 NOABORT SIGL=4 CTYPE=3 CITER=10
     CALPHA=0.05 NOPRIOR=1 NSIG=2

; The results of ITS are used as the initial values for the
; SAEM method. A maximum of 3000 ; stochastic iterations (NBURN)
; is requested, but may end early if statistical test determines
; that variations in all parameters is stationary
; (note that any settings from the previous $EST
; carries over to the next $EST statement, within a $PROB).
; The SAEM is a Monte Carlo process,
; so setting the SEED assures repeatability of results.
; Each iteration obtains only 2 Monte Carlo samples ISAMPLE),
;  so they are very fast.
; But many iterations are needed, so PRINT only
; every 100th iteration.
; After the stochastic phase, 500 accumulation iterations will be
; Performed (NITER), to obtain good parameters estimates with
; little stochastic noise.
; As a new FILE has not been given, the SAEM results will append to
; example1.ext.

$EST METHOD=SAEM INTERACTION NBURN=3000 NITER=500 PRINT=100
     SEED=1556678 ISAMPLE=2

; After the SAEM method, obtain good estimates of the marginal
; density (objective function),
; along with good estimates of the standard errors.
; This is best done with importance sampling ; (IMP),
; performing the expectation step only (EONLY=1), so that
; final population parameters remain at the final SAEM result.
; Five iterations (NITER) should allow the importance sampling
; proposal density to become stationary.
; This is observed by the objective function settling
; to a particular value (with some stochastic noise).
; By using 3000 Monte Carlo samples
; (ISAMPLE), this assures a precise assessment of standard errors.

$EST METHOD=IMP  INTERACTION EONLY=1 NITER=5 ISAMPLE=3000 PRINT=1
     SIGL=8 NOPRIOR=1

; The Bayesian analysis is performed.
; While 10000 burn-in iterations are requested as a maximum,
; because the termination test is on (CTYPE<>0, set at the
; first $EST statement), and because the initial parameters are at
; the SAEM result, which is the maximum likelihood position,
; the analysis should settle down to a stationary distribution in
; several hundred iterations.
; Prior information is also used to facilitate Bayesian analysis.
; The individual Bayesian iteration results are important,
; and may be need for post-processing analysis.
```

```
; So specify a separate FILE for the Bayesian analysis.

$EST METHOD=BAYES INTERACTION FILE=example1.txt NBURN=10000
     NITER=10000 PRINT=100 NOPRIOR=0

; Just for old-times sake, let's see what the traditional
; FOCE method will give us.
; And, remember to introduce a new FILE, so its results won't
; append to our Bayesian FILE.
; Appending to example1.ext with the EM methods is fine.

$EST METHOD=COND INTERACTION MAXEVAL=9999 NSIG=3 SIGL=10
     PRINT=5 NOABORT NOPRIOR=1
     FILE=example1.ext

; Time for the standard error results.
; You may request a more precise gradient precision (SIGL)
; that differed from that used during estimation.

$COV MATRIX=R PRINT=E UNCONDITIONAL SIGL=12

; Print out results in tables. Include some of the new weighted
; residual types

$TABLE ID TIME PRED RES WRES CPRED CWRES EPRED ERES EWRES NOAPPEND
       ONEHEADER FILE=example1.TAB NOPRINT
$TABLE ID CL V1 Q V2 FIRSTONLY NOAPPEND NOPRINT FILE=example1.PAR
$TABLE ID ETA1 ETA2 ETA3 ETA4 FIRSTONLY NOAPPEND
        NOPRINT FILE=example1.ETA
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 1B

This is example1B.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory.

```
;Model Desc: Two compartment Model, Using ADVAN3, TRANS4
;Project Name: nm7examples
;Project ID: NO PROJECT DESCRIPTION

$PROB RUN# Example 1B (from samp5l)
$INPUT C SET ID JID TIME  DV=CONC AMT=DOSE RATE EVID MDV CMT CLX
       V1X QX V2X SDIX SDSX
$DATA example1b.csv IGNORE=C

$SUBROUTINES ADVAN3 TRANS4

$PK
; The thetas are MU modeled.
; Best that there is a linear relationship between THETAs and Mus
; The linear MU modeling of THETAS allows them to be efficiently
; Gibbs sampled.

MU_1=THETA(1)
MU_2=THETA(2)
MU_3=THETA(3)
MU_4=THETA(4)
CL=DEXP(MU_1+ETA(1))
V1=DEXP(MU_2+ETA(2))
Q=DEXP(MU_3+ETA(3))
V2=DEXP(MU_4+ETA(4))
S1=V1

$ERROR
Y = F + F*EPS(1)

; Initial values of THETA
$THETA
(0.001, 2.0) ;[LN(CL)]
(0.001, 2.0) ;[LN(V1)]
(0.001, 2.0) ;[LN(Q)]
(0.001, 2.0) ;[LN(V2)]

;INITIAL values of OMEGA
$OMEGA BLOCK(4)
0.15   ;[P]
0.01  ;[F]
0.15   ;[P]
0.01  ;[F]
0.01  ;[F]
0.15   ;[P]
```

```
0.01  ;[F]
0.01  ;[F]
0.01  ;[F]
0.15   ;[P]

;Initial value of SIGMA
$SIGMA
(0.6 )   ;[P]

;NTHETA=number of Thetas to be estimated
;NETA=number of Etas to be estimated
;  (and to be described by NETAxBETA OMEGA matrix)
;NTHP=number of thetas which have a prior
;NETP=number of Omegas with prior
;Prior information is important for MCMC Bayesian analysis,
; not necessary for maximization methods
$PRIOR NWPRI NTHETA=4, NETA=4, NTHP=4, NETP=4

; Prior information of THETAS
$THETAP (2.0 FIX) (2.0 FIX) (2.0 FIX) (2.0 FIX)

; Variance to prior information of THETAS.
; Because variances are very large, this
; means that prior information to the THETAS is highly uninformative.

$THETAPV BLOCK(4)
10000 FIX
0.00 10000
0.00  0.00 10000
0.00  0.00 0.0 10000

; Prior information to the OMEGAS.

$OMEGAP BLOCK(4)
0.2 FIX
0.0  0.2
0.0  0.0 0.2
0.0  0.0 0.0 0.2

; Degrees of freedom to prior OMEGA matrix.
; Because degrees of freedom is very low, equal to the
; the dimension of the prior OMEGA, this means that the
; prior information to the OMEGAS is highly uninformative

$OMEGAPD (4 FIX)

; The first analysis is iterative two-stage,
; maximum of 500 iterations (NITER), iteration results
; are printed every 5 iterations, gradient precision (SIGL) is 4.
; Termination is tested on all of
```

```
; the population parameters (CTYPE=3),
; and for less then 2 significant digits change (NSIG).
; Prior information is not necessary for ITS, so NOPRIOR=1.
; The intermediate and final results
; of the ITS method will be recoded in row/column format in
; example1b.ext

$EST METHOD=ITS MAPITER=0 INTERACTION FILE=example1b.ext NITER=500
     PRINT=5 CINTERVAL=5 NOABORT SIGL=4 CTYPE=3 CITER=10
     CALPHA=0.05 NOPRIOR=1 NSIG=2

; The results of ITS are used as the initial values for the SAEM
; method.  A maximum of 3000 ; stochastic iterations (NBURN) is
; requested, but may end early if statistical test determines
; that variations in all parameters is stationary
; (note that any settings from the previous $EST
; carries over to the next $EST statement, within a $PROB).
; The SAEM is a Monte Carlo process,
; so setting the SEED assures repeatability of results.
; Each iteration obtains only 2 Monte Carlo samples ISAMPLE),
;  so they are very fast.
; But many iterations are needed, so PRINT only
; every 100th iteration.
; After the stochastic phase, 500 accumulation iterations will be
; Performed (NITER), to obtain good parameters estimates with
; little stochastic noise.
; As a new FILE has not been given, the SAEM results will append to
; example1b.ext.

$EST METHOD=SAEM INTERACTION NBURN=3000 NITER=500
     PRINT=250 CINTERVAL=100
     SEED=1556678 ISAMPLE=2

; After the SAEM method, obtain good estimates of the marginal
; density (objective function),
; along with good estimates of the standard errors.
; This is best done with importance sampling ; (IMP),
; performing the expectation step only (EONLY=1), so that
; final population parameters remain at the final SAEM result.
; Five iterations (NITER) should allow the importance sampling
; proposal density to become stationary.
; This is observed by the objective function settling
; to a particular value (with some stochastic noise).
; By using 3000 Monte Carlo samples
; (ISAMPLE), this assures a precise assessment of standard errors.

 $EST METHOD=IMP  INTERACTION EONLY=1  MAPITER=0 NITER=5
  ISAMPLE=3000 PRINT=1 SIGL=8 NOPRIOR=1
 $EST METHOD=DIRECT INTERACTION EONLY=1 NITER=3 ISAMPLE=3000
  PRINT=1 SIGL=8 NOPRIOR=1
```

```
; The Bayesian analysis is performed.
; While 10000 burn-in iterations are requested as a maximum,
; because the termination test is on (CTYPE<>0, set at the
; first $EST statement), and because the initial parameters are at
; the SAEM result, which is the maximum likelihood position,
; the analysis should settle down to a stationary distribution in
; several hundred iterations.
; Prior information is also used to facilitate Bayesian analysis.
; The individual Bayesian iteration results are important,
; and may be need for post-processing analysis.
; So specify a separate FILE for the Bayesian analysis.

$EST METHOD=BAYES INTERACTION FILE=example1b.txt NBURN=10000
     NITER=10000 PRINT=1000 CINTERVAL=100 NOPRIOR=0

; Just for old-times sake, let's see what the traditional
; FOCE method will give us.
; And, remember to introduce a new FILE, so its results won't
; append to our Bayesian FILE.
; Appending to example1b.ext with the EM methods is fine.

$EST METHOD=COND INTERACTION MAXEVAL=9999 NSIG=3 SIGL=10
     PRINT=5 NOABORT NOPRIOR=1
     FILE=example1b.ext

; Time for the standard error results.
; You may request a more precise gradient precision (SIGL)
; that differed from that used during estimation.

$COV MATRIX=R PRINT=E UNCONDITIONAL SIGL=12

; Print out results in tables. Include some of the new weighted
; residual types

$TABLE ID TIME PRED RES WRES CPRED CWRES EPRED ERES EWRES NOAPPEND
       ONEHEADER FILE=example1b.TAB NOPRINT
$TABLE ID CL V1 Q V2 FIRSTONLY NOAPPEND NOPRINT FILE=example1b.PAR
       FORMAT=,1PE15.8
$TABLE ID ETA1 ETA2 ETA3 ETA4 FIRSTONLY NOAPPEND NOPRINT
       FILE=example1b.ETA FORMAT=,1PE15.8
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 2

This is example2.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory.

```
;Model Desc: Two Compartment model with Clearance and
; central volume modeled with covariates age and gender
;Project Name: nm7examples
;Project ID: NO PROJECT DESCRIPTION

$PROB RUN# example2 (from sampc)
$INPUT C SET ID JID TIME DV=CONC AMT=DOSE RATE EVID MDV CMT GNDR AGE
$DATA example2.csv IGNORE=C
$SUBROUTINES ADVAN3 TRANS4

$PK
; LCLM=log transformed clearance, male
LCLM=THETA(1)
;LCLF=log transformed clearance, female.
LCLF=THETA(2)
; CLAM=CL age slope, male
CLAM=THETA(3)
; CLAF=CL age slope, female
CLAF=THETA(4)
; LV1M=log transformed V1, male
LV1M=THETA(5)
; LV1F=log transformed V1, female
LV1F=THETA(6)
; V1AM=V1 age slope, male
V1AM=THETA(7)
; V1AF=V1 age slope, female
V1AF=THETA(8)
; LAGE=log transformed age
LAGE=DLOG(AGE)

;Mean of ETA1, the inter-subject deviation of Clearance,
; is ultimately modeled as linear function of THETA(1) to THETA(4).
; Relating thetas to Mus by linear functions is not essential for
; ITS, IMP, or IMPMAP methods, but is very helpful for MCMC methods
; such as SAEM and BAYES.

MU_1=(1.0-GNDR)*(LCLM+LAGE*CLAM) + GNDR*(LCLF+LAGE*CLAF)

; Mean of ETA2, the inter-subject deviation of V1,
; is ultimately modeled as linear function of THETA(5) to THETA(8)

MU_2=(1.0-GNDR)*(LV1M+LAGE*V1AM) + GNDR*(LV1F+LAGE*V1AF)
MU_3=THETA(9)
MU_4=THETA(10)
CL=DEXP(MU_1+ETA(1))
```

```
V1=DEXP(MU_2+ETA(2))
Q=DEXP(MU_3+ETA(3))
V2=DEXP(MU_4+ETA(4))
S1=V1

$ERROR
CALLFL=0
; Option to model the residual error coefficient in THETA(11),
; rather than in SIGMA.
SDSL=THETA(11)
W=F*SDSL
Y = F + W*EPS(1)
IPRED=F
IWRES=(DV-F)/W

;Initial THETAs
$THETA
( 0.7 ) ;[LCLM]
( 0.7 ) ;[LCLF]
( 2 )   ;[CLAM]
( 2.0);[CLAF]
( 0.7 ) ;[LV1M]
( 0.7 ) ;[LV1F]
( 2.0 )   ;[V1AM]
( 2.0 )   ;[V1AF]
( 0.7 ) ;[MU_3]
(  0.7 );[MU_4]
( 0.3 )     ;[SDSL]

;Initial OMEGAs
$OMEGA BLOCK(4)
0.5  ;[p]
0.001  ;[f]
0.5  ;[p]
0.001 ;[f]
0.001 ;[f]
0.5  ;[p]
0.001 ;[f]
0.001 ;[f]
0.001 ;[f]
0.5 ;[p]

; SIGMA is 1.0 fixed, serves as unscaled variance for EPS(1).
; THETA(11) takes up the residual error scaling.
$SIGMA
(1.0 FIXED)

;Prior information is important for MCMC Bayesian analysis,
; not necessary for maximization methods
; In this example, only the OMEGAs have a prior distribution,
```

```
; the THETAS do not.
; For Bayesian methods, it is most important for at least the
; OMEGAs to have a prior, even an uninformative one,
; to stabilize the analysis. Only if the number of subjects
; exceeds the OMEGA dimension number by at least 100,
; then you may get away without priors on OMEGA for BAYES analysis.
$PRIOR NWPRI
; Prior OMEGA matrix
$OMEGAP BLOCK(4) FIX VALUES(0.01,0.0)
; Degrees of freedom to OMEGA prior matrix:
$OMEGAPD 4 FIX

; The first analysis is iterative two-stage.
; Note that the GRD specification is THETA(11) is a
; Sigma-like parameter.  This will allow NONMEM to make
; efficient gradient evaluations for THETA(11), which is useful
; for later IMP,IMPMAP, and SAEM methods, but has no impact on
; ITS and BAYES methods.

$EST METHOD=ITS INTERACTION FILE=example2.ext NITER=1000 NSIG=2
     PRINT=5 NOABORT SIGL=8 NOPRIOR=1 CTYPE=3 GRD=TS(11)

; Results of ITS serve as initial parameters for the IMP method.

$EST METHOD=IMP INTERACTION EONLY=0 MAPITER=0 NITER=100 ISAMPLE=300
     PRINT=1 SIGL=8

; The results of IMP are used as the initial values for the SAEM method.

$EST METHOD=SAEM NBURN=3000 NITER=2000 PRINT=10 ISAMPLE=2
     CTYPE=3 CITER=10 CALPHA=0.05

; After the SAEM method, obtain good estimates of the marginal density
; (objective function),
; along with good estimates of the standard errors.

$EST METHOD=IMP INTERACTION EONLY=1 NITER=5 ISAMPLE=3000
     PRINT=1 SIGL=8 SEED=123334
     CTYPE=3 CITER=10 CALPHA=0.05

; The Bayesian analysis is performed.

$EST METHOD=BAYES INTERACTION FILE=example2.TXT NBURN=10000
     NITER=3000 PRINT=100 NOPRIOR=0
     CTYPE=3 CITER=10 CALPHA=0.05

; Just for old-times sake, lets see what the traditional
; FOCE method will give us.
; And, remember to introduce a new FILE, so its results wont
; append to our Bayesian FILE.
```

```
$EST  METHOD=COND INTERACTION MAXEVAL=9999 FILE=example2.ext NSIG=2
  SIGL=14 PRINT=5 NOABORT NOPRIOR=1

$COV MATRIX=R UNCONDITIONAL
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 3

This is example3.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory.

```
;Model Desc: Population Mixture Problem in 1 Compartment model,
; with Volume and rate constant parameters and their inter-subject
; variances modeled from two sub-populations
;Project Name: nm7examples
;Project ID: NO PROJECT DESCRIPTION

$PROB RUN# example3 (from ad1tr1m2s)
$INPUT C SET ID JID TIME CONC=DV DOSE=AMT RATE EVID MDV CMT VC1 K101
   VC2 K102 SIGZ PROB
$DATA example3.csv IGNORE=C

$SUBROUTINES ADVAN1 TRANS1

; The mixture model uses THETA(5) as the mixture proportion parameter,
; defining the proportion of subjects in sub-population 1 (P(1),
; and in sub-population 2 (P(2)

$MIX
P(1)=THETA(5)
P(2)=1.0-THETA(5)
NSPOP=2


$PK
;  The MUs should always be unconditionally defined, that is,
;  they should never be defined in IF/THEN blocks
; THETA(1) models the Volume of sub-population 1
MU_1=THETA(1)
; THETA(2) models the clearance of sub-population 1
MU_2=THETA(2)
; THETA(3) models the Volume of sub-population 2
MU_3=THETA(3)
; THETA(4) models the clearance of sub-population 2
MU_4=THETA(4)
VCM=DEXP(MU_1+ETA(1))
K10M=DEXP(MU_2+ETA(2))
VCF=DEXP(MU_3+ETA(3))
K10F=DEXP(MU_4+ETA(4))
Q=1
IF(MIXNUM.EQ.2) Q=0
V=Q*VCM+(1.0-Q)*VCF
K=Q*K10M+(1.0-Q)*K10F
S1=V

$ERROR
```

```
Y = F + F*EPS(1)

; Initial THETAs
$THETA
(-1000.0  4.3 1000.0) ;[MU_1]
(-1000.0 -2.9 1000.0) ;[MU_2]
(-1000.0 4.3 1000.0)  ;[MU_3]
(-1000.0 -0.67 1000.0) ;[MU_4]
(0.0001 0.667 0.9999)   ;[P(1)]

;Initial OMEGA block 1, for sub-population 1
$OMEGA BLOCK(2)
 .04 ;[p]
 .01 ; [f]
 .027; [p]

;Initial OMEGA block 2, for sub-population 2
$OMEGA BLOCK(2)
 .05; [p]
 .01; [f]
 .06; [p]

$SIGMA
0.01 ;[p]

; Prior information setup for OMEGAS only
$PRIOR NWPRI

; Prior OMEGA block 1.  Note that because the OMEGA is separated
; into blocks, so their priors should have the same block design.

$OMEGAP BLOCK(2)
 0.05 FIX
 0.0 0.05

; Prior OMEGA block 2

$OMEGAP BLOCK(2)
0.05 FIX
0.0 0.05

; Degrees of Freedom defined for Priors.
; One for each OMEGA block defining each sub-popluation
$OMEGAPD (2 FIX) (2 FIX)


$EST METHOD=ITS INTERACTION NITER=20 PRINT=1 NOABORT SIGL=8
     FILE=example3.ext CTYPE=3 CITER=10
     CALPHA=0.05 NOPRIOR=1
```

```
$EST NBURN=500 NITER=500 METHOD=SAEM INTERACTION PRINT=10 SIGL=6
     ISAMPLE=2

$EST METHOD=IMP INTERACTION NITER=5 ISAMPLE=1000 PRINT=1 NOABORT
     SIGL=6 EONLY=1 MAPITER=0

$EST METHOD=BAYES INTERACTION NBURN=2000 NITER=1000 PRINT=10
     FILE=example3.txt SIGL=8 NOPRIOR=0

$EST MAXEVAL=9999 NSIG=3 SIGL=12 PRINT=1 FILE=example3.ext
     METHOD=CONDITIONAL INTERACTION NOABORT
     NOPRIOR=1

$COV MATRIX=R UNCONDITIONAL
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 4

This is example4.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory.

```
;Model Desc: Population Mixture Problem in 1 Compartment model,
; with rate constant parameter and its inter-subject variances
; modeled as coming from two sub-populations
;Project Name: nm7examples
;Project ID: NO PROJECT DESCRIPTION

$PROB RUN# example4 (from ad1tr1m2t)
$INPUT C SET ID JID TIME CONC=DV DOSE=AMT RATE EVID MDV CMT VC1
       K101 VC2 K102 SIGZ PROB
$DATA example4.csv IGNORE=C

$SUBROUTINES ADVAN1 TRANS1

$MIX
P(1)=THETA(4)
P(2)=1.0-THETA(4)
NSPOP=2


$PK
MU_1=THETA(1)
MU_2=THETA(2)
MU_3=THETA(3)
V=DEXP(MU_1+ETA(1))
K10M=DEXP(MU_2+ETA(2))
K10F=DEXP(MU_3+ETA(3))
Q=1
IF(MIXNUM.EQ.2) Q=0
K=Q*K10M+(1.0-Q)*K10F
S1=V

$ERROR
Y = F + F*EPS(1)

$THETA
(-1000.0  4.3 1000.0) ;[MU_1]
(-1000.0 -2.9 1000.0) ;[MU_2]
(-1000.0 -0.67 1000.0) ;[MU_3]
(0.0001 0.667 0.9999)   ;[P(1)]


$OMEGA BLOCK(3)
 .04  ;[p]
 0.01  ;[f]
 .027 ;[p]
```

```
 0.01  ;[f]
 0.001 ;[f]
 0.06 ;[p]

$SIGMA
 0.01 ;[p]

; Prior information setup for OMEGAS only
$PRIOR NWPRI

; Prior OMEGA
$OMEGAP BLOCK(3)
 0.05 FIX
 0.0 0.05
 0.0 0.0  0.05

; Degrees of Freedom defined for Priors.
$OMEGAPD (3 FIX)

$EST METHOD=ITS INTERACTION NITER=30 PRINT=5 NOABORT SIGL=6
     FILE=example4.ext NOPRIOR=1 CTYPE=3 CITER=10 CALPHA=0.05

$EST METHOD=IMP INTERACTION NITER=20 ISAMPLE=300 PRINT=1
     NOABORT SIGL=6 NOPRIOR=1

$EST NBURN=500 NITER=500 METHOD=SAEM INTERACTION PRINT=10 SIGL=6
     ISAMPLE=2 NOPRIOR=1 MAPITER=0

$EST METHOD=IMP INTERACTION EONLY=1 NITER=20 ISAMPLE=3000 PRINT=1
     NOABORT SIGL=6 NOPRIOR=1

$EST METHOD=BAYES INTERACTION NBURN=2000 NITER=5000 PRINT=10
     FILE=example4.txt SIGL=6 NOPRIOR=0

$EST MAXEVAL=9999 NSIG=3 SIGL=12 PRINT=1
     METHOD=CONDITIONAL INTERACTION
     NOABORT FILE=example4.ext NOPRIOR=1

$COV MATRIX=R UNCONDITIONAL SIGL=10
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 5

This is example5.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory.

```
;Model Desc: Population Mixture Problem in 1 Compartment model,
; with rate constant parameter mean modeled for two
; sub-populations, but its inter-subject variance is the same in
; both sub-populations Project Name: nm7examples
;Project ID: NO PROJECT DESCRIPTION

$PROB RUN# example5 (from ad1tr1m4t)
$INPUT C SET ID JID TIME CONC=DV DOSE=AMT RATE EVID MDV CMT VC1
       K101 VC2 K102 SIGZ PROB
$DATA example5.csv IGNORE=C

$SUBROUTINES ADVAN1 TRANS1

$MIX
P(1)=THETA(4)
P(2)=1.0-THETA(4)
NSPOP=2


$PK
Q=1
IF(MIXNUM.EQ.2) Q=0
MU_1=THETA(1)

; Note that MU_2 can be modeled as THETA(2) or THETA(3),
; depending on the MIXNUM value.
; Also, we are avoiding IF/THEN blocks.

MU_2=Q*THETA(2)+(1.0-Q)*THETA(3)
V=DEXP(MU_1+ETA(1))
K=DEXP(MU_2+ETA(2))
S1=V

$ERROR
Y = F + F*EPS(1)

$THETA
(-1000.0  4.3 1000.0)  ;[MU_1]
(-1000.0 -2.9 1000.0)  ;[MU_2-1]
(-1000.0 -0.67 1000.0) ;[MU_2-2]
(0.0001 0.667 0.9999)  ;[P(1)]

$OMEGA BLOCK(2)
0.04 ;[p]
0.01  ;[f]
```

```
0.04 ;[p]

$SIGMA
0.01 ;[p]

$EST METHOD=ITS INTERACTION NITER=100 PRINT=1 NOABORT SIGL=8
     FILE=example5.ext CTYPE=3

$EST METHOD=IMPMAP INTERACTION NITER=20 ISAMPLE=300 PRINT=1
     NOABORT SIGL=8

$EST METHOD=IMP INTERACTION NITER=20 ISAMPLE=1000 PRINT=1
     NOABORT SIGL=6

$EST NBURN=500 NITER=500 METHOD=SAEM INTERACTION PRINT=10
     SIGL=6 ISAMPLE=2

$EST METHOD=IMP INTERACTION NITER=5 ISAMPLE=1000 PRINT=1
     NOABORT SIGL=6 EONLY=1 MAPITER=0

$EST METHOD=BAYES INTERACTION NBURN=2000 NITER=5000 PRINT=10
     FILE=example5.txt SIGL=8

$EST MAXEVAL=9999 NSIG=2 SIGL=8 PRINT=10 FILE=example5.ext
     METHOD=CONDITIONAL INTERACTION NOABORT

$COV MATRIX=R
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 6

This is example6.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory.

```
;Model Desc: Receptor Mediated Clearance model with Dynamic Change
;            in Receptors
;Project Name: nm7examples
;Project ID: NO PROJECT DESCRIPTION

$PROB RUN# example6 (from r2compl)
$INPUT C SET ID JID TIME DV=CONC DOSE=AMT RATE EVID MDV CMT
$DATA example6.csv IGNORE=C

; The new numerical integration solver is used, although ADVAN=9
; is also efficient for this problem.

$SUBROUTINES ADVAN13 TRANS1 TOL=4
$MODEL NCOMPARTMENTS=3

$PK
MU_1=THETA(1)
MU_2=THETA(2)
MU_3=THETA(3)
MU_4=THETA(4)
MU_5=THETA(5)
MU_6=THETA(6)
MU_7=THETA(7)
MU_8=THETA(8)
VC=EXP(MU_1+ETA(1))
K10=EXP(MU_2+ETA(2))
K12=EXP(MU_3+ETA(3))
K21=EXP(MU_4+ETA(4))
VM=EXP(MU_5+ETA(5))
KMC=EXP(MU_6+ETA(6))
K03=EXP(MU_7+ETA(7))
K30=EXP(MU_8+ETA(8))
S3=VC
S1=VC
KM=KMC*S1
F3=K03/K30

$DES
DADT(1) = -(K10+K12)*A(1) + K21*A(2) - VM*A(1)*A(3)/(A(1)+KM)
DADT(2) = K12*A(1) - K21*A(2)
DADT(3) =  -VM*A(1)*A(3)/(A(1)+KM) - K30*A(3) + K03

$ERROR
CALLFL=0
ETYPE=1
```

```
IF(CMT.NE.1) ETYPE=0
IPRED=F
Y = F + F*ETYPE*EPS(1) + F*(1.0-ETYPE)*EPS(2)


$THETA
;Initial Thetas
( 4.0 )  ;[MU_1]
( -2.1 ) ;[MU_2]
( 0.7 )  ;[MU_3]
( -0.17 );[MU_4]
( 2.2 ) ;[MU_5]
( 0.14 )  ;[MU_6]
( 3.7 )  ;[MU_7]
( -0.7) ;[MU_8]


;Initial Omegas
$OMEGA BLOCK(8)
0.2 ;[p]
-0.0043  ;[f]
0.2 ;[p]
0.0048   ;[f]
-0.0023  ;[f]
0.2 ;[p]
0.0032   ;[f]
0.0059   ;[f]
-0.0014  ;[f]
0.2 ;[p]
0.0029   ;[f]
0.0027 ;[f]
-0.00026 ;[f]
-0.0032  ;[f]
0.2 ;[p]
-0.0025  ;[f]
0.00097  ;[f]
0.0024   ;[f]
0.00197  ;[f]
-0.0080  ;[f]
0.2 ;[p]
0.0031   ;[f]
-0.00571 ;[f]
0.0030   ;[f]
-0.0074  ;[f]
0.0025   ;[f]
0.0034   ;[f]
0.2 ;[p]
0.00973  ;[f]
0.00862  ;[f]
0.0041   ;[f]
```

```
0.0046   ;[f]
0.00061  ;[f]
-0.0056  ;[f]
0.0056   ;[f]
0.2 ;[p]

$SIGMA
0.1 ;[p]
0.1 ;[p]

$PRIOR NWPRI
; Omega prior
$OMEGAP BLOCK(8)
0.2 FIX
0.0 0.2
0.0 0.0 0.2
0.0 0.0 0.0 0.2
0.0 0.0 0.0 0.0 0.2
0.0 0.0 0.0 0.0 0.0 0.2
0.0 0.0 0.0 0.0 0.0 0.0 0.2
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2
; degrees of freedom for OMEGA prior
$OMEGAPD
(8 FIXED)           ;[dfo]

; Starting with a short iterative two stage analysis brings the
; results closer so less time needs to be spent during the
; burn-in of the BAYES analysis

$EST METHOD=ITS INTERACTION SIGL=4 NITER=15 PRINT=1
     FILE=example6.ext NOABORT NOPRIOR=1

$EST METHOD=BAYES INTERACTION NBURN=4000 SIGL=4 NITER=10000
     PRINT=10 CTYPE=3 FILE=example6.txt NOABORT NOPRIOR=0

; By default, ISAMPLE_M* are 2.  Since there are many data points
; per subject, setting these to 1 is enough, and it reduces the
; time of the analysis

     ISAMPLE_M1=1 ISAMPLE_M2=1 ISAMPLE_M3=1 IACCEPT=0.4

$COV MATRIX=R UNCONDITIONAL
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 6S

This is example6s.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory. Note the old style syntax for priors.

```
;Model Desc: Receptor Mediated Clearance model with Dynamic Change
;            in Receptors
;Project Name: nm7examples
;Project ID: NO PROJECT DESCRIPTION

$PROB RUN# example6s (from r2compl)
$INPUT C SET ID JID TIME DV=CONC DOSE=AMT RATE EVID MDV CMT
$DATA example6.csv IGNORE=C

; The new numerical integration solver is used, although ADVAN=9
; is also efficient for this problem.

$SUBROUTINES ADVAN13 TRANS1 TOL=4
$MODEL NCOMPARTMENTS=3

$PRIOR NWPRI NTHETA=8, NETA=8, NTHP=0, NETP=8, NPEXP=1

$PK
MU_1=THETA(1)
MU_2=THETA(2)
MU_3=THETA(3)
MU_4=THETA(4)
MU_5=THETA(5)
MU_6=THETA(6)
MU_7=THETA(7)
MU_8=THETA(8)
VC=EXP(MU_1+ETA(1))
K10=EXP(MU_2+ETA(2))
K12=EXP(MU_3+ETA(3))
K21=EXP(MU_4+ETA(4))
VM=EXP(MU_5+ETA(5))
KMC=EXP(MU_6+ETA(6))
K03=EXP(MU_7+ETA(7))
K30=EXP(MU_8+ETA(8))
S3=VC
S1=VC
KM=KMC*S1
F3=K03/K30

$DES
DEL=1.0E-10
DADT(1) = -(K10+K12)*A(1) + K21*A(2) - VM*A(1)*A(3)/(A(1)+KM+DEL)
DADT(2) = K12*A(1) - K21*A(2)
DADT(3) =  -VM*A(1)*A(3)/(A(1)+KM+DEL) - K30*A(3) + K03
```

```
$ERROR
CALLFL=0
ETYPE=1
IF(CMT.NE.1) ETYPE=0
IPRED=F
Y = F + F*ETYPE*EPS(1) + F*(1.0-ETYPE)*EPS(2)


$THETA
;Initial Thetas
( 4.0 )  ;[MU_1]
( -2.1 ) ;[MU_2]
( 0.7 )  ;[MU_3]
( -0.17 );[MU_4]
( 2.2 ) ;[MU_5]
( 0.14 )  ;[MU_6]
( 3.7 )  ;[MU_7]
( -0.7) ;[MU_8]
; degrees of freedom for OMEGA prior
(8 FIXED)           ;[dfo]


;Initial Omegas
$OMEGA BLOCK(8)
0.2 ;[p]
-0.0043  ;[f]
0.2 ;[p]
0.0048   ;[f]
-0.0023  ;[f]
0.2 ;[p]
0.0032   ;[f]
0.0059   ;[f]
-0.0014  ;[f]
0.2 ;[p]
0.0029   ;[f]
0.0027 ;[f]
-0.00026 ;[f]
-0.0032  ;[f]
0.2 ;[p]
-0.0025  ;[f]
0.00097  ;[f]
0.0024   ;[f]
0.00197  ;[f]
-0.0080  ;[f]
0.2 ;[p]
0.0031   ;[f]
-0.00571 ;[f]
0.0030   ;[f]
-0.0074  ;[f]
0.0025   ;[f]
```

```
0.0034   ;[f]
0.2 ;[p]
0.00973  ;[f]
0.00862  ;[f]
0.0041   ;[f]
0.0046   ;[f]
0.00061  ;[f]
-0.0056  ;[f]
0.0056   ;[f]
0.2 ;[p]

; Omega prior
$OMEGA BLOCK(8)
0.2 FIX
0.0 0.2
0.0 0.0 0.2
0.0 0.0 0.0 0.2
0.0 0.0 0.0 0.0 0.2
0.0 0.0 0.0 0.0 0.0 0.2
0.0 0.0 0.0 0.0 0.0 0.0 0.2
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2

$SIGMA  BLOCK(1) 0.1 ;[p]
;$SIGMA  BLOCK(1) 0.1 ;[p]
$SIGMA  BLOCK(1) SAME

; Starting with a short iterative two stage analysis brings the
; results closer so less time needs to be spent during the burn-in
; of the BAYES analysis

$EST METHOD=ITS INTERACTION SIGL=4 NITER=30 PRINT=1
     FILE=example6s.ext NOABORT NOPRIOR=1 CTYPE=3

$EST METHOD=IMP INTERACTION SIGL=4 NITER=30 PRINT=1
     FILE=example6s.ext NOABORT NOPRIOR=1 ISAMPLE=100

$EST METHOD=BAYES INTERACTION NBURN=4000 SIGL=4 NITER=1000
     PRINT=10 CTYPE=3 FILE=example6s.txt NOABORT NOPRIOR=0
     PSAMPLE_M1=1 PSAMPLE_M2=1 PSAMPLE_M3=1

; By default, ISAMPLE_M* are 2.  Since there are many data points
; per subject, setting these to 1 is enough, and it reduces the
; time of the analysis

     ISAMPLE_M1=1 ISAMPLE_M2=1 ISAMPLE_M3=1 IACCEPT=0.4

$COV MATRIX=R UNCONDITIONAL
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 6SB

This is example6sb.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory. Note the old style syntax for priors.

```
;Model Desc: Receptor Mediated Clearance model with Dynamic Change
;            in Receptors
;Project Name: nm7examples
;Project ID: NO PROJECT DESCRIPTION

$PROB RUN# example6sb (from r2compl)
$INPUT C SET ID JID TIME DV=CONC DOSE=AMT RATE EVID MDV CMT
$DATA example6.csv IGNORE=C

; The new numerical integration solver is used, although ADVAN=9
; is also efficient for this problem.

$SUBROUTINES ADVAN13 TRANS1 TOL=4
$MODEL NCOMPARTMENTS=3

$PRIOR NWPRI NTHETA=8, NETA=8, NTHP=0, NETP=8, NPEXP=1

$PK
MU_1=THETA(1)
MU_2=THETA(2)
MU_3=THETA(3)
MU_4=THETA(4)
MU_5=THETA(5)
MU_6=THETA(6)
MU_7=THETA(7)
MU_8=THETA(8)
VC=EXP(MU_1+ETA(1))
K10=EXP(MU_2+ETA(2))
K12=EXP(MU_3+ETA(3))
K21=EXP(MU_4+ETA(4))
VM=EXP(MU_5+ETA(5))
KMC=EXP(MU_6+ETA(6))
K03=EXP(MU_7+ETA(7))
K30=EXP(MU_8+ETA(8))
S3=VC
S1=VC
KM=KMC*S1
F3=K03/K30

$DES
DADT(1) = -(K10+K12)*A(1) + K21*A(2) - VM*A(1)*A(3)/(A(1)+KM)
DADT(2) = K12*A(1) - K21*A(2)
DADT(3) =  -VM*A(1)*A(3)/(A(1)+KM) - K30*A(3) + K03

$ERROR
```

```
CALLFL=0
ETYPE=1
IF(CMT.NE.1) ETYPE=0
IPRED=F
Y = F + F*ETYPE*EPS(1) + F*(1.0-ETYPE)*EPS(1)


$THETA
;Initial Thetas
( 4.0 )  ;[MU_1]
( -2.1 ) ;[MU_2]
( 0.7 )  ;[MU_3]
( -0.17 );[MU_4]
( 2.2 ) ;[MU_5]
( 0.14 )  ;[MU_6]
( 3.7 )  ;[MU_7]
( -0.7) ;[MU_8]
; degrees of freedom for OMEGA prior
(8 FIXED)           ;[dfo]


;Initial Omegas
$OMEGA BLOCK(8)
0.2 ;[p]
-0.0043  ;[f]
0.2 ;[p]
0.0048   ;[f]
-0.0023  ;[f]
0.2 ;[p]
0.0032   ;[f]
0.0059   ;[f]
-0.0014  ;[f]
0.2 ;[p]
0.0029   ;[f]
0.0027 ;[f]
-0.00026 ;[f]
-0.0032  ;[f]
0.2 ;[p]
-0.0025  ;[f]
0.00097  ;[f]
0.0024   ;[f]
0.00197  ;[f]
-0.0080  ;[f]
0.2 ;[p]
0.0031   ;[f]
-0.00571 ;[f]
0.0030   ;[f]
-0.0074  ;[f]
0.0025   ;[f]
0.0034   ;[f]
```

```
0.2 ;[p]
0.00973  ;[f]
0.00862  ;[f]
0.0041   ;[f]
0.0046   ;[f]
0.00061  ;[f]
-0.0056  ;[f]
0.0056   ;[f]
0.2 ;[p]

; Omega prior
$OMEGA BLOCK(8)
0.2 FIX
0.0 0.2
0.0 0.0 0.2
0.0 0.0 0.0 0.2
0.0 0.0 0.0 0.0 0.2
0.0 0.0 0.0 0.0 0.0 0.2
0.0 0.0 0.0 0.0 0.0 0.0 0.2
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2

$SIGMA  BLOCK(1) 0.1 ;[p]

; Starting with a short iterative two stage analysis brings the
; results closer so less time needs to be spent during the burn-in
; of the BAYES analysis

$EST METHOD=ITS INTERACTION SIGL=4 NITER=30 PRINT=1
     FILE=example6sb.ext NOABORT NOPRIOR=1 CTYPE=3

$EST METHOD=IMP INTERACTION SIGL=4 NITER=30 PRINT=1
     FILE=example6sb.ext NOABORT NOPRIOR=1 ISAMPLE=100

$EST METHOD=BAYES INTERACTION NBURN=4000 SIGL=4 NITER=1000 PRINT=10
     CTYPE=3 FILE=example6sb.txt NOABORT NOPRIOR=0

; By default, ISAMPLE_M* are 2.  Since there are many data points
;  per subject, setting these to 1 is enough, and it reduces the
; time of the analysis

     ISAMPLE_M1=1 ISAMPLE_M2=1 ISAMPLE_M3=1 IACCEPT=0.4

$COV MATRIX=R UNCONDITIONAL
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 7R

This is example7r.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory.

```
;Model Desc: Interoccasion Variability
;Project Name: nm7examples
;Project ID: NO PROJECT DESCRIPTION

$PROB run# example7r
$INPUT C SET ID  TIME  AMT RATE EVID MDV CMT DV OCC
$ABBR REPLACE ETA(OCC_CL)=ETA(3,4,5)
$DATA example7r.csv IGNORE=C

$SUBROUTINES ADVAN1 TRANS2

$PK
MU_1=THETA(1)
MU_2=THETA(2)
V=DEXP(MU_1+ETA(1))
S1=V
VC=V
CL=DEXP(MU_2+ETA(2))*EXP(ETA(OCC_CL))

$ERROR
IPRED=F
Y = F+F*EPS(1)

;Initial Thetas
$THETA
 2.0  ;[MU_1]
 2.0  ;[MU_2]

;Initial omegas
$OMEGA BLOCK(2)
 .3 ;[p]
 -.01  ;[f]
 .3 ;[p]
$OMEGA BLOCK(1)
 .1  ;[p]
$OMEGA BLOCK(1) SAME(2)

$SIGMA
 0.1 ;[p]

$PRIOR NWPRI
; Degrees of freedom for Prior Omega blocks
$OMEGAPD (2.0 FIXED) (1.0 FIXED)
; Prior Omegas
$OMEGAP BLOCK(2)
```

```
 .14 FIX
 0.0 .125
$OMEGAP BLOCK(1) .0164 FIX
$OMEGAP BLOCK(1) SAME(2)

$EST METHOD=ITS INTERACTION FILE=example7r.ext   NITER=10000
     PRINT=5 NOABORT SIGL=8 CTYPE=3 CITER=10
     NOPRIOR=1 CALPHA=0.05 NSIG=2

$EST METHOD=SAEM INTERACTION NBURN=30000 NITER=500 SIGL=8
     ISAMPLE=2 PRINT=10 SEED=1556678 CTYPE=3
     CITER=10 CALPHA=0.05 NOPRIOR=1

$EST METHOD=IMP  INTERACTION EONLY=1 NITER=4 ISAMPLE=3000
     PRINT=1 SIGL=10 NOPRIOR=1 MAPITER=0

$EST METHOD=BAYES INTERACTION FILE=example7r.txt NBURN=10000
     NITER=10000 PRINT=100 CTYPE=3 CITER=10
     CALPHA=0.05 NOPRIOR=0

$EST METHOD=COND INTERACTION MAXEVAL=9999 NSIG=3 SIGL=10 PRINT=5
     NOABORT NOPRIOR=1
     FILE=example7r.ext

$COV MATRIX=R PRINT=E UNCONDITIONAL
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 8

This is example8.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory.

```
;Model Desc: Two compartment Model, Using ADVAN3, TRANS4
;Project Name: nm7examples
;Project ID: NO PROJECT DESCRIPTION

$PROB RUN# Example 8 (from samp5l)
$INPUT C SET ID JID TIME  DV=CONC AMT=DOSE RATE EVID MDV CMT
       CLX V1X QX V2X SDIX SDSX
$DATA example8.csv IGNORE=C

$SUBROUTINES ADVAN3 TRANS4


$PK
include nonmem_reserved_general
; Request extra information for Bayesian analysis.
; An extra call will then be made for accepted samples
BAYES_EXTRA_REQUEST=1
MU_1=THETA(1)
MU_2=THETA(2)
MU_3=THETA(3)
MU_4=THETA(4)
CL=DEXP(MU_1+ETA(1))
V1=DEXP(MU_2+ETA(2))
Q=DEXP(MU_3+ETA(3))
V2=DEXP(MU_4+ETA(4))
S1=V1
; When Bayes_extra=1, then this particular set of individual
; parameters were "accepted" So you may record them if you wish
  IF(BAYES_EXTRA==1 .AND. ITER_REPORT>=0 .AND. TIME==0.0) THEN
"  WRITE(51,98) ITER_REPORT,ID,CL,V1,Q,V2
" 98 FORMAT(I12,1X,F14.0,4(1X,1PG12.5))
ENDIF

$ERROR
include nonmem_reserved_general
BAYES_EXTRA_REQUEST=1
Y = F + F*EPS(1)
IF(BAYES_EXTRA==1 .AND. ITER_REPORT>=0 ) THEN
" WRITE(52,97) ITER_REPORT,ID,TIME,F
" 97 FORMAT(I12,1X,F14.0,2(1X,1PG12.5))
ENDIF

; Initial values of THETA
$THETA
(2.0) ;[LN(CL)]
```

```
(2.0) ;[LN(V1)]
(2.0) ;[LN(Q)]
(2.0) ;[LN(V2)]
;INITIAL values of OMEGA
$OMEGA BLOCK(4)
0.15   ;[P]
0.01  ;[F]
0.15   ;[P]
0.01  ;[F]
0.01  ;[F]
0.15   ;[P]
0.01  ;[F]
0.01  ;[F]
0.01  ;[F]
0.15   ;[P]
;Initial value of SIGMA
$SIGMA
(0.6 )   ;[P]


$PRIOR NWPRI
; Prior information to the Thetas.
$THETAP (2.0 FIX)x4
$THETAPV BLOCK(4) FIX VALUES(10000.0,0.0)

; Prior information to the OMEGAS.
$OMEGAP BLOCK(4)
0.2 FIX
0.0  0.2
0.0  0.0 0.2
0.0  0.0 0.0 0.2
$OMEGAPD (4 FIX)

$EST METHOD=BAYES INTERACTION FILE=example8.ext NBURN=10000
     NITER=1000 PRINT=100 NOPRIOR=0 CTYPE=3 CINTERVAL=100
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 9

This is example9.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory.

```
;Model Desc: Two compartment Model, Using ADVAN3, TRANS4
;Project Name: nm7examples
;Project ID: NO PROJECT DESCRIPTION

$PROB RUN# Example 9 (from samp5l)
$INPUT C SET ID JID TIME  DV=CONC AMT=DOSE RATE EVID MDV CMT
       CLX V1X QX V2X SDIX SDSX
$DATA example9.csv IGNORE=C

$SUBROUTINES ADVAN3 TRANS4 OTHER=aneal.f90

$PK
MU_1=THETA(1)
MU_2=THETA(2)
MU_3=THETA(3)
MU_4=THETA(4)
CL=DEXP(MU_1+ETA(1))
V1=DEXP(MU_2+ETA(2))
Q=DEXP(MU_3+ETA(3))
V2=DEXP(MU_4+ETA(4))
S1=V1

$ERROR
Y = F + F*EPS(1)

; Initial values of THETA
$THETA
(0.001, 2.0) ;[LN(CL)]
(0.001, 2.0) ;[LN(V1)]
(0.001, 2.0) ;[LN(Q)]
(0.001, 2.0) ;[LN(V2)]
;INITIAL values of OMEGA
$OMEGA BLOCK(4)
0.05   ;[P]
0.01  ;[F]
0.05   ;[P]
0.01  ;[F]
0.01  ;[F]
0.05   ;[P]
0.01  ;[F]
0.01  ;[F]
0.01  ;[F]
0.05   ;[P]
;Initial value of SIGMA
$SIGMA
```

```
(0.6 )   ;[P]

$EST METHOD=SAEM INTERACTION FILE=example9.ext NBURN=5000
     NITER=500 PRINT=10 NOABORT SIGL=6 CTYPE=3 CINTERVAL=100
     CITER=10 CALPHA=0.05
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 10

This is example10.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory.

```
$PROB  F_FLAG04est2a.ctl
$INPUT C ID DOSE=AMT TIME DV WT TYPE
$DATA example10.csv IGNORE=@

$SUBROUTINES  ADVAN2 TRANS2


$PK
   CALLFL=1
   MU_1=DLOG(THETA(1))
   KA=DEXP(MU_1+ETA(1))
   MU_2=DLOG(THETA(2))
   V=DEXP(MU_2+ETA(2))
   MU_3=DLOG(THETA(3))
   CL=DEXP(MU_3+ETA(3))
   SC=V/1000

$THETA  5.0 10.0 2.0 0.1 0.1

$OMEGA BLOCK (3)
0.5
0.01 0.5
0.01 0.01 0.5


; Because THETA(4) and THETA(5) have no inter-subject variability
; associated with them, the algorithm must use a more computationally
; expensive gradient evaluation for these two parameters

$SIGMA 0.1


$PRIOR NWPRI
; Priors to Omegas
$OMEGAP BLOCK (3)
0.09 FIX
0.0 0.09
0.0 0.0 0.09
$OMEGAPD (3 FIX)

$ERROR
    EXPP=THETA(4)+F*THETA(5)
IF (TYPE.EQ.0) THEN
; PK Data
```

```
    F_FLAG=0
    Y=F+F*ERR(1) ; a prediction
 ELSE
; Categorical data
    F_FLAG=1
; Use protected exponent PEXP, to avoid numerical overflow
    A=PEXP(EXPP)
    B=1+A
    Y=DV*A/B+(1-DV)/B      ; a likelihood
 ENDIF



$EST METHOD=ITS INTER LAP NITER=1000 PRINT=5 SIGL=6 NSIG=2
     NOABORT NOPRIOR=1 CTYPE=3 CITER=10 CALPHA=0.05
     FILE=example10.ext
; Because of categorical data, which can make conditional density highly
; non-normal, select a t-distribution with 4 degrees of freedom for
; importance sampling proposal density
$EST METHOD=IMP INTER LAP NITER=1000 PRINT=1 ISAMPLE=300 DF=4
     IACCEPT=1.0
$EST METHOD=IMP EONLY=1 NITER=5 ISAMPLE=1000 PRINT=1 DF=4
     IACCEPT=1.0 MAPITER=0

$EST METHOD=SAEM EONLY=0 INTER LAP NBURN=2000 NITER=1000 PRINT=50
     DF=0 IACCEPT=0.4
$EST METHOD=IMP EONLY=1 NITER=5 ISAMPLE=1000 PRINT=1 DF=4
     IACCEPT=1.0 MAPITER=0

$EST METHOD=BAYES NBURN=3000 NSAMPLE=3000 PRINT=100
     FILE=example10.txt DF=0 IACCEPT=0.4 NOPRIOR=0

$EST METHOD=COND LAP INTER MAXEVAL=9999 PRINT=1 FILE=example10.ext
     NOPRIOR=1 NOHABORT

$COV UNCONDITIONAL PRINT=E MATRIX=R SIGL=10
$TABLE ID DOSE WT TIME TYPE DV A NOPRINT FILE=example10.tab
```

REFERENCES: Guide Introduction_7

BAYES EXAMPLE 10L

This is example10l.ctl from the NONMEM 7 distribution medium. It, along with the data file, can be found in the examples directory.

```
$PROB  F_FLAG04est2a.ctl
$INPUT C ID DOSE=AMT TIME DV WT TYPE
$DATA example10l.csv IGNORE=@

$SUBROUTINES  ADVAN2 TRANS2


$PK
   CALLFL=1
   MU_1=THETA(1)
   KA=DEXP(MU_1+ETA(1))
   MU_2=THETA(2)
   V=DEXP(MU_2+ETA(2))
   MU_3=THETA(3)
   CL=DEXP(MU_3+ETA(3))
   SC=V/1000

$THETA  1.6 2.3 0.7 0.1 0.1

$OMEGA BLOCK (3)
0.5
0.01 0.5
0.01 0.01 0.5


; Because THETA(4) and THETA(5) have no inter-subject variability
; associated with them, the algorithm must use a more computationally
; expensive gradient evaluation for these two parameters

$SIGMA 0.1


$PRIOR NWPRI
; Priors to Omegas
$OMEGAP BLOCK (3)
0.09 FIX
0.0 0.09
0.0 0.0 0.09
$OMEGAPD (3 FIX)


$ERROR
    EXPP=THETA(4)+F*THETA(5)
IF (TYPE.EQ.0) THEN
```

```
; PK Data
    F_FLAG=0
    Y=F+F*ERR(1) ; a prediction
 ELSE
; Categorical data
    F_FLAG=1
; Use protected exponent PEXP, to avoid numerical overflow
    A=PEXP(EXPP)
    B=1+A
    Y=DV*A/B+(1-DV)/B      ; a likelihood
 ENDIF

$EST METHOD=ITS INTER LAP NITER=1000 PRINT=5 SIGL=6 NSIG=2 NOABORT
     NOPRIOR=1 CTYPE=3 CITER=10 CALPHA=0.05 FILE=example10l.ext
; Because of categorical data, which can make conditional density highly
; non-normal, select a t-distribution with 4 degrees of freedom for
; importance sampling proposal density
$EST METHOD=IMP INTER LAP NITER=1000 PRINT=1 ISAMPLE=300 DF=4 IACCEPT=1.0
$EST METHOD=IMP EONLY=1 NITER=5 ISAMPLE=1000 PRINT=1 DF=4 IACCEPT=1.0
     MAPITER=0

$EST METHOD=SAEM EONLY=0 INTER LAP NBURN=2000 NITER=1000 PRINT=50
     DF=0 IACCEPT=0.4
$EST METHOD=IMP EONLY=1 NITER=5 ISAMPLE=1000 PRINT=1 DF=4
     IACCEPT=1.0 MAPITER=0

$EST METHOD=BAYES NBURN=3000 NSAMPLE=3000 PRINT=100
     FILE=example10l.txt DF=0 IACCEPT=0.4 NOPRIOR=0

$EST METHOD=COND LAP INTER MAXEVAL=9999 PRINT=1 FILE=example10l.ext
     NOPRIOR=1

$COV UNCONDITIONAL PRINT=E MATRIX=R SIGL=10
$TABLE ID DOSE WT TIME TYPE DV A NOPRINT FILE=example10l.tab
```

REFERENCES: Guide Introduction_7

BIND EXAMPLE

EXAMPLES OF THE USE OF $BIND

Case 1.

Suppose a fragment of the dose records looks like this:

```
TIME AMT PREP    X
  0  100    4    .
 10  100    1   90
 20    0    0   80
```

Suppose $INPUT and $PK include this code:

```
$INPUT ID TIME AMT DV PREP X
$PK
    F1=1
    IF (PREP.EQ.4) F1=THETA(3)
    ALAG1=THETA(5)
    CL=THETA(1)*X*EXP(ETA(1))
```

It is intended that theta(3) be the ratio of bioavailabilities of preparation 4 to preparation 1. However, calls to PK are made only at event times 0, 10 and 20. Suppose the value of theta(5) (lag time) is 5. After 5 time units, the first dose enters the system. The value of PREP (dose preparation type) is taken from the event record at time 10, which is 1. F1 is computed incorrectly. If the estimation step is run, the gradient with theta(3) will be 0, indicating the problem.

Case 2.

If lag time is not modeled (no ALAG1), but the doses are multiple (ADDL>0, II>0), a more subtle error can occur. Suppose a fragment of the data records looks like this:

```
TIME AMT PREP    X  II  ADDL
  0  100    4    .   5     1
 10  100    1   90   5     1
 20    0    0   80   .     .
```

Suppose $INPUT and $PK include this code:

```
$INPUT ID TIME AMT DV PREP X II ADDL
$PK
    F1=1
    IF (PREP.EQ.4) F1=THETA(3)
    CL=THETA(1)*X*EXP(ETA(1))
```

F1 is computed correctly for the initiating dose records at times 0 and 10, and for the additional dose at time 15, but incorrectly for the additional dose at time 5. Theta(3) applies only to the first dose having PRED=4. (The gradient for theta(3) is not 0.) The same problem could have occurred with the dose having PREP=1, except that by an accident of coding, PREP=0 (on the record having TIME=20) is treated the same as PREP=1, so that the additional dose at time 15 has F1=1 as it should.

On the other hand, the values of X are presumably recorded so that 90 is valid for the time 0 through 10 and 80 is valid for the time 10 through 20. Therefore, the model for CL is correct in both examples.

In either of these two examples, the model can be corrected by the insertion of $BIND and CALLFL=-2:

```
$INPUT ID TIME AMT DV PREP X II ADDL ; II & ADDL only in case 2
$BIND   -    -   -  - DOSE NEXT
$PK
    CALLFL=-2
```

Because of CALLFL=-2 in the abbreviated code, additional calls to PK are made at the nonevent dose times (5 in case 1; 5 and 15 in case 2). Because of the $BIND record, PREP has the value at these calls from the initiating dose record (4 at time 5, 1 at time 15). F1 is computed appropriately for the dose preparation. X still has values from the next event record, so the computation of CL is unchanged, e.g., it is the same from time=0 to time=5 as it is from time=5 to time=10.

(See **$BIND**).

REFERENCES: Guide IV, section V.C.2 , V.C.5

REFERENCES: Guide VI, section III.B.2 , V.K

BIVARIATE FUNCTION

This is a fully-worked out example of the use of the $ABBREVIATED FUNCTION option to declare a user-defined function. In this example, the function BIVARIATE is used to compute the integral of the bivariate normal distribution of two correlated data points, DV1 and DV2, with correlation RHO.

This is bivariate.ctl from the NONMEM 7 distribution medium. This file, the data file bivariate.csv, and the function's source code bivariate.f90 can be found in the examples directory.

```
$PROB BIVARIATE EXAMPLE
; THESE DECLARATIONS ALLOW ANY FUNCTION TO HAVE
; ALTERNATIVE DIMENSIONS FOR THEIR ARRAYS
; BUT, USER DEFINED DIMENSIONS ARE PASSED AS THE
; LAST ARGUMENT TO FUNC, SUCH AS:
; BV=BIVARIATE(VBI(1),FNC001_1(1,1),FNC001_2(1,1,1),5)
$ABBR FUNCTION BIVARIATE(VBI,5)

$INPUT SIM ID DOSE DV TIME
$DATA bivariate.csv IGNORE=@

$SUBROUTINES OTHER=bivariate.f90

$PRED
  B1=THETA(1)
  B2=THETA(2)
  B3=THETA(3)
  K =LOG(2)/EXP(THETA(4))
  ED50=EXP(THETA(5))
  U =(1-EXP(-K*TIME ))

  MU_1=B1+B3*DOSE/(DOSE+ED50)
  MU_2=B2
  MXB=MU_1+ETA(1)
  MXU=MU_2+ETA(2)
  MX =MXB + MXU*U  ;***Current model prediction***;

  PHIMX=PHI(MX)

   IF(NEWIND.NE.2) THEN
   TIMEP=0
   MXP=0
   DVP=0
   PHIMXP=0.5
   ENDIF

  RHOB=(2/(1+EXP(-THETA(8)))-1)
  IF(RHOB>0.0)  RHO=RHOB**(TIME-TIMEP)
  IF(RHOB==0.0) RHO=0.0
  IF(RHOB<0.0)  RHO=-(-RHOB)**(TIME-TIMEP)
```

```
  PC =(1-PHIMX) *(1-DV ) + PHIMX*DV
  IF(PC.LE.0.0) EXIT


  V=SQRT(1+OMEGA(1,1)+OMEGA(2,2)*U**2)
  POPP = (B1+B2*U +B3*DOSE/(DOSE+ED50))/V ;*Population mean prediction*;

  IF (TIME.EQ.1) THEN
    JP=PC
    PCP=1.0
  ELSE
  ;***Pass information to bivariate normal***;
   VBI(1)=RHO
   VBI(2)=MX
   VBI(3)=MXP
   VBI(4)=1  ;*0 = Upper tail as in Drezner & Wesolowsky; 1 = Bottom tail*;
   VBI(5)=1  ;*0 = 3 pt approximation; 1 = 5 point approximation*;
   BV=BIVARIATE(VBI)
   JP=((DV-1)*(DVP-1)+(DV-1)*(1-2*DVP)*PHIMXP+(DVP-1) &
   *(1-2*DV)*PHIMX+(1-2*DV)*(1-2*DVP)*BV)
  ENDIF
   IF(JP.LE.0.0) EXIT
   LOGL=LOG(JP/PCP)
   Y = -2*LOGL

  MXP=MX
  PCP=PC
  DVP=DV
  TIMEP=TIME
  PHIMXP=PHIMX

$THETA
      -1.7   ; 1  B1
       1.2   ; 2  B2
       2.9   ; 3  B3
       1.4   ; 4  LOG(B4)
       1.2   ; 5  LOG(B5)
       (0.0 FIXED)   ; 6  LOG SQRT VAR(ETA1)
       (0.0 FIXED)  ; 7  LOG SQRT VAR(ETA2)
       2.2   ; 8  RHO parameter

$OMEGA DIAGONAL(2)
      0.8       ; V1
      0.8       ; V2

;$EST METHOD=IMP LAPLACE -2LL PRINT=1 NITER=300 ISAMPLE=300
;SIGL=6 CTYPE=3 NOHABORT
$EST MAX=0 PRINT=1 METHOD=1 LAPLACE -2LL SIGL=10 NOHABORT
$COV COMPRESS MATRIX=R PRINT=E UNCONDITIONAL
```

Here is a fragment of the data file bivariate.csv:

```
SIM,ID,DOSE,DV,TIME
1,1,0,0,1
1,1,0,0,2
1,1,0,0,4
1,1,0,0,8
1,1,0,0,16
1,1,0,0,24
1,1,0,0,36
```

This is the file bivariate.f90. (The extension .f90 in the file name is not actually required because the contents of the file is copied to FSUBS and compiled as part of FSUBS. It is not compiled independently.)

```
! Integral of Bivariate normal distribution of two correlated data points,
! DV1, DV2, correlation RHO.
! BV1, BV2 and GAUSS FROM Drezner and Wesolowsky,
! J. Statistical Computat. Simul. 35, pp. 101-107, 1990.
! BV2 is more accurate for extreme values of rho
! Partial derivatives worked out by Robert Bauer.
!
! How to use in NONMEM 7.4
! VBI is a suggested vector name.  Could be any name.
! But the function name must match what is in bivariate.f90
! $ABBR FUNCTION BIVARIATE(VBI,5)
! $SUBROUTINES ... OTHER=bivariate.f90
! $PK
! ...
! DV1 AND DV2 SHOULD BE TWO SEPARATE DATA ITEMS ON EACH RECORD.
!   VBI(1)=RHO
!   VBI(2)=DV1 ! H
!   VBI(3)=DV2 !K
!   VBI(4)=DTYPE !=0 FOR H TO INF, K TO INF, OR 1 FOR -INF TO H, -INF TO K
!   VBI(5)=BVTYPE ! BVTYPE=0 USES SIMPLE INTEGRATOR, BVTYPE=1 USES MORE
                  ! ACCURATE ONE FOR LARGE VALUES OF RHO
!   F_FLAG=1
! IT IS SAFER TO AVOID BV BEING EXACTLY 0, JUST MAKE SURE
! IT IS REALLY SMALL, TO AVOID LOG() ERRORS
!    BV=BIVARIATE(VBI)+1.0D-30
!    IF YOU USE -2LL FORMAT (DEFAULT):
!    Y=-2.0D+00*LOG(BV)

      FUNCTION BIVARIATE(X,X1,X2,NDIM)
! NDIM SHOULD EQUAL 5
!      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT NONE
      INTEGER NDIM
      REAL*8 X(NDIM),X1(NDIM),X2(NDIM,NDIM)
      REAL*8 R,H,K,ITYPE,PI,PI2,SPI2,R1,RSQR,HRK,KRH,PHRK,PKRH,XH,XK
      REAL*8 XHRK,XKRH,XC,BV,BV2,BIVARIATE,GAUSS,BV1
      EXTERNAL GAUSS,BV1,BV2
      INTEGER BVTYPE
```

```
! unload the arguments
      R=X(1)
      H=X(2)
      K=X(3)
      ITYPE=X(4)
      BVTYPE=X(5)
! ITYPE=0 INTGRATE H TO INF, K TO INF
! ITYPE=1 INTGRATE -INF TO H -INF TO K
! BVTYPE=0 USES THE SIMPLER BV1
! BVTYPE=1 USES THE MORE ACCURATE BV2
! SET UP SOME USEFUL CALCUATIONS FOR FIRST
! AND SECOND DERIVATIVES OF THE ARGUMENTS
! DERIVATIVES NOT NEEDED IF YOU USE IMP, SAEM, BAYES, OR IF
! YOU USE LAPLACE OR ITS WITH OPTMAP=1 ETADER=3 (NONMEM 7.3), OR
! DERIVATIVES NEEDED IF YOU USE STANDARD LAPLACE
      PI=3.14159265358979238D+00
      PI2=2.0D+00*pi
      spi2=sqrt(pi2)
      R1=1.0D+00-R*R
      RSQR=SQRT(ABS(R1))
      IF(RSQR.LE.0.0D+00) RSQR=1.0D-100
      IF(R1.LE.0.0D+00) R1=1.0D-100
      HRK=(H-R*K)/RSQR
      KRH=(K-R*H)/RSQR
      PHRK=1.0D+00-GAUSS(HRK)
      PKRH=1.0D+00-GAUSS(KRH)
      XH=EXP(-H*H/2.0D+00)/SPI2
      XK=EXP(-K*K/2.0D+00)/SPI2
      XHRK=EXP(-HRK*HRK/2.0D+00)/SPI2/RSQR
      XKRH=EXP(-KRH*KRH/2.0D+00)/SPI2/RSQR
      XC=EXP(-(H*H-2.0D+00*R*H*K+K*K)/2.0D+00/R1)/RSQR/PI2

! parital F WRT RHO, from Drezner and Wesolowsky, equation (4)
      X1(1)=XC
! parital F WRT H
      X1(2)=XH*PKRH+(ITYPE-1.0D+00)*XH
! parital F WRT K
      X1(3)=XK*PHRK+(ITYPE-1.0D+00)*XK
! 2ND parital F WRT RHO,RHO
      X2(1,1)=R/R1*XC+XC*(H*K*(1.0D+00-R)*(1.0D+00-R)-R*(H-K)*(H-K))/R1/R1
! 2ND parital F WRT RHO,H
      X2(1,2)=XC*(R*K-H)/R1
! 2ND parital F WRT RHO,K
      X2(1,3)=XC*(R*H-K)/R1
      X2(2,1)=X2(1,2)
      X2(3,1)=X2(1,3)
! 2ND parital F WRT H,K
      X2(2,3)=XC
      X2(3,2)=X2(2,3)
! 2ND parital F WRT H,H
```

```
      X2(2,2)=-X1(2)*H-XC*R
! 2ND parital F WRT K,K
      X2(3,3)=-X1(3)*K-XC*R
!      write(50,*) 'A ',x1(1),x1(2),x1(3)
!      write(50,*) 'B ',x2(1,1),x2(1,2),x2(1,3),x2(2,1), &
!      x2(2,2),x2(2,3),x2(3,1),x2(3,2),x2(3,3)

! bV2 is more accurate for exterme rho values.
      IF(BVTYPE==0) THEN
      BV=BV1(H,K,R)
      ELSE
      BV=BV2(H,K,R)
      ENDIF

      IF(ITYPE==1.0D+00) BV=BV-GAUSS(H)-GAUSS(K)+1.0D+00
      BIVARIATE=BV

      RETURN
      END

      FUNCTION BV1(H1,H2,R)
      IMPLICIT NONE
      INTEGER I
!      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 H1,H2,R,H12,BV,BV1,RR,RR2,H3,GAUSS
      EXTERNAL GAUSS
      REAL*8 X(5),W(5),DVAL
      DATA X/.04691008,.23076534,.5,.76923466,.95308992/
      DATA W/.018854042,.038088059,.0452707394,.038088059,.018854042/
      H12 =(H1*H1+ H2*H2)/2.0D+00
      H3 = H1*H2
      BV = 0.0D+00
      DO 1 I= 1,5
      RR = R*X(I)
      RR2= 1.0D+00-RR*RR
      IF(RR2.LE.0.0D+00) RR2=1.0D-100
      DVAL=(RR*H3 - H12)/RR2
      IF(DVAL.GT.300.0D+00) DVAL=300.0D+00
      BV = BV + W(I)*EXP(DVAL)/SQRT(RR2)
    1 CONTINUE
      BV = BV*R + GAUSS(H1)*GAUSS(H2)
      BV1=BV
      RETURN
      END

      FUNCTION BV2(H1,HK,R)
      IMPLICIT NONE
!      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 X(5),W(5)
      REAL*8 H1,HK,R,H2,H12,BV,BV2,R2,R3,H3,H7,H6,H5,AA,AB,R1,RR,GAUSS,RR2
```

```
      REAL*8 DVAL,DVAL2
      EXTERNAL GAUSS
      INTEGER I
      DATA X/.04691008,.23076534,.5,.76923466,.95308992/
      DATA W/.018854042,.038088059,.0452707394,.038088059,.018854042/
      H2 = HK
      H12 = (H1*H1+ H2*H2)/2.0D+00
      BV = 0.0D+00
      IF(ABS(R).LT.0.7D+00)GO TO 4
      R2= 1.0D+00-R*R
      R3 = SQRT(R2)
      IF(R.LT.0.0D+00)H2 = -H2
      H3=H1*H2
      H7 = EXP( -H3/2.0D+00)
      IF(R2.EQ.0.0D+00) GO TO 3
      H6 = ABS(H1 - H2)
      H5 = H6*H6/2.0D+00
      H6 = H6/R3
      AA = 0.5D+00 - H3/8.0D+00
      AB = 3.0D+00 - 2.0D+00*AA*H5
      BV = .13298076D+00*H6*AB*GAUSS(H6) &
      -EXP(-H5/R2)*(AB + AA*R2)*.053051647D+00
      DO 2 I= 1,5
      R1= R3*X(I)
      RR = R1*R1
      R2 = SQRT(DABS(1.0D+00 - RR))
      IF(R2.EQ.0.0D+00) R2=1.0D-100
      IF(RR.EQ.0.0D+00) RR=1.0D-100
      DVAL=-H5/RR
      IF(DVAL.GT.300.0D+00) DVAL=300.0D+00
      DVAL2=-H3/(1.0D+00 + R2)
      IF(DVAL2.GT.300.0D+00) DVAL2=300.0D+00
      BV = BV - W(I)*EXP(DVAL)*(EXP(DVAL2)/R2/H7 - 1.0D+00 - AA*RR)
    2 CONTINUE
    3 IF(R.GT.0.0D+00)BV = BV*R3*H7 + GAUSS(MAX(H1,H2))
      IF(R.LT.0.0D+00)BV = MAX(0.0D+00,GAUSS(H1) - GAUSS(H2)) - BV*R3*H7
      BV2=BV
      RETURN
    4 H3=H1*H2
      DO 1 I= 1,5
      R1= R*X(I)
      RR2= 1.0D+00-R1*R1
      IF(RR2.EQ.0.0D+00) RR2=1.0D-100
    1 BV = BV + W(I)*EXP((R1*H3 -H12)/RR2)/SQRT(RR2)
      BV2 =GAUSS(H1)*GAUSS(H2) + R*BV
      RETURN
      END

      FUNCTION GAUSS(Z)
      IMPLICIT NONE
```

```
!      IMPLICIT REAL*8 (A-H,O-Z)
       REAL*8 Z,GAUSS,X,G
       REAL*8 A(4)
       INTEGER I
       DATA A/ -.72657601,.71070688,-.142248368,.127414796/
       X= 1.0D+00/(1.0D+00 + .23164189D+00*ABS(Z))
       G = .53070271D+00
       DO 1 I= 1,4
     1 G=G*X+A(I)
       GAUSS = G*X*EXP( - Z*Z/2.0D+00)
       IF(Z.LT.0.0D+00)GAUSS = 1.0D+00 -GAUSS
       RETURN
       END

! USE PROGRAM HEADER FOR STAND-ALONE EXECUTABLE TESTING
!      PROGRAM BIVTEST
       SUBROUTINE BIVTEST
       IMPLICIT REAL*8 (A-H,O-Z)
       REAL*8 A(9),A1(9),A2(9,9)
       REAL*8 AH(9),AH1(9),AH2(9,9)
       REAL*8 AJ(9),AJ1(9),AJ2(9,9)
       REAL*8 B(9),B1(9),B2(9,9)
       REAL*8 H,K
       HDEL=1.0D-04
     1 CONTINUE
       WRITE(*,*) 'ENTER RHO,H,K,INTEGRAND-TYPE,BVTYPE'
       READ(*,*) RHO,H,K,DTYPE,BVTYPE
       A(1)=RHO
       A(2)=H
       A(3)=K
       A(4)=DTYPE
       A(5)=BVTYPE
       BV=FUNCA(A,A1,A2)
       WRITE(*,*) 'VALUE ',BV
       WRITE(*,*) GAUSS(-1.96D+00),GAUSS(1.96D+00)
       AH(1:5)=A(1:5)
       AH(1)=RHO-HDEL
       AJ(1:5)=A(1:5)
       AJ(1)=RHO+HDEL
       BVH=FUNCA(AH,AH1,AH2)
       BVJ=FUNCA(AJ,AJ1,AJ2)
       B1(1)=(BVJ-BVH)/2.0D+00/HDEL
       B2(1,1)=(AJ1(1)-AH1(1))/2.0D+00/HDEL
       B2(1,2)=(AJ1(2)-AH1(2))/2.0D+00/HDEL
       B2(1,3)=(AJ1(3)-AH1(3))/2.0D+00/HDEL
       WRITE(*,*)
       WRITE(*,*) 'ANALOG RHO',A1(1),A2(1,1),A2(1,2),A2(1,3)
       WRITE(*,*) 'NUMER RHO',B1(1),B2(1,1),B2(1,2),B2(1,3)

       AH(1:5)=A(1:5)
```

```
AH(2)=H-HDEL
AJ(1:5)=A(1:5)
AJ(2)=H+HDEL
BVH=FUNCA(AH,AH1,AH2)
BVJ=FUNCA(AJ,AJ1,AJ2)
B1(2)=(BVJ-BVH)/2.0D+00/HDEL
B2(2,1)=(AJ1(1)-AH1(1))/2.0D+00/HDEL
B2(2,2)=(AJ1(2)-AH1(2))/2.0D+00/HDEL
B2(2,3)=(AJ1(3)-AH1(3))/2.0D+00/HDEL
WRITE(*,*)
WRITE(*,*) 'ANALOG H',A1(2),A2(2,1),A2(2,2),A2(2,3)
WRITE(*,*) 'NUMER H',B1(2),B2(2,1),B2(2,2),B2(2,3)


AH(1:5)=A(1:5)
AH(3)=K-HDEL
AJ(1:5)=A(1:5)
AJ(3)=K+HDEL
BVH=FUNCA(AH,AH1,AH2)
BVJ=FUNCA(AJ,AJ1,AJ2)
B1(3)=(BVJ-BVH)/2.0D+00/HDEL
B2(3,1)=(AJ1(1)-AH1(1))/2.0D+00/HDEL
B2(3,2)=(AJ1(2)-AH1(2))/2.0D+00/HDEL
B2(3,3)=(AJ1(3)-AH1(3))/2.0D+00/HDEL
WRITE(*,*)
WRITE(*,*) 'ANALOG K',A1(3),A2(3,1),A2(3,2),A2(3,3)
WRITE(*,*) 'NUMER K',B1(3),B2(3,1),B2(3,2),B2(3,3)

GO TO 1
END
```

REFERENCES: Guide Introduction_7

CCONTR EXAMPLE

This example illustates the use of the CONTR and CCONTR stubs with NONMEM 7. This example was developed by Bill Frame.

```
$PROBLEM
$DATA NMDATA7.CSV
$INPUT ID DV
$SUB CONTR=CONTR.txt CCONTR=CCONTR.txt
$PRED
     W=THETA(2)                  ;SD
     CL=THETA(1)*EXP(ETA(1))     ;CL/F
     PRE=1/CL                    ;@ SS ASSUMING INPUT RATE = 1
     LAM=THETA(3)
     Y=(PRE**LAM-1)/LAM+EPS(1)*W
     RES1=(DV-PRE)/W
$THETA
     (0,.1)       ;CL/F
     (0,2)        ;SD ADDITIVE
     (0,.251)        ;BOX COX LAMBDA PARAMETER
$OMEGA
     .003
$SIGMA
     1 FIX
$EST MAXEVALS=9999 METH=1 PRINT=1
$COV PRINT=E
```

Files CONTR.txt and CCONTR.txt are as follows:

```
      SUBROUTINE CONTR (ICALL,CNT,IER1,IER2)
      DOUBLE PRECISION CNT
      CALL NCONTR (CNT,IER1,IER2,L2R)
      RETURN
      END

      SUBROUTINE CCONTR(I,CNT,P1,P2,IER1,IER2)
      USE SIZES, ONLY: ISIZE,DPSIZE
      USE ROCM_REAL,   ONLY: THETA=>THETAC,Y=>DV_ITM2
      USE NM_INTERFACE,ONLY: CELS
      IMPLICIT NONE
      INTEGER(KIND=ISIZE), INTENT(IN OUT) :: I,IER1,IER2
      REAL(KIND=DPSIZE),   INTENT(IN OUT) :: CNT,P1(:),P2(:,:)
      REAL(KIND=DPSIZE) :: ONE,TWO,W
      DATA ONE,TWO/1.00D+00,2.00D+00/
      SAVE
      IF (I.LE.1) RETURN
      W=Y(1)
      Y(1)=(Y(1)**THETA(3)-ONE)/THETA(3)
      CALL CELS (CNT,P1,P2,IER1,IER2)
      Y(1)=W
```

```
      CNT=CNT-TWO*(THETA(3)-ONE)*LOG(Y(1))
      RETURN
      END
```

The first individual's data follows.

```
id,dv
1,223.28
1,41.59
1,87.6
1,72.11
1,216.32
1,118.1
1,232.29
```

REFERENCES: Guide Introduction_7

CONTROL3 EXAMPLE

This is file CONTROL3 from the NONMEM distribution medium.

```
$PROBLEM  THEOPHYLLINE   SINGLE SUBJECT DATA
$INPUT  DOSE=AMT TIME CP=DV
$DATA  DATA3
$SUBROUTINES  ADVAN2

$PK
CALLFL=1
KA=THETA(1)
K=THETA(2)
SC=THETA(3)

$ERROR
Y=F+ERR(1)

$THETA  (0,1.7)  (0,.102)  (0,29)

$ESTIMATION  MAXEVAL=240  PRINT=2
$COVR
$TABLE TIME
$SCAT    CP VS TIME
$SCAT    PRED VS TIME
$SCAT    RES VS TIME
$SCAT    PRED VS CP  UNIT
```

The data is file DATA3 from the NONMEM distribution media.

```
    320    .0     .
     .     .27   1.71
     .     .52   7.91
     .    1.     8.31
     .    1.92   8.33
     .    3.5    6.85
     .    5.02   6.08
     .    7.03   5.4
     .    9.     4.55
     .   12.     3.01
     .   24.3     .90
```

REFERENCES: Guide III, section VI.1

CONTROL4 EXAMPLE

This is file CONTROL4 from NONMEM distribution media

```
$PROB  THEOPHYLLINE   POPULATION DATA
$INPUT      ID DOSE TIME CP=DV WT
$DATA       THEO

$PRED
;THETA(1)=MEAN ABSORPTION RATE CONSTANT (1/HR)
;THETA(2)=MEAN ELIMINATION RATE CONSTANT (1/HR)
;THETA(3)=SLOPE OF CLEARANCE VS WEIGHT RELATIONSHIP (LITERS/HR/KG)
;DOSE=WT-ADJUSTED DOSE (MG/KG)
;DS=NON-WT-ADJUSTED DOSE (MG)
   IF (DOSE.NE.0) THEN
      DS=DOSE*WT
      W=WT
   ENDIF
   KA=THETA(1)+ETA(1)
   KE=THETA(2)+ETA(2)
   CL=THETA(3)*W+ETA(3)
   D=EXP(-KE*TIME)-EXP(-KA*TIME)
   E=CL*(KA-KE)
   F=DS*KE*KA/E*D
   Y=F+EPS(1)

$THETA  (.1,3,5) (.008,.08,.5) (.004,.04,.9)
$OMEGA BLOCK(3)  6 .005 .0002 .3 .006 .4
$SIGMA  .4

$EST     MAXEVAL=450  PRINT=5
$COV
$TABLE           ID DOSE WT TIME
$SCAT            (RES WRES) VS TIME BY ID
```

The data is file THEO from the NONMEM distribution media. The first individual's data follows.

```
         1      4.02      0.          .74     79.6
         1       .        0.25      2.84       .
         1       .        0.57      6.57       .
         1       .        1.12     10.5        .
         1       .        2.02      9.66       .
         1       .        3.82      8.58       .
         1       .        5.1       8.36       .
         1       .        7.03      7.47       .
         1       .        9.05      6.89       .
         1       .       12.12      5.94       .
         1       .       24.37      3.28       .
```

REFERENCES: Guide III, section VI.3
REFERENCES: Guide IV, Appendix II

CONTROL5 EXAMPLE

This is file CONTROL5 from NONMEM distribution media

```
$PROB  THEOPHYLLINE   POPULATION DATA
$INPUT      ID DOSE=AMT TIME CP=DV WT
$DATA       THEOPP

$SUBROUTINES  ADVAN2

$PK
;THETA(1)=MEAN ABSORPTION RATE CONSTANT (1/HR)
;THETA(2)=MEAN ELIMINATION RATE CONSTANT (1/HR)
;THETA(3)=SLOPE OF CLEARANCE VS WEIGHT RELATIONSHIP (LITERS/HR/KG)
;SCALING PARAMETER=VOLUME/WT SINCE DOSE IS WEIGHT-ADJUSTED
   CALLFL=1
   KA=THETA(1)+ETA(1)
   K=THETA(2)+ETA(2)
   CL=THETA(3)*WT+ETA(3)
   SC=CL/K/WT

$THETA  (.1,3,5) (.008,.08,.5) (.004,.04,.9)
$OMEGA BLOCK(3)  6 .005 .0002 .3 .006 .4

$ERROR
   Y=F+EPS(1)

$SIGMA  .4

$EST     MAXEVAL=450  PRINT=5
$COV
$TABLE          ID DOSE WT TIME
$SCAT           (RES WRES) VS TIME BY ID
```

The data is file THEOPP from the NONMEM distribution media. The first individual's data follows.

```
         1      4.02      0.          .       79.6
         1       .        0.         .74       .
         1       .        0.25      2.84       .
         1       .        0.57      6.57       .
         1       .        1.12     10.5        .
         1       .        2.02      9.66       .
         1       .        3.82      8.58       .
         1       .        5.1       8.36       .
         1       .        7.03      7.47       .
         1       .        9.05      6.89       .
         1       .       12.12      5.94       .
         1       .       24.37      3.28       .
```

REFERENCES: Guide III, VI.3
REFERENCES: Guide IV, Appendix VI

CONTROL6 EXAMPLE

This is file CONTROL6 from NONMEM distribution media.
It is discussed in Guide IV, Appendix VIII.

```
$PROB  THEOPHYLLINE   POPULATION DATA
$INPUT      ID DOSE=AMT TIME CP=DV WT
$DATA       THEOPP
$SUBROUTINES  ADVAN7
$MODEL COMP=(DEPOT,INITIALOFF,DEFDOSE) COMP=(CENTRAL,DEFOBS,NOOFF)
$PK
;THETA(1)=MEAN ABSORPTION RATE CONSTANT (1/HR)
;THETA(2)=MEAN ELIMINATION RATE CONSTANT (1/HR)
;THETA(3)=SLOPE OF CLEARANCE VS WEIGHT RELATIONSHIP (LITERS/HR/KG)
;SCALING PARAMETER=VOLUME/WT SINCE DOSE IS WEIGHT-ADJUSTED
   CALLFL=1
   K12=THETA(1)+ETA(1)
   K20=THETA(2)+ETA(2)
   CL=THETA(3)*WT+ETA(3)
   S2=CL/K20/WT
$THETA  (.1,3,5) (.008,.08,.5) (.004,.04,.9)
$OMEGA BLOCK(3)  6 .005 .0002 .3 .006 .4
$ERROR
   Y=F+EPS(1)
$SIGMA  .4
$EST     MAXEVAL=450  PRINT=5
$COV
$TABLE           ID DOSE WT TIME
$SCAT            (RES WRES) VS TIME BY ID
```

The data is file THEOPP from the NONMEM distribution media. The first individual's data follows.

```
         1      4.02      0.          .       79.6
         1       .        0.         .74       .
         1       .        0.25      2.84       .
         1       .        0.57      6.57       .
         1       .        1.12     10.5        .
         1       .        2.02      9.66       .
         1       .        3.82      8.58       .
         1       .        5.1       8.36       .
         1       .        7.03      7.47       .
         1       .        9.05      6.89       .
         1       .       12.12      5.94       .
         1       .       24.37      3.28       .
```

REFERENCES: Guide III, VI.3
REFERENCES: Guide IV, Appendix VI , VIII

CONTROL7 EXAMPLE

This is file CONTROL7 from NONMEM distribution media
It is discussed in Guide IV, Appendix VIII.

```
$PROB  THEOPHYLLINE   POPULATION DATA
$INPUT      ID DOSE=AMT TIME CP=DV WT
$DATA       THEOPP
$SUBROUTINES  ADVAN6  TOL=5
$MODEL COMP=(DEPOT,INITIALOFF,DEFDOSE) COMP=(CENTRAL,DEFOBS,NOOFF)
$PK
;THETA(1)=MEAN ABSORPTION RATE CONSTANT (1/HR)
;THETA(2)=MEAN ELIMINATION RATE CONSTANT (1/HR)
;THETA(3)=SLOPE OF CLEARANCE VS WEIGHT RELATIONSHIP (LITERS/HR/KG)
;SCALING PARAMETER=VOLUME/WT SINCE DOSE IS WEIGHT-ADJUSTED
   CALLFL=1
   KA=THETA(1)+ETA(1)
   KE=THETA(2)+ETA(2)
   CL=THETA(3)*WT+ETA(3)
   S2=CL/KE/WT
$THETA  (.1,3,5) (.008,.08,.5) (.004,.04,.9)
$OMEGA BLOCK(3)  6 .005 .0002 .3 .006 .4
$DES
   DADT(1)=-KA*A(1)
   DADT(2)= KA*A(1)-KE*A(2)
$ERROR
   Y=F+EPS(1)
$SIGMA  .4
$EST     MAXEVAL=450  PRINT=5
$COV
$TABLE           ID DOSE WT TIME
$SCAT            (RES WRES) VS TIME BY ID
```

The data is file THEOPP from the NONMEM distribution media. The first individual's data follows.

```
         1      4.02      0.          .       79.6
         1       .        0.         .74       .
         1       .        0.25      2.84       .
         1       .        0.57      6.57       .
         1       .        1.12     10.5        .
         1       .        2.02      9.66       .
         1       .        3.82      8.58       .
         1       .        5.1       8.36       .
         1       .        7.03      7.47       .
         1       .        9.05      6.89       .
         1       .       12.12      5.94       .
         1       .       24.37      3.28       .
```

REFERENCES: Guide III, section VI.3
REFERENCES: Guide IV, Appendix VI , VIII

DATA AVERAGE EXAMPLE

A template record is a data record that contains a non-zero value for the raw-data (RAW_) data item. Only the DV values with those observation records (i.e., records with MDV=0) matching the template data record will be included in the average. This average replaces the DV item in the template record, and it may be displayed in a table or scatter-plot.

An observation record matches a template if all its data items are identical to all those on the template, except that the values of NONMEM data items do not affect the match. (Any nonNONMEM data item type may in fact be excluded from the match (See **omit**)).

Example 1

Here is a fragment of a data set (which may or may not actually be for PREDPP):

```
#ID  TIME  AMT  WT  AGE  EVID  MDV  DV  RAW_  GEND
 1   5     0    70  50   0     1    2   1      1    template record
 4   5     0    70  50   0     0    4   0      1    observation record
```

These two records match. The NONMEM items are the ID, MDV, DV, and RAW_ items. The DV item with the observation record will be included in the average. The DV item in the template record is ignored for the purpose of computing the average (MDV=1); the displayed DV item with this record will be the average value, and the displayed RES item will be the difference between this average and the PRED item. The average is taken over all records matching the TIME, AMT, WT, AGE, EVID, and GEND items of the template record.

If the omit record

```
$OMIT GEND
```

appears, then the two records

```
#ID  TIME  AMT  WT  AGE  EVID  MDV  DV  RAW_  GEND
 1   5     0    70  50   0     1    2   1      1    template record
 4   5     0    70  50   0     0    4   0      2    observation record
```

also match. In this case the average is taken over all records matching the TIME, AMT, WT, AGE, and EVID items of the template record.

Example 2

Here is an example of categorical dose-response with four different bolus doses per individual (separated by washouts), and data averages are formed for each of the four doses.

```
#ID  DOSE  DV  MDV  RAW_
 100 5     .   1    1     set DV item to avr. for DOSE=5
 100 10    .   1    1     set DV item to avr. for DOSE=10
 100 20    .   1    1     set DV item to avr. for DOSE=20
 100 40    .   1    1     set DV item to avr. for DOSE=40
 1   5     1   0    0
 1   10    1   0    0
 1   20    2   0    0
 1   40    2   0    0
 2   5     1   0    0
```

```
 2    10    1   0    0
 2    20    1   0    0
 2    40    1   0    0
 3    5     1   0    0
 3    10    2   0    0
 3    20    2   0    0
 3    40    3   0    0
 .
 .
```

Example 3

An average of a PRED-defined item may also be displayed.

```
$ABBREV COMRES=1 COMSAV=1
 ...
$PRED
 COM(1)=LOG(DV)
 ...

$TABLE COM(1)=LDV
```

For template records, the value of LDV displayed in the table is the average of LOG DV over all observation records matching the template.

(See **data average block, raw**).

REFERENCES: none.

ENTERHEPATIC CIRCULATION EXAMPLES

The help item "Model Time examples" gives a fragment of code for modelling EHC (Enterohepatic Circulation, which is also called Enterohepatic Recycling). It illustrates the use of MTIME parameters to model instantaneous changes in differential equations. This help item, "Enterhepatic circulation examples", describes two fully-worked out control streams in the examples directory. They demonstrate how to generalize the fragment to multiple sequential doses.

Both examples use the same data. (hillss.dat and mtimess.dat are identical). There is a steady-state bolus dose at time 0. The interdose interval II is 12, and there are enough additional doses (ADDL=100) to continue the dosing pattern throughout the data set. There are "other" records every 4 units till time 140 to allow compartment amounts to be displayed and there is a final observation record at time 144.

mtimess.ctl

This example incorporates the fragment into a complete control stream. MTIME parameters are used to turn on and off the EHC terms in the differential equations. The variable FLAG is 1 between times MTIME(1) and MTIME(2) after each dose event and turns on the EHC terms. After time MTIME(2) is reached, a new set of MTIME's is defined which affect the next dosing interval. MTIME parameters are not dose-related parameters and have no effect on steady-state dose events. Even if PK computes `MTIME(i)` < II, this produces future changes in the system, and does not apply retroactively to the preceding implied doses.

(See **Guide VI, Section V.F.4, Note 4**).

A steady-state dose record should not be used. Instead, the SS dose record is described as a transient dose with SS=DROP on the $INPUT record.

hillss.ctl

This control stream does not use MTIME. Instead, a smooth step model using Hill terms in a sigmoid emax model is used. The $DES code has to compute all the necessary variables. Flag1 and flag2 are continuous variables that change from 0 to 1 at the times corresponding to the MTIME's. The FLAG variable is similar to FLAG in mtimess.ctl. The changes to the differential equations are not instantaneous, but they are continuous. If the exponent in the Hill term is made larger, the predictions approach those of the MTIME model. However, very large values of the exponent can lead to numerical difficulties in PREDPP. Smaller values of the exponent may be more realistic physiologically. A Steady-State dose event record is used with this model.

Note that SS dose records should only be used when the kinetics implemented in the model coincides with the II (interdose interval) of the SS record. Just as MTIME's cannot affect the differential equations retroactively, changes to the differential equations that happen in the future cannot affect the Steady-state calculations. For example, with II=12, then the kinetics should not be different in the interval of time 0 to 12 vs. time 12 to 24 or time 24 to 36, etc. In each interval, changes occur at theta(8) and theta(8)+theta(9) after the start of the interval. $DES computes the first change time for each interval using the INT function:

```
mt1=II*INT(T/II)+theta(8)
```

NM-TRAN gives a warning about the use of the INT function in $DES:

```
 (WARNING  68) THE INT, MOD, MIN, OR MAX FUNCTION IS BEING USED OUTSIDE
OF A
 SIMULATION BLOCK. IF THE FUNCTION VALUE AFFECTS THE VALUE OF THE
 OBJECTIVE FUNCTION, THEN AN ERROR WILL PROBABLY OCCUR.
```

This warning may be disregarded. Discontinuties occur at the ends of the integration intervals, but the kinetics are unaffected. For example, when T<12, the value of 12*INT(T/12) is 0. The values of FLAG1 and FLAG2 are initially 0 and FLAG is 0. As T approaches the end point T=12, the values of FLAG1 and FLAG2 both become 1 and FLAG is 0. At the end point when T=12, the value of 12*INT(T/12) is 12 and both FLAG1 and FLAG2 are 0. The discontinuity in mt1 and mt2 does not affect the FLAG variable because FLAG1 and FLAG2 are both 0 or both 1 in the neighborhood of the discontinuity, and the kinetics are continuous.

In these examples, EHC is driven by the dose events. The EHC changes can also driven by the clock. Suppose every 12 hours, a new EHC cycle begins. (The value 12 is chosen so that the two versions will give the same predictions.) E.g., instead in mtimess.ctl, instead of

```
 MTIME(1) = MTIME(1)+II
```

compute

```
 MTIME(1) = MTIME(1)+12
```

In hillss.ctl, instead of

```
 mt1=inter*INT(T/inter)+theta(8)
```

compute

```
 mt1=12*INT(T/12)+theta(8)
```

There is no difference in the results.

REFERENCES: Guide VI, section III.F.9
REFERENCES: Guide VI, section V.F
REFERENCES: Guide IV, section V.C.5

EXAMPLES USING MTIME TO MODEL PERIODIC DISCONTINUITIES IN $DES

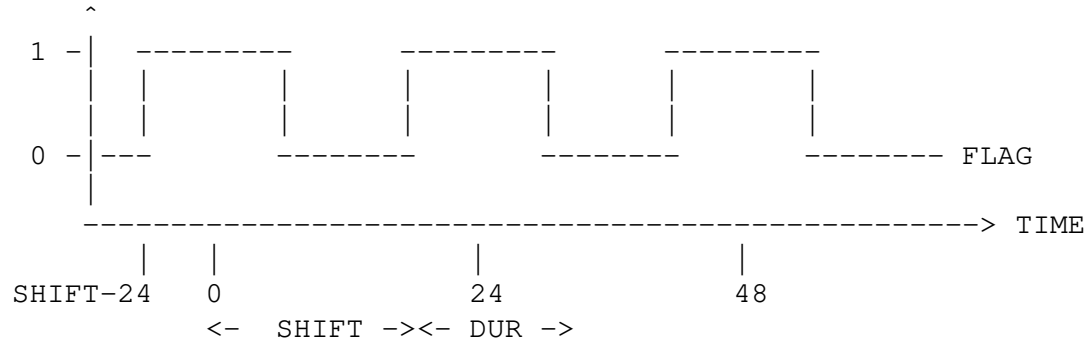
Some systems defined with differential equations can be discontinuous with respect to time. Discontinuities are typically introduced in the system by suddenly changing the value of one or more model variables at specific points of time. Such changes can be periodic due to e.g. a circadian rhythm. It is important to define discontinuous variables in $PK using model event time (MTIME) variables rather than updating them in $DES using IF-ELSE-ENDIF tests of T or using the discontinuous INT function. In fact, when integrating from time ta to time tb, routines such as DVERK (ADVAN6) may go slightly beyond time tb (i.e., it may happen that $DES is called at T>tb). The values of DADT should be continuous. If any element of DADT changes at T=b, this should happen at the next integration interval, integrating from time b to time c.

Two examples are provided for the implementation of periodic discontinuities using MTIME variables. The first (step_circexa.ctl) shows how to model the daily reset of a step function and illustrates how NONMEM update the MTIME and closely related MNOW variables at and in between event records. The second (idr_circexa.ctl) applies this step function in an indirect response model that describes a truncated sinusoidal baseline reponse. It also shows how values of T that range from 0 to infinity can be transformed into repeated values of 0-24h. All files may be found in the examples directory. These examples were suggested by Sebastien Bihorel, Luann Phillips and Jill Fiedler-Kelly.

The data set used in both examples is circadian.csv. This data set has 4 subjects with sparse data (one observation every 24 hours from time 0 to time 168) and 4 subjects with finely-spaced data (every 0.5 hours from time 0 to time 48). It is with the second group of subjects that scatters of the step function, variable T24, and indirect response vs. time can best be seen.
There are no dose event records (i.e. AMT=0 for all records) because the step function and indirect response are generated endogenously. If this model is incorporated in a larger model, dose records can be used for other compartments.

step_circexa.ctl

This example illustrates how to turn on a step function, FLAG, at a certain time of the day, SHIFT, turn it off after a certain duration, DUR, and repeat this every 24 hours.



The MTIME(1) and MTIME(2) variables respectively define the times at which FLAG is set to 1 and 0, such as MTIME(2)-MTIME(1)=DUR. After MTIME(2) is reached, both variables are incremented by 24, and the FLAG update process perpetuates itself every 24 hours. Compartment 1 is a dummy compartment

intended to monitor the pro- per update of the FLAG variable: the amount A1 starts at 0 and is incremented by DUR every day.

Because the FLAG variable is not necessarily updated at the time of event records, both $TABLE and WRITE statements within the ab- breviated code are used in step_circexa.ctl to report the values of the variables of interest. WRITE statements are used because lines in a table file are produced only for event records, but not for nonevent times such as Model times (MTIMES) at which the variables computed in $PK are of special interest here.

Besides MTIME(1) and MTIME(2), the following variables are of interest in step_circexa.ctl:
* MPAST(1) and MPAST(2): variables automatically updated based on MTIME(1) and MTIME(2), and used to set FLAG in $DES. MPAST(i)=0 until the call to PK subsequent to the one for which MNOW=i. At that call and until MTIME(i) is redefined, MPAST(i)=1.
* INTMTIME ("interval MTIME") is the value of MTIME(1) for the entire integration interval during which FLAG=1, ie from MTIME(1) up to and including the endpoint at MTIME(2). It is not specifically used in step_circexa.ctl, but will be in the next example. INTMTIME is computed and displayed here so that it can be discussed with the other variables.
* TSTATE is a reserved variable giving the time at which the current state vector (compartment amounts) was computed. It gives the time to which the system was most recently advanced.

The FLAG variable can be computed in either $PK or $DES, but the latter is preferred.

Background details:

Suppose that the values of TIME on the event records are t1 t2 t3 etc. and that PK is called with every event record and NONEVENT (ADDITIONAL AND LAGGED) DOSE TIMES AND AT MODEL TIMES.

Suppose there are k MTIME variables with values
t1 <= MTIME(1) <= MTIME(2) <= ... <= MTIME(k) <= t2

The interval [t1,t2] is integrated by smaller intervals
[t1, MTIME(1)], [MTIME(1), MTIME(2)] ... [MTIME(k), t2].

Calls to DES during any integration interval [ta, tb] will have ta <= T, but it may happen that T > tb. For calls to PK and ERROR, the sequence of calls for the record with TIME=t2 is:
* for each i from 1 to k: call PK with the record at MTIME(i)
(MNOW=i, TSTATE=MTIME(i-1)) (if i=1, TSTATE=t1)
* call PK with the record at t2
(MNOW=0; TSTATE=MTIME(k))
* call ERROR with the record at t2
(MNOW is set to 9 to identify the WRITE in $ERROR; TSTATE=t2)

The same calls to PK occur even if MTIME(k) happens to corres- pond to t2. In such a case, the call with MNOW=k precedes the call with MNOW=0, and the values in the table file are from the second call.

The lines of the file step_circexa.txt that corresponding to the second event record (TIME=24) for ID 1 (set with SHIFT=0.9088 and DUR=20.8385), are:

```
        MNOW    TIME    MT1     MT2     MP1    MP2    TSTATE INTMTIME
(1)    1.0000 24.0000  0.9088 21.7473 0.0000 0.0000  0.0000  0.9088
(2)    2.0000 24.0000 24.9088 45.7473 1.0000 0.0000  0.9088  0.9088
(3)    0.0000 24.0000 24.9088 45.7473 0.0000 0.0000 21.7473 24.9088
(4)    9.0000 24.0000 24.9088 45.7473 0.0000 0.0000 24.0000 24.9088
```

where MTi and MPi stand for MTIME(i) and MPAST(i).

At line 1, FLAG=0. This corresponds to the advance from time 0 to MTIME(1): no update of MTIME variables is performed.
At line 2, FLAG=1. This corresponds to the advance from MTIME(1) to MTIME(2), at which point the MTIME(1) and MTIME(2) variables are updated, but INTMTIME retains its value.
At line 3, FLAG=0. This corresponds to the advance from MTIME(2) to time 24, i.e. the call to PK with MNOW=0, in which INMTIME is updated to the new value of MTIME(1) defined at the previous call to PK.
At line 4, FLAG=0. This corresponds to the call to ERROR.

The corresponding line of the table file step_circexa_debug.tab is:

```
TIME        MT1        MT2        STS     INTMTIME
24.0000   24.9088   45.7473   21.7473   24.9088
```

Note that TSTATE values are:
0 (time of 1st. event record)
0.9088 after advance to MTIME(1)=0.9088
21.7473 after advance to MTIME(2)=21.7473
24.0000 after advance to t2=24.

There will be as many lines with MNOW=0.0 in step_circexa.txt, as there are lines in step_circexa_debug.tab. These lines will be consistent except for the values of MNOW.

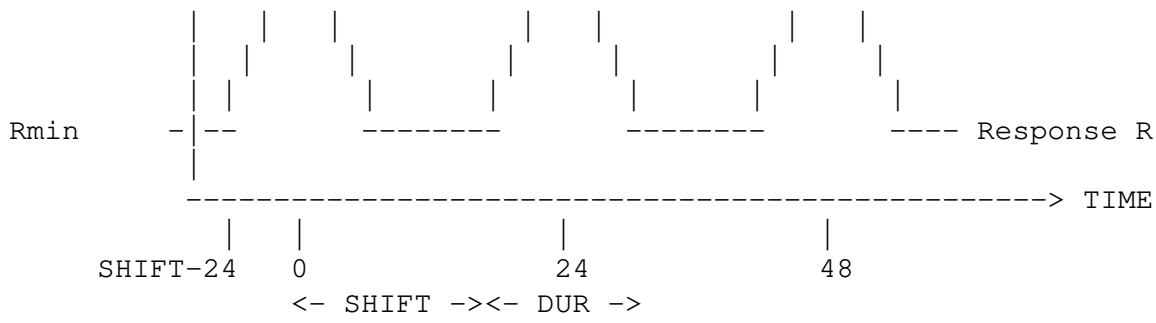
INMTIME retains the value of MTIME(1)=0.9088 that pertains to the entire integration interval, up to and including the end point at MTIME(2). It is set in $PK before the MTIME values are changed when MNOW=2. The values in the table are those set in $PK when MNOW=0 (the final call to PK with this data record) by which time INTMTIME and the MTIME(i) have their new values for the next set of advances.

In this particular example, with simulated etas, it sometimes happens that MTIME(1)<t1. From a biological point of view, the step function (and the response) likely started well before the data collection. The beginning of the data collection may occur with FLAG=1 (in the next example, this is during the oscillation of the response) or with FLAG=0 (during the flat part of the response). When MTIME(1) < t1, MTIME(1) is ignored during the set of calls to PK described above.

idr_circexa.ctl

This example extends the concepts introduced in step_circexa.ctl and illustrates how to apply the daily reset of the step function to model a process whose baseline is characterized by a sinusoid function during part of the day and a flat line for the rest of the day. It also show how to create a 24h clock time variable T24 in $DES by transforming the T variable ranging from 0 to infinity into repeating intervals of continuous time between 0 and 24.

```
              ^
 Rmin+Amp -|     ---             ---             ---
```



The baseline response depicted above can be described by:

$$R(t) = \begin{cases} Rmin + Amp*\sin(z(t)) & \text{for } t \text{ in } [SHIFT+n*24, SHIFT+DUR+n*24] \\ Rmin & \text{for any other value of } t \end{cases} \quad \text{(Eq.1)}$$

where

Rmin is the minimum response
Amp is the amplitude of the sinusoid function of time
z(t) is a circadian function of time which scales the time intervals [SHIFT+n*24, SHIFT+DUR+n*24] to [0, PI], that is an interval of x such that sin(x)>=0:
z(t) = PI * (t-(SHIFT+n*24))/DUR
n is (-1,0,1,2,...) if (SHIFT+DUR)>24
is (0,1,2,...) if (SHIFT+DUR)<=24

The case when SHIFT+DUR>24 is when the flag function is already 1 at time 0, that is it started before the first event. Therefore, an offset of 24 is needed.

z(t) is the function that drives the circadian rhythm of R and is implemented using the INTMTIME variable described in the previous example and that represents SHIFT+n*24. MTIME(1) and MTIME(2) are defined as in step_circexa.ctl to periodically update INTMTIME and the FLAG variable used later in the $DES block.

The baseline function, R(t), can be fitted using Eq.1 in absence of drug effect. However, some biological processes which follow such circadian rhythms can be influenced by drugs (eg, metabolic effects of corticoids). If the effect of the drug is not direct, one can express the biological response and the drug effect in terms of an indirect response model (IDR). Assuming that the drug stimulates the formation of the biological response, one can parameterize the system using the following differential equation:

$$dR/dt = KIN(t)*(1+S(Cp)) - KOUT*R \quad \text{(Eq.2)}$$

where

KOUT is the elimination rate of the biological response and is stationary
KIN(t) is the formation rate of the biological response and is time-varying dependent on the step function FLAG:
FLAG = MPAST(1)-MPAST(2)
S(Cp) is a stimulation function of the drug concentration.

In absence of drug, there is no stimulation (i.e., S(Cp) = 0) and the baseline response R(t) described in Eq.1 can be reparameterized as follows:

$$dR/dt = KIN(t) - KOUT*R \quad \text{(Eq.3)}$$

The model is defined such as, when FLAG=0, KIN(t) is constant and equal to KOUT*Rmin, and such as, when FLAG=1, KIN(t) has a sinusoidal shape, increasing from its minimum value to its maximum, and decreasing back to its minimum within the duration DUR. One can obtain the explicit expression of KIN(t) within the time when FLAG=1, by determining the first derivative of R(t), as defined in the first part of Eq.1 and solving for KIN(t) using Eq.3. This expression of KIN(t) is used in $DES.

The $DES block also includes an independent line of code meant to define a continuous 24h clock time. This code is supported by the definition and circadian reset of the MTIME(3) variable in $PK. More specifically, this code is dependent on INTMTIME3, the value of MTIME(3) for the 24-hour intervals.

Note: Original idea for this example stems from Andreas Krause's post on the NMUsers email list dated from 5/26/2011, subject "Coding INTEGER Function in NONMEM". The original code was modified to use model event time variables.

(See **mtime**).

REFERENCES: None.

EXOGENOUS SUPPLEMENTATION EXAMPLE

DISCUSSION:

In this example, an oral "drug" is given exogenously, and it also exists as an endogenous substance. There is an unknown dosing history prior to the observation period (i.e., prior to time zero). This example illustrates how three sources of drug can be modeled: pre-existing endogenous drug, pre-existing drug from an unknown prior dosing history, and drug from known doses. Any combination of the three could be modeled without the others.

The rate of endogenous drug production is assumed to be constant, with no feedback control of production. Thus endogenous drug is at steady-state, and, with linear kinetics, its effect is simply to add a constant increment to exogenous drug in the sampled compartment (the increment is modeled as theta(7)).

For the drug with unknown dosing history, it is assumed that the subject is at steady state with respect to this drug. This part of total drug is modeled by a steady state infusion dose into the depot compartment, ending at time 0, and having an unknown rate. The result of the SS dose is to introduce drug into all compartments of the system (not just the central compartment) because it is distributed throughout the system and is subject to elimination from the system. The unknown rate is modeled as theta(5). NONMEM will adjust theta(5) to best fit not only the "baseline" observation at time 0, but also the later observations.

Note that if samples are not taken sufficiently long after the time of the last dose (> 4 half-lives), then theta(7) and theta(5) may not be separately identifiable. Note that the value of theta(7) may be determined by the residual concentration after all exogenous drug has disappeared.

A combined additive and ccv error model is used. Theta(8) is the ratio of the C.V. of the proportional component to the standard deviation of the additive component.

Any ADVAN/TRANS combination could be used. Population data could also be modeled in this manner, with eta variables in the $PK block.

```
$PROBLEM Example of pre-existing drug.
$INPUT ID TIME DV AMT SS II RATE
$DATA DATA1
$SUBROUTINES   ADVAN4 TRANS5
$PK
  AOB=THETA(1)
  ALPHA=THETA(2)
  BETA=THETA(3)
  KA=THETA(4)
  R1=THETA(5)
  S2=THETA(6)
$ERROR
  FP=THETA(7)+F  ; adds endogenous component
  W=(1+THETA(8)*THETA(8)*FP*FP)**.5
  Y=FP+W*ERR(1)
```

Note that, if there are other doses into the depot compartment with modeled rates, it is necessary to assign a value to R1 conditionally. E.g.,

```
IF (TIME.EQ.0) THEN
   R1=THETA(5)  ; rate for SS infusion record at time 0
ELSE
   R1=....; rate of other kind of dose
ENDIF
```

Note also that the combined additive and ccv error model can also be modeled using two random variables:

```
Y = F*(1+ERR(1)) + ERR(2)
```

A fragment of the data follows. Record 1 specifies the SS infusion for the pre-existing drug, which ends at time 0. Record 2 gives the baseline observation. Record 3 specifies an oral bolus dose. Record 4 gives an observation.

```
       1        0        .       0        1        0       -1
       1        0     62.2       .        .        .        .
       1     0.01        .      95        .        .        .
       1     0.50   235.93       .        .        .        .
```

REFERENCES: Guide V, section 8
REFERENCES: Guide VI, section III.F.5

EXPECTATION EXAMPLE

These are examples of the use of the Marginal (MRG_) data item. Example 1

Suppose that the probability that a particular subject experiences a pain relief score of 2 is computed. Suppose also one wants to compute the (posterior population) expectation of the probability with each of 4 different bolus doses, not all of which are among those used to obtain observations. A fragment of the control file follows.

```
$INPUT ID DOSE MDV MRG_ ...

$PRED
 ....
 Y = likelihood of observation given ETA

$EST METH=COND LAPLACE LIKELIHOOD
$TABLE DOSE
```

A fragment of the data follows, with comments following ";".

```
#ID DOSE DV MDV MRG_
 100 5    2  1   1    ; PRED item is set to expectation for DOSE=5
 100 10   2  1   1    ; PRED item is set to expectation for DOSE=10
 100 20   2  1   1    ; PRED item is set to expectation for DOSE=20
 100 40   2  1   1    ; PRED item is set to expectation for DOSE=40
 1   3    1  0   0
 1   10   1  0   0
 1   25   2  0   0
 1   30   2  0   0
 2   3    1  0   0
 2   10   1  0   0
 2   25   1  0   0
 2   30   1  0   0
 3   3    1  0   0
 3   10   2  0   0
 3   25   2  0   0
 3   30   3  0   0
 ... etc ...
```

Example 2

This example produces a plot of four residuals, formed by the differences between the raw-data-averages and their (posterior population) expectations, versus the doses used to obtain the data.

```
$INPUT ID DOSE MDV MRG_ RAW_
$PRED
 ...
IF (ICALL.EQ.5) THEN
  Y = expectation of observation given ETA
ELSE
  Y = likelihood of observation given ETA
ENDIF
$EST METHOD=COND LAPLACE LIKELIHOOD
```

```
$SCAT RES VS DOSE

#ID DOSE DV MDV MRG_ RAW_
 100 5    .  1   1    1
 100 10   .  1   1    1
 100 20   .  1   1    1
 100 40   .  1   1    1
 1   5    1  0   0    0
 1   10   1  0   0    0
 1   20   2  0   0    0
 1   40   2  0   0    0
 ... etc ...
```

Example 3

This example produces a plot of four residuals, formed by the differences between the proportion of subjects in the data set with pain relief score 2 and the (posterior population) expectation of the probability that a subject experiences a pain score of 2, versus the doses used to obtain the data.

```
$INPUT ID DOSE MDV MRG_ RAW_
$PRED
 ...
  Y = likelihood of observation given ETA
  IF (ICALL.EQ.6) THEN
     DVR=DV
     DV=0
     IF (DVR.EQ.2) DV=1
   ENDIF
$EST METHOD=COND LAPLACE LIKELIHOOD
$SCAT RES VS DOSE

#ID DOSE DV MDV MRG_ RAW_
 100 5    2  1   1    1
 100 10   2  1   1    1
 100 20   2  1   1    1
 100 40   2  1   1    1
 1   5    1  0   0    0
 1   10   1  0   0    0
 1   20   2  0   0    0
 1   40   2  0   0    0
 2   5    1  0   0    0
 2   10   1  0   0    0
 2   20   1  0   0    0
 2   40   3  0   0    0
 3   5    1  0   0    0
 3   10   2  0   0    0
 3   20   2  0   0    0
 3   40   3  0   0    0
 ... etc ...
```

(See **mrg**, **expectation block**, **data average block**, **raw**).

REFERENCES: none.

FINALIZATION EXAMPLE ($PRED AND $INFN)

This example contains abbreviated code which can be inserted in a $PRED or $INFN block. This code outputs final parameter estimates, standard errors, minimum value of the objective function, and conditional estimates of etas to various user files. The return codes from Estimation and Covariance steps (zero for normal termination) are also output.

```
IF (ICALL.EQ.3) THEN
  DO WHILE(DATA)
  IF (NEWIND.LE.1) WRITE (50,*) ETA
  ENDDO
  WRITE (51,*) OBJECT
  WRITE (52,*) THETA
  WRITE (52,*) THETAFR
  WRITE (53,*) SETHET
  WRITE (53,*) SETHETR
  WRITE (54,*) OMEGA(BLOCK)
  WRITE (55,*) SEOMEG(BLOCK)
  WRITE (56,*) SIGMA(BLOCK)
  WRITE (57,*) SESIGM(BLOCK)
  WRITE (58,*) IERE,IERC
ENDIF
```

REFERENCES: Guide VI, section VI.A

FINEDATA EXAMPLE

This example is new to NONMEM 7. It is an example of the use of the finedata utility program to fill in missing values of an independent variable (covariate).

Other examples (infn1, infn2) show the use of the INFN routines for interpolation. In data file THEOPP used for CONTROL5, where only one value of WT is present, the INFN routine is used to fill in missing values of WT. Presumably these examples would be part of a NONMEM run involving Estimation.

In this example, finedata is used to fill in the missing values of WT and create a new data set, THEOPPfull. This data set can be used in subsequent NONMEM runs for estimation.

USAGE:

```
finedata theofine.ctl
```

Contents of theofine.ctl are as follows:

```
$PROB  THEOPHYLLINE   POPULATION DATA
$INPUT       ID DOSE=AMT TIME CP=DV WT
$DATA        THEOPP
$FINEDATA  AXIS=TIME FILE=THEOPPfull WT=LIN MISSING=. NEVAL=-1
```

(See **control5**).
(See **infn1 example**).
(See **infn2 example**).

REFERENCES: Guide Introduction_7

FOCE PARALLEL EXAMPLE

The example control stream file foce_parallel.ctl is in the ..\examples directory.

```
; Used for comparing single versus parallel computing for FOCE method.
;$SIZES LVR=30
;$SIZES LTH=15
;$SIZES LIM1=100
$PROB RUN# Example 1 (from samp5l)
$INPUT C SET ID JID TIME  DV=CONC AMT=DOSE RATE EVID MDV CMT CLX
       V1X QX V2X SDIX SDSX
$DATA example1.csv IGNORE=C

$SUBROUTINES ADVAN3 TRANS4

$PK
MU_1=THETA(1)
MU_2=THETA(2)
MU_3=THETA(3)
MU_4=THETA(4)
CL=DEXP(MU_1+ETA(1))
V1=DEXP(MU_2+ETA(2))
Q=DEXP(MU_3+ETA(3))
V2=DEXP(MU_4+ETA(4))
S1=V1

$ERROR
Y = F + F*EPS(1)

; Initial values of THETA
$THETA
(0.001, 2.0) ;[LN(CL)]
(0.001, 2.0) ;[LN(V1)]
(0.001, 2.0) ;[LN(Q)]
(0.001, 2.0) ;[LN(V2)]
;INITIAL values of OMEGA
$OMEGA BLOCK(4)
0.15   ;[P]
0.01  ;[F]
0.15   ;[P]
0.01  ;[F]
0.01  ;[F]
0.15   ;[P]
0.01  ;[F]
0.01  ;[F]
0.01  ;[F]
0.15   ;[P]
;Initial value of SIGMA
$SIGMA
(0.6 )   ;[P]
```

```
;$EST METHOD=CHAIN FILE=foce_parallel.chn ISAMPLE=3 NSAMPLE=0
$EST METHOD=1 INTERACTION NSIG=3 PRINT=1
     NOABORT FORMAT=s1PE23.16 MAXEVAL=9999 ; PARAFILE=parallel_file.pnm
$COV MATRIX=S PRINT=E UNCONDITIONAL ; PARAFILE=OFF
```

REFERENCES: Guide Introduction_7

IGNORE ACCEPT EXAMPLE

The options IGNORE=(list) and ACCEPT=(list) of the $DATA record allow for simple selection criteria for records that are to be excluded or included, respectively, in the NONMEM data set. A list has the format:
cond,cond,cond,...
where each cond is a test of a data item: "label operator value". The commas are implied .OR. operators. The .AND. and .NOT. operators are not permitted.

Suppose an .AND. is needed. E.g.,

```
IGNORE=(GEN.EQ.1.AND.AGE.GT.60)
```

If the expressions are simple, this may be done by negating both sides:

```
.NOT.IGNORE =.NOT.(GEN.EQ.1.AND.AGE.GT.60)
ACCEPT=.NOT.GEN.EQ.1.OR..NOT.AGE.GT.60
```

Thus,

```
ACCEPT=(GEN.NE.1,AGE.LE.60)
```

But logical operators inside parenthesis are not permitted, so that the following cannot be coded:

```
$DATA ACCEPT=((A==1.OR.A==2).AND.B<100)
```

Even if negated, this is:

```
IGNORE=(.NOT.(A==1.OR.A==2)).OR..NOT.B<100)
```

which is:

```
IGNORE=((A.NE.1.AND.A.NE.2).OR.B>=100)
```

There is still an .AND. inside parentheses.

There is a workaround that takes advantage of the fact that A takes numeric values and there are no values between 1 and 2.

```
(A.NE.1.AND.A.NE.2)
```

can be implemented with an "or":

```
(A.LT.1.OR.A.GT.2).
```

The following are equivalent:

```
IGNORE=(B.GE.100)  IGNORE=(A.LT.1) IGNORE=(A.GT.2)
IGNORE=(B.GE.100,A.LT.1,A.GT.2)
```

This workaround cannot handle continuous values of A or more complicated conditions.

With NONMEM through 7.3, two separate NONMEM problems can be used, in two separate runs, for a complicated condition. In the first problem, abbreviated code is used to append a new variable to the data with a value indicating ignore/accept; in the second run the variable is used in the $DATA statement to choose from the data set.

Suppose ACC is the variable that is to have values 0/1 for ignore/accept. The desired code is:

```
ACC=0
IF ((A == 1.OR.A == 2).AND.B<100) ACC=1
```

The above can be coded using the FORTRAN language. But NM-TRAN abbreviated code is a subset of FORTRAN. Logical operators .NOT., .AND., .OR. may not be used within parentheses. There are two workarounds.

One is to clear the parentheses:

```
IF (A==1.AND.B<100.OR.A==2.AND.B<100) ACC=1
```

(this is always possible, no matter how complicted the conditional expression).

The second is to use several statements. There are may ways to do this. In the following, the .AND. of multiple conditions is false if any of the conditions is false.

```
ACC=1
IF (A.NE.1.AND.A.NE.2) ACC=0
IF (B.GE.100) ACC=0
```

In prob1.ctl, the first problem appends ACC to the (new) data set. In prob2.ctl, the $DATA record uses ACC to select records.

All required data items for NONMEM (and PREDPP, if used) must be present in the original data file testab.dat. The control stream for the first run:

```
$PROB test of accept Run #1
 ; Example of implementation of
 ; $DATA ACCEPT=((A == 1.OR.A == 2).AND.B<100)
 $INPUT      NO A B ID DV MDV
 $DATA       testab.dat IGNORE C
 $PRED
 ; Implements:
 ; ACC=1 IF ((A == 1.OR.A == 2).AND.B < 100)
    ACC=1
    IF (A.NE.1.AND.A.NE.2) ACC=0
    IF (B.GE.100) ACC=0
 ; The model for this run is unimportant. Keep it simple.
    Y=THETA(1)+ETA(1)+EPS(1)
 $THETA  1
 $OMEGA 1
 $SIGMA  .4
 $TABLE NO A B ID DV MDV ACC
  NOPRINT NOAPPEND NOHEADER FILE=run1.tab
```

The control stream for the second run:

```
 $PROB test of accept Run #2
 $INPUT NO A B ID DV MDV ACC
 ; Must do a numeric test for ACC==1, because it appears in
 ; table file run1.tab as 1.0000E+00
 $DATA run1.tab ACCEPT=(ACC.EQN.1)
 $PRED
 ; the model in this problem is the model
 ; used for simulation, analysis, etc.
  Y=THETA(1)+ETA(1)+EPS(1)
 $THETA 1
 $OMEGA 1
 $SIGMA .4
 $TABLE NO A B ID DV MDV
 NOPRINT NOAPPEND NOHEADER FILE=run2.tab
```

With NONMEM 7.4, it is possible to combine both problems into one NONMEM run using the $TABLE record option "EXCLUDE_BY list". If any variable in list is non-0, the record is excluded from the table file probe.tab. The $DATA ACCEPT/IGNORE option is not needed. Because probe.tab does not exist when NM-TRAN processes the second problem in the control stream, $DATA .. NOOPEN is needed and a format specification is needed.

```
 $PROB  test of NONMEM 7.4 exclude_by feature
 $INPUT      NO A B ID DV MDV
 $DATA       testab.dat IGNORE C
 $PRED
 ; sets ACC=0 if (a==1.or.a==2).and.b<100)
 ; uses the clear the parenthesis approach.
 ACC=1
 IF (A==1.AND.B<100.OR.A==2.AND.B<100) ACC=0
 ; The following is the model for both problems.
 ; It may be more complicated than this.
    Y=THETA(1)+ETA(1)+EPS(1)
 $THETA  1
 $OMEGA 1
 $SIGMA  .4
 ; probe.tab has only those records that have ACC=0.
 $TABLE NO A B ID  DV MDV EXCLUDE_BY ACC
  NOPRINT NOAPPEND NOHEADER FILE=probe.tab

 $PROB test of exclude_by #2
 $INPUT NO A B ID DV MDV
 $DATA probe.tab (10F12.0) NOOPEN
 $THETA 1
 $OMEGA 1
 $SIGMA .4
 ; This problem should have $ESTIMATION and other tasks
$TABLE NO A B ID DV NOAPPEND NOHEADER NOPRINT FILE=probe2.tab
```

Here is a fragment of the data testab.dat

```
C NO A B    ID DV MDV
  1 1 100    1  0  0
  2 2 100    1  0  0
  3 3 100    1  0  0
  4 1 101    1  0  0
  5 2 101    1  0  0
  6 3 101    1  0  0
  7 1  99    1  0  0
  8 2  99    1  0  0
  9 3  99    1  0  0
```

(See **$data, $table**).

REFERENCES: none.

INFN INTERPOLATION EXAMPLE 1

This example is adapted from Guide VI (PREDPP), Figure 37.
It was revised for NONMEM 7.3. Earlier versions were untested and did not work correctly. This version allows both single-subject and population data. It requires a "missing independent variable" data item.

In this example, the INFN subroutine is used to fill in (interpolate) missing values of WT in the THEOPPMV data set. THEOPPMV is a version of the data set THEOPP used in CONTROL5,
(See **control5**).

If WT is recorded on every record, then the $PK block can be simpler. The $PK block can be called with every record (no need for CALLFL=1) and no special code is needed to save the value of WT for subsequent records.

Example infn2 shows how the INFN routine may be implemented in abbreviated code. Example fine1 shows how the finedata utility program can be used to fill in the values without the use of the INFN routine.

USAGE:

```
$PROB  THEOPHYLLINE   POPULATION DATA
$INPUT      ID DOSE=AMT TIME CP=DV WT MISS
$DATA       THEOPPMV
$SUBROUTINES  ADVAN2 INFN=infnsub
```

The file infnsub contains the following:

```
! THIS EXAMPLE IS ADAPTED FROM GUIDE VI (PREDPP), FIGURE 37.
! FOR HELP FILE infn1.exa
! MODIFIED BY AJB 9/2012
 ! INFN ROUTINE FOR COMPUTING LINEARLY INTERPOLATED VALUES
 ! OF AN INDEPENDENT VARIABLE V.    ILLUSTRATES USE OF ROUTINE PASS.
 ! USE FOR SINGLE-SUBJECT OR POPULATION DATA
 ! WORKS WITH 1 OR MORE NON-MISSING VALUES.
 ! A SINGLE SUCH VALUE MAY BE ANYWHERE IN DATA SET.
 ! DATREC(UI)=TIME DATA ITEM
 ! DATREC(VI)=INDEPENDENT VARIABLE DATA ITEM
 ! DATREC(MI)=MISSING INDEPENDENT VARIABLE DATA ITEM
 !   =0 INDEP VAR NOT MISSING
 !   >0 IF THIS DATA RECORD HAS MISSING INDEP VAR AND:
 !   =1    THIS DATA RECORD PRECEDES FIRST NON-MISSING VALUE
 !   =3    THIS DATA RECORD FOLLOWS LAST NON-MISSING VALUE
 !   =2    THIS DATA RECORD IS BETWEEN NON-MISSING VALUES
 !
       SUBROUTINE INFN (ICALL,THETA,DATREC,INDXS,NEWIN)
       USE SIZES, ONLY: ISIZE,DPSIZE,NO,MAXIDS
! NOTE THAT NEWIND IS NOT OBTAINED FROM THE SUBROUTINE ARGUMENT
       USE NMPRD_INT, ONLY: NEWIND=>NWIND
       INTEGER(KIND=ISIZE), INTENT(IN) :: ICALL,INDXS(*),NEWIN
       REAL(KIND=DPSIZE),   INTENT(IN) :: THETA(*)
       REAL(KIND=DPSIZE),   INTENT(IN OUT) :: DATREC(*)
       REAL(KIND=DPSIZE) :: U,V
```

```
      DIMENSION U(NO,MAXIDS),V(NO,MAXIDS),DEPVAR(MAXIDS),LASTI(MAXIDS)
      INTEGER UI,VI,MI,IS,LASTI
! IMPORTANT: CHANGE UI, VI, MI ACCORDING TO LAYOUT OF DATA SET
      UI=3
      VI=5
      MI=6
!
      IF (ICALL.NE.1) RETURN
      I=0
      IS=0
 ! INITIALIZE PASS
      MODE=0
      CALL PASS (MODE)
      MODE=2
 ! PASS THROUGH DATA
    5 CALL PASS (MODE)
      IF (MODE.EQ.0) GO TO 10
      IF (NEWIND<2) THEN
        I=0
        IS=IS+1
      ENDIF
 ! IF INDEP VAR IS PRESENT, STORE TIME AND VALUE
      IF (DATREC(MI).EQ.0.) THEN
         I=I+1
         U(IS,I)=DATREC(UI)
         V(IS,I)=DATREC(VI)
         LASTI(IS)=I
      ENDIF
      IF (I == 1) THEN   ! SAVE VALUE IN CASE ONLY RECORDED ONCE
         DEPVAR(IS)=V(IS,1)
      ENDIF
      GO TO 5
 ! INITIALIZE PASS A SECOND TIME
   10 I=0
      IS=0
      MODE=0
      CALL PASS (MODE)
      MODE=2
 ! PASS THROUGH DATA A SECOND TIME
   15 CALL PASS (MODE)
      IF (MODE.EQ.0) RETURN
      IF (NEWIND<2) THEN
        I=0
        IS=IS+1
      ENDIF
! IF INDEP VAR IS MISSING AND ONLY RECORDED ONCE, COPY IT
      IF (DATREC(MI).NE.0. .AND. LASTI(IS) == 1 ) THEN
       DATREC(VI)=DEPVAR(IS)
       GO TO 15
      ENDIF
```

```
 ! IF INDEP VAR IS MISSING, STORE INTERPOLATED VALUE
       IF (DATREC(MI).EQ.0.) THEN
         I=I+1
       ELSE
       IF (DATREC(MI).EQ.1.) THEN   ! EXTRAPOL. FROM FIRST 2 VALUES
         K=1
         L=2
       ELSEIF (DATREC(MI).EQ.2.) THEN  ! INTERPOL. FROM BEFORE AND AFTER
         K=I
         L=I+1
       ELSEIF (DATREC(MI).EQ.3.) THEN   ! EXTRAPOL. FROM LAST 2 VALUES
       K=I-1
       L=I
       ENDIF
       B=(V(IS,K)-V(IS,L))/(U(IS,K)-U(IS,L))
       DATREC(VI)=V(IS,K)+B*(DATREC(UI)-U(IS,K))
       ENDIF
       GO TO 15
       END
```

Data file THEOPPMV is identical to THEOPP, but has an extra data item MISS (Missing Dependent Variable). The data for the first subject is:

```
  1      4.02      0.          .       79.6   0
  1       .        0.         .74        .    3
  1       .        0.25      2.84        .    3
  1       .        0.57      6.57        .    3
  1       .        1.12     10.5         .    3
  1       .        2.02      9.66        .    3
  1       .        3.82      8.58        .    3
  1       .        5.1       8.36        .    3
  1       .        7.03      7.47        .    3
  1       .        9.05      6.89        .    3
  1       .       12.12      5.94        .    3
  1       .       24.37      3.28        .    3
```

(See **infn2 example**).
(See **fine1 example**).

REFERENCES: Guide VI, section VI.A , Figure 37

INFN INTERPOLATION EXAMPLE 2

This example is new to NONMEM 7. It is an improved version of infn1.exa. In this example, instead of using a stand-alone INFN subroutine, $INFN abbreviated code is used to fill in missing values of WT in the THEOPP data set.
(See **control5**).

If WT is recorded on every record, then the $PK block can be simpler. The $PK block can be called with every record (no need for CALLFL=1) and no special code is needed to save the value of WT for subsequent records.

No "missing independent variable" data item is needed, so the THEOPP data file can be used as-is. The $ABBR abbreviated code is located in file infnabbr for convenient re-use. This is also an example of the $ABBR REPLACE feature, which allows easy replacement of character strings in the included code, and is also an example of declared subscripted variables.

Example fine1 shows how the finedata utility program can be used to fill in the values without the use of the INFN routine.

USAGE:

```
$PROB  THEOPHYLLINE   POPULATION DATA
$INPUT      ID DOSE=AMT TIME CP=DV WT
$DATA       THEOPP
$SUBROUTINES  ADVAN2
$ABBR REPLACE INTVBL=WT
$ABBR REPLACE NULLVAL=0.0
 INCLUDE 'infnabbr'
```

The file infnabbr contains the following:

```
; INFNABBR: ABBREVIATED CODE FOR HELP infn2.exa
; THIS INFN BLOCK INTERPOLATES AND EXTRAPOLATES VALUES OF A SINGLE
; DATA ITEM CALLED INTVBL.
; BY DEFAULT, THE AXIS VARIABLE IS CALLED TIME.
; THE CHARACTER STRING NULLVAL DENOTES A NULL VALUE.
;
; IF THERE IS ONLY ONE NON-NULL VALUE, THIS VALUE IS COPIED TO
; ALL NULL VALUES. OTHERWISE:
; LINEAR INTERPOLATION IS PERFORMED FOR NULL VALUES THAT
; LIE BETWEEN TWO NON-NULL VALUES.
; LINEAR EXTRAPOLATION IS PERFORMED FOR VALUES BEFORE THE
; FIRST NON-NULL VALUE
; LINEAR EXTRAPOLATION IS PERFORMED FOR VALUES AFTER THE
; LAST NON-NULL VALUE
; INFNABBR CAN BE USED WITH BOTH SINGLE
; SUBJECT AND POPULATION DATA SET.
; WRITTEN BY AJB 9/2012
;
; TYPICAL USAGE:
; $ABBR REPLACE INTVBL=WT
; $ABBR REPLACE NULLVAL=0.0
; $ABBR REPLACE TIME=T  (ONLY IF T IS THE AXIS VARIABLE RATHER THAN TIME)
```

```
;  INCLUDE 'INFNINTERP'
;
;
$ABBR DECLARE U(MAXIDS,NO), V(MAXIDS,NO)
$ABBR DECLARE INTEGER IS
$ABBR DECLARE INTEGER I, INTEGER J, INTEGER L
$ABBR DECLARE IVAL(NO)
$INFN
  ; $INFN FOR COMPUTING INTERPOLATED VALUES OF AN
  ; INDEPENDENT VARIABLE NAMED INTVBL
  ; E.g., INTVBL MIGHT BE WEIGHT
  ; ASSUME THAT INTVBL = NULLVAL  WHEN MISSING.
  ; IS=SUBJECT #
  ; IVAL(IS)= # OF NON-NULL VALUES OF THE INDEP VAR. FOR SUBJECT IS
  ; VI(IS,I) = ITH. NON-NULL VALUE OF THE INDEPENDENT VARIABLE
  ; UI(IS,I) = TIME OF THE ITH. NON-NULL VALUE OF THE INDEPENDENT VARIABLE

   IS=0
   IF (ICALL.EQ.1) THEN
; FIRST PASS.  SAVE VALUES OF TIME & INDEP VAR. WHEN INDEP VAR. IS NON-NULL
        DO WHILE (DATA)
        IF (NEWIND < 2) THEN  ; INITIALIZE NEW INDIVIDUAL
          IS=IS+1
          IVAL(IS)=0
        ENDIF
        IF (INTVBL /= NULLVAL) THEN     ; SAVE NON-NULL VALUE
           IVAL(IS)=IVAL(IS)+1
           I=IVAL(IS)
           U(IS,I)=TIME
           V(IS,I)=INTVBL
        ENDIF
      ENDDO
; SECOND PASS
        IS=0
        DO WHILE (DATA)
        IF (NEWIND < 2) THEN  ; INITIALIZE NEW INDIVIDUAL
          IS=IS+1
          I=0
        ENDIF
; IF INDEP VAR IS MISSING AND WAS ONLY NON-NULL ONCE, COPY IT
        IF (INTVBL == NULLVAL .AND. IVAL(IS) == 1) THEN
            INTVBL=V(IS,1)
        ELSE
         IF (INTVBL /= NULLVAL) THEN  ; COUNT ANOTHER NON-NULL VALUE
            I=I+1
         ELSE  ; CURRENT RECORD HAS NULL VALUE
            IF (I==0) THEN ; EXTRAPOLATE FROM FIRST 2 VALUES
               J=1
               L=2
            ELSEIF (I>0 .AND. I<IVAL(IS)) THEN ; INTERPOL. BEFORE AND AFTER
```

```
               J=I
               L=I+1
            ELSEIF (I==IVAL(IS)) THEN  ; EXTRAPOL. FROM LAST 2 VALUES
              J=I-1
              L=I
            ENDIF
            SLOPE=(V(IS,J)-V(IS,L))/(U(IS,J)-U(IS,L))
            INTVBL=V(IS,J)+SLOPE*(TIME-U(IS,J))
          ENDIF
        ENDIF
        ENDDO
   ENDIF
```

(See **control5**).
(See **infn1 example**).
(See **fine1 example**).

REFERENCES: Guide VI, section VI.A , Figure 37
REFERENCES: Guide Introduction_7

INITIAL STEADY STATE EXAMPLE

This is an example of the use of the Initial Steady State (I_SS) reserved variable in $PK. It was provided by Nick Holford, Dept Pharmacology & Clinical Pharmacology, University of Auckland.

Glucose and insulin are both endogenous substances, whose rates of formation are given by terms RGLU and RINS in the differential equations. In the absence of dose events, both glucose and insulin are assumed to be at steady state levels. Exogenous doses of both glucose and insulin are also present in the first subject's data.

It is possible to intialize the steady state condition with a dose record having AMT=0, RATE=0, SS=1, but this is artificial. The initial condition that the system is at steady state is a feature of the model, not of the data. In this example, the $PK block contains the statement I_SS=1. This causes the system to be initialized appropriately. The statement I_SS=1 could also have been placed among the $MODEL statements. However the initialization is performed, the result is the same: non-zero concentrations GLU and INS are computed at TIME=0.

The SS data item need not be present in the data set for the I_SS computation. It is present in this example in order to demonstrate that SS may also be computed using a dose record (see the third and fourth subjects' data).

Note that SIGMA and OMEGA are fixed to 0 for purposes of illustration, so that the simulated values are exactly equal to those that arise from the model without the random variability due to ETA and EPS. NONMEM issues warning messages ("INITIAL ESTIMATE OF WITHIN INDIVIDUAL VARIANCE IS ZER0"). These may be ignored.

This example also illustrates the use of A(n) on the right in the $ERROR block to obtain the compartment amounts that are used for the computation of concentrations GLU and INS.

```
$PROB nicksiss6.ctl I_SS=1 in $PK ADVAN6
$DATA Nicksiss.dat
$INPUT ID TIME CMT AMT RATE SS DV
$SIM (200070927) ONLYSIM NSUB=1

;Silber HE, Jauslin PM, Frey N, Gieschke R, Simonsson US, Karlsson MO.
;An integrated model for glucose and insulin regulation in healthy
;volunteers and type 2 diabetic patients following intravenous glucose
;provocations.  J Clin Pharmacol. 2007 Sep;47(9):1159-71.

$THETA
40 ; POP_RGLU MMOL/H/70KG - glucose input rate
40 ; POP_VGLU L/70KG - volume of distribution for glucose
5 ; POP_CLGLU L/H/70KG - glucose clearance
1 ; POP_EMXGLU - EMAX for effect of glucose on insulin input
10 ; POP_C50GLU MMOL/L - EC50 for effect of glucose on Insulin input
5 ; POP_HILGLU - Hill parameter for effect of glucose on Insulin input

5000 ; POP_RINS PMOL/H/70KG - insulin input rate
5 ; POP_VINS L/70KG - volume of distribution for insulin
70 ; POP_CLINS L/H/70KG - insulin clearance
```

```
2 ; POP_EMXINS - EMAX for effect of insulin on glucose clearance
50 ; POP_C50INS PMOL/L - EC50 for effect of insulin on glucose clearance
2 ; POP_HILINS - Hill parameter for effect of insulin on glucose clearance

$OMEGA
0 FIX ;PPV_RGLU
0 FIX ;PPV_RINS

$SIGMA
0 FIXED ;0.1  ;G_EXP_RUV
0 FIXED ;1    ;G_ADD_RUV MMOL/L
0 FIXED ;0.1  ;I_EXP_RUV
0 FIXED ;1    ;I_ADD_RUV PMOL/L

$SUBR ADVAN6 TOL=3

$MODEL
   COMP (GLUCOSE)
   COMP (INSULIN)

$PK
   I_SS=1

   ; GLUCOSE
   RGLU=THETA(1)*EXP(ETA(1))
   VGLU=THETA(2)
   CLGLU=THETA(3)
   EMXGLU=THETA(4)
   C50GLU=THETA(5)
   HILGLU=THETA(6)

   ;INSULIN
   RINS=THETA(7)*EXP(ETA(2))
   VINS=THETA(8)
   CLINS=THETA(9)
   EMXINS=THETA(10)
   C50INS=THETA(11)
   HILINS=THETA(12)

   S1=VGLU
   S2=VINS

$DES
   DGLU=A(1)/VGLU
   DINS=A(2)/VINS
   DGLUH=DGLU**HILGLU
   DGEFF=EMXGLU*DGLUH/(C50GLU**HILGLU+DGLUH) ; effect of glucose on insulin
   DINSH=DINS**HILINS
   DIEFF=EMXINS*DINSH/(C50INS**HILINS+DINSH) ; effect of insulin on glucose
   DADT(1)=RGLU - CLGLU*(1+DIEFF)*DGLU       ; glucose
```

```
   DADT(2)=RINS*(1+DGEFF)- CLINS*DINS        ; insulin

$ERROR
   GLU=A(1)/VGLU   ; glucose concentration
   INS=A(2)/VINS   ; insulin concentration
   IF (CMT.EQ.1)THEN
     Y=GLU*(1+ERR(1))+ERR(2)
   ENDIF

   IF (CMT.EQ.2)THEN
     Y=INS*(1+ERR(3))+ERR(4)
   ENDIF

$TABLE ID TIME CMT AMT RATE SS GLU INS
 ONEHEADER NOPRINT FILE=nicksiss6.tab
```

The data (Nicksiss.dat) follows.

```
#ID,TIME,CMT,AMT,RATE,SS,DV
1    0   1   0    0    0 .
1    0   2   0    0    0 .
1   .1   1   25   0    0 .
1   1.0  2   10   0    0 .
1    3   1   0    0    0 .
1    3   2   0    0    0 .
1  100   1   .    .    . .
1  100   2   .    .    . .
2    0   1   0    0    0 .
2    0   2   0    0    0 .
2  100   1   .    .    . .
2  100   2   .    .    . .
3    0   1   0    0    1 .
3  100   1   .    .    . .
3  100   2   .    .    . .
4    0   1   0    0    1 .
4  100   1   .    .    . .
4  100   2   .    .    . .
```

(See **Initial Steady State**,**I_SS,ISSMOD**)
(See **i_ss**, **initial_condition**).

REFERENCES: none.

INTEROCCASION VARIABILITY EXAMPLE

This example is based on work described in Karlsson and Sheiner, JPB 21(6):735-750 (1994). The model accounts for random variability in a subject's parameters between study occasions.

It illustrates a data file in which EVID data items are present. In the fragment of data below, EVID has values 1 (dose event), 0 (observation event), and 3 (reset event).

The reset event is present at the start of the subject's second study occasion. It causes PREDPP to re-initialize the kinetic system as if for a new individual, and permits time to be reset to 0. Note that the subject has the same dosage regimen prior to both study occasions (a bolus dose of AMT 1000 every 12 hours leading to steady state). However, it would have been possible to specify a different dosing regimen prior to the second occasion.

In the abbreviated code, EXIT statements are used to constrain TVCL and TVV (typical value of clearance and volume) to positive values. Because the models for TVCL and TVV include both thetas and covariates, it is difficult to bound the thetas so as to insure that TVCL and TVV are non-negative for all subjects. The NOABORT option is present on the $ESTIMATION record. Hence NONMEM will perform PRED error recovery, and will avoid values of theta (subsequent to the initial values) that result in non-positive TVCL or TVV for any subject.

Explanation of user data items:

AGE	age
RACE	race (1 or 2)
HT	height
HCTZ	hydrochlorothiazide (0 or 1)
OCC	occasion (0 or 1)

```
;THIS IS A NMTRAN CONTROL STREAM For THE PRAZOSIN PK ANALYSIS
$PROB prazosin data
$INPUT ID AGE RACE HT HCTZ AMT TIME DV EVID SS II OCC
$DATA praz21
$SUBROUTINE ADVAN2 TRANS2
$PK
    OCC2=1-OCC
    R=0
    IF (RACE.EQ.2) R=1
    TMP=(HT-160)*THETA(1)
    IF (TMP.LE.0.) TMP=0.0
    TMP2=(AGE-60)*THETA(2)
    IF (AGE.LE.60.) TMP2=0.0
    TVCL=THETA(3)+TMP-TMP2
    TVCL=TVCL+THETA(4)*R+THETA(5)*HCTZ
    IF (TVCL.LE.0.) EXIT 1 100
    TVV=THETA(6)+HCTZ*THETA(7)
    IF (TVV.LE.0.) EXIT 1 200
    TVKA=THETA(8)
    CL=TVCL*EXP(ETA(3)*OCC+ETA(5)*OCC2+ETA(1))
```

```
    V =TVV *EXP(ETA(4)*OCC+ETA(6)*OCC2+ETA(2))
    KA=TVKA*EXP(ETA(8)*OCC+ETA(9)*OCC2+ETA(7))
    S2=V
$THETA (0,0.4) (0.,0.3) 20. 4 -4.
$THETA  80 -25
$THETA (0,1.)
$OMEGA BLOCK(2) 0.2 0.1 0.2
$OMEGA BLOCK(2) 0.2 0.1 0.2
$OMEGA BLOCK(2) SAME
$OMEGA 1.2
$OMEGA BLOCK(1) .5
$OMEGA BLOCK(1) SAME
$ERROR
    Y=F*(1+EPS(1))
$SIGMA  0.1
$EST NOABORT SIG=3 MAX=3000 PRINT=10 POSTHOC
$COVARIANCE
$TABLE ID OCC ETA1 ETA3 ETA5 ETA2 ETA4 ETA6 NOPRINT FILE=praz32.tab
```

Data for one subject follow.

```
    21    57     1   180     1  1000     0  0.62     1     1    12     0
    21    57     1   180     1   0.0     1 12.06     0     0     0     0
    21    57     1   180     1   0.0     3  6.81     0     0     0     0
    21    57     1   180     1   0.0     4  4.89     0     0     0     0
    21    57     1   180     1   0.0     5  4.04     0     0     0     0
    21    57     1   180     1   0.0     6  2.82     0     0     0     0
    21    57     1   180     1   0.0     7  2.72     0     0     0     0
    21    57     1   180     1   0.0     8  1.78     0     0     0     0
    21    57     1   180     1   0.0    10  1.07     0     0     0     0
    21    57     1   180     1   0.0    12  0.75     0     0     0     0
    21    57     1   180     1   0.0     0  2.83     3     0     0     0
    21    57     1   180     1  1000     0  2.83     1     1    12     1
    21    57     1   180     1   0.0     1  6.27     0     0     0     1
    21    57     1   180     1   0.0     2 12.72     0     0     0     1
    21    57     1   180     1   0.0     3  9.99     0     0     0     1
    21    57     1   180     1   0.0     4  6.90     0     0     0     1
    21    57     1   180     1   0.0     5  6.59     0     0     0     1
    21    57     1   180     1   0.0     6  4.07     0     0     0     1
    21    57     1   180     1   0.0     7  3.11     0     0     0     1
    21    57     1   180     1   0.0     8  2.79     0     0     0     1
    21    57     1   180     1   0.0    10  1.72     0     0     0     1
    21    57     1   180     1   0.0    12  1.17     0     0     0     1
```

(See **$estimation**, **abbreviated code**, **evid**).

REFERENCES: Guide VI, section V.B , VII.C.2
REFERENCES: Guide IV, section III.B.10 , IV.G
REFERENCES: Guide V, section 12.4.15

INTERPOLATION IN $DES

This is an example of code that can be used in $DES to interpolate a covariate such as WT between values that are recorded on the data records. This example is new to NONMEM 7.4, although the code can be used with previous versions.

If D_WT is used in the model for volume of distribution, then this would allow a time-varying volume of distribution to be integrated.

All the code in $PK to compute OLDTIME and OLDWT and SLOPE, and the code for D_WT in $DES, could be copied to user's control stream. Additional code (for integrating D_WT in $DES and the geometric solution in $ERROR) is for testing and would not be part of the user's control stream.

The code cannot be used when any WT values are missing. $PK sees only the previous and current record. If WT is missing from the current record, $PK has no way of knowing what WT will be on the next record.

The code in NONMEM 7.3 help file INFN_INTERPOLATION EXAMPLE 2 (infn2.exa) can be used to fill in missing WT values in the event records. The $INFN block makes two passes through the data set. After the first pass, it knows what the WT values are on all the event records, so that it can interpolate to fill in a missing WT.

The finedata utility (fine1.exa) can also be used to fill in the missing values, and performs interpolation if necessary.

(See **infn2.exa**).
(See **fine1.exa**).

Here is the control file:

```
$PROB  INTERPOLATE WT IN $DES
; this example shows how to interpolate WT in $DES.
; it is assumed that WT is recorded on every data record.
; As a test, the value of D_WT in $DES is integrated to
; obtain AUC of WT VS. T
; This is also calculated geometrically in $ERROR.

$INPUT      ID TIME WT DV
$DATA       desinterp.dat

$SUBROUTINES  ADVAN6  TOL=5
$MODEL
COMP=(AUC_WT DEFOBS)

$PK
; initialize OLDTIME and OLDWT
IF (NEWIND.LE.1) THEN
  OLDTIME=TIME
  OLDWT=WT
ENDIF

; calculate the slope for $DES
DELTA_TIME=TIME-OLDTIME
DELTA_WT=WT-OLDWT
```

```
IF (DELTA_TIME>0) THEN
   SLOPE=DELTA_WT/DELTA_TIME
ELSE
   SLOPE=0.
ENDIF

; save wt and time for next $PK record
OLDTIME=TIME
OLDWT=WT

$DES
   D_WT=OLDWT+SLOPE*(T-OLDTIME) ; D_WT is the value of WT at time T
   DADT(1)=D_WT                 ; compute AUC of D_WT as a test

$ERROR

   Y=F+ETA(1)+EPS(1)

; Compute geometric solution as a test.
; Does not use compartment amounts.
; Use only the values of WT and TIME on event records.
; Suppose WT vs T looks like this:
;
;
;         WT
;          |
;          |                w3   w4
;          |           w2            w5
;          |      w1
;          |
;           --------------------------------> TIME
;                  t1  t2   t3   t4  t5
;
; at t2, the contribution to the sum is
;   the rectangle  w1 x (t2-t1)
;   plus the triangular piece
;   (w2-w1)/(t2-t1) / 2
;
;                    w2
;                   /|
;                  / |
;                 /__|
;              w1    |
;               |    |
;               |    |
;            ------------
;                t1  t2
IF (NEWIND.LE.1) THEN
   PREV_WT=WT ; Initialize WT from previous data record
   SUM=0
```

```
ELSE
   SUM=SUM+PREV_WT*DELTA_TIME+DELTA_WT*DELTA_TIME/2
ENDIF
PREV_WT=WT ; save WT from previous data record

$OMEGA 1
$SIGMA 1

$TABLE  ID TIME WT PRED=AUC_WT SUM FILE=desinterp.tbl NOPRINT NOAPPEND
; The following two values should always be equal:
; PRED (which is the AUC of WT obtained by integrating WT)
; SUM (which is the geometric solution) computed in $ERROR
```

Here is the data file for the first subject. Note that WT sometimes is constant and sometimes decreases:

```
  1 0.    10  0
  1 1.    20  0
  1 2.    35  0
  1 2.    35  0
  1 4.    45  0
  1 5.    40  0
```

Here is the table file:

```
TABLE NO.  1
 ID           TIME         WT           AUC_WT       SUM
  1.0000E+00  0.0000E+00  1.0000E+01  0.0000E+00  0.0000E+00
  1.0000E+00  1.0000E+00  2.0000E+01  1.5000E+01  1.5000E+01
  1.0000E+00  2.0000E+00  3.5000E+01  4.2500E+01  4.2500E+01
  1.0000E+00  2.0000E+00  3.5000E+01  4.2500E+01  4.2500E+01
  1.0000E+00  4.0000E+00  4.5000E+01  1.2250E+02  1.2250E+02
  1.0000E+00  5.0000E+00  4.0000E+01  1.6500E+02  1.6500E+02
```

REFERENCES:

LOGISTIC REGRESSION EXAMPLE

Here is a typical NM-TRAN control stream for dichotomous response population data. Such data is referred to as "odd-type" data. Note that the eta is a population eta but there are no epsilons.

```
$PROB  DICHOTOMOUS RESPONSE
$DATA data
$INPUT ID DOSE DV
$PRED
LOGIT=THETA(1)+THETA(2)*DOSE**THETA(3)+ETA(1)
A=EXP(LOGIT)
P=A/(1+A)
IF (DV.EQ.1) THEN
   Y=P
ELSE
   Y=1-P
ENDIF
$THETA .1 (25,100) (1,3,5)
$OMEGA 1
$ESTIMATION  MET=COND LAPLACE LIKE MAX=500
$SCAT ETA(1) VS DOSE
```

Here is a typical NM-TRAN control stream when, one wants to first simulate the data.

```
$PROB  DICHOTOMOUS RESPONSE
$DATA data
$INPUT ID DOSE DV
$PRED
LOGIT=THETA(1)+THETA(2)*DOSE**THETA(3)+ETA(1)
A=EXP(LOGIT)
P=A/(1+A)
IF (ICALL.EQ.4) THEN
   CALL RANDOM (2,R)
   IF (R.LE.P) DV=1
   IF (R.GT.P) DV=0
ENDIF
IF (DV.EQ.1) Y=P
IF (DV.EQ.0) Y=1-P
$THETA .1 (25,100) (1,3,5)
$OMEGA 1
$SIMULATION (7755399) (45211 UNIFORM)
$ESTIMATION  MET=COND LAPLACE LIKE MAX=500
$SCAT ETA(1) VS DOSE
```

REFERENCES: None.

MICHAELIS MENTEN EXAMPLE

The kinetic system in this example is composed of depot, peripheral, and central compartments, and includes Michaelis-Menten Elimination from the central compartment. Either ADVAN6 or ADVAN8 could be used.

As with any ADVAN, one or more doses could be input to any of the compartments. Steady state doses are also possible.

```
$PROBLEM EXAMPLE OF M-M ELIMINATION WITH DEPOT AND PERIPH
$INPUT   ID TIME AMT DV
$DATA   datafile
$SUBROUTINES  ADVAN6  TRANS1 TOL=4
$MODEL COMP=(DEPOT,DEFDOS),COMP=(CENTRAL,DEFOBS) COMP=(PERIPH)

$PK
   VM  = THETA(1)*EXP(ETA(1))
   KM  = THETA(2)*EXP(ETA(2))
   S2  = THETA(3)*EXP(ETA(3))
   K12 = THETA(4)*EXP(ETA(4))
   K23 = THETA(5)*EXP(ETA(5))
   K32 = THETA(6)*EXP(ETA(6))

$ERROR
   Y = F + ERR(1)

$DES
    C2      =  A(2)/S2
    DADT(1) = -K12*A(1)
    DADT(2) =  K12*A(1) -K23*A(2) +K32*A(3) -C2*VM/(KM+C2)
    DADT(3) =           K23*A(2) -K32*A(3)

; requires $THETA, $OMEGA, $SIGMA etc.
```

If the units of KM is amount, rather than concentration, the $DES block need not compute C2. The change is as follows:

```
    DADT(2) =  K12*A(1) -K23*A(2) +K32*A(3) -A(2)*VM/(KM+A(2))
```

(ADVAN10 assumes the units of KM is amount.)

REFERENCES: Guide VI, section VII.C.6 , VII.C.10

MIXTURE MODEL EXAMPLE

In this example of a mixture model, it is assumed that some unknown fraction of the population has one set of typical values of CL and Vd, and that the remaining fraction has another set of typical values. Reserved variables of interest:

MIXNUM
: NONMEM sets MIXNUM to 1 or 2 according to whether PK is to compute parameter values for the 1st or 2nd subpopulations, respectively.

MIXEST
: For each individual, NONMEM computes an estimate of the number of the subpopulation of which each individual is a member, and stores this estimate in the integer variable MIXEST.

EST
: In the $PK block, MIXEST is stored in the PK-defined variable EST, which enables MIXEST to be displayed in tables and scatterplots.

```
$PROB    PHENOBARB POPULATION DATA    MIXTURE MODEL
$DATA  DATA2 (6F7.0)
$INPUT  ID TIME AMT WT APGR CP=DV
$SUBROUTINES ADVAN1 TRANS2
$PK
       CALLFL=1
       EST=MIXEST
       IF (MIXNUM.EQ.2) THEN
         CL=THETA(2)*THETA(1)*EXP(ETA(3))
         V=THETA(4)*THETA(3)*EXP(ETA(4))
       ELSE
         CL=THETA(1)*EXP(ETA(1))
         V=THETA(3)*EXP(ETA(2))
       ENDIF
       S1=V
$ERROR
       Y=F*EXP(EPS(1))
$MIX
       P(1)=THETA(5)
       P(2)=1.-THETA(5)
       NSPOP=2
$THETA  (0,.0047)  (0,1)  (0,.99)  (0,1)  (0,.5,1)
$OMEGA  BLOCK(2) .05  .01  .03
$OMEGA  BLOCK(2) SAME
$SIGMA  .02
$ESTM   MAXEVAL=500   PRINT=5
$TABLE ID  EST FIRSTONLY NOAPPEND
```

(See **displayed_values**, **$mix**, **mixnum mixest**).

REFERENCES: Guide VI, section III.L.2 , Figures 6, 23

MODEL TIME EXAMPLES

These are examples of the use of model times in PREDPP.

Enterohepatic Recycling

This fragment of abbreviated code may be used to model EHC. The transfer of drug from compt. 4 to 1 is controlled by `FLAG`, which is 1 between the times specified by MTIME(1) and MTIME(2), and is 0 otherwise.

```
$PK
MTIME(1)=THETA(8)
MTIME(2)=MTIME(1)+THETA(9)
 ....
$DES
FLAG=MPAST(1)-MPAST(2)
DADT(1)=-KA*A(1)+K41*A(4)*FLAG
DADT(4)=K1G*A(2)-K41*A(4)*FLAG
 ....
```

Flexible Input Rate

This fragment of abbreviated code may be used to model a flexible modeled infusion rate `R1`.

The rate is 400*EXP(ETA(1)) from time 0 to 1.5
The rate is 300*EXP(ETA(2)) from time 1.5 to 2.5
The rate is 200*EXP(ETA(3)) from time 2.5 till end of infusion.

In the three assignment statments for `R1`, exactly one of the right-side expressions is non-zero at each call to PK.

```
$PK
MTIME(1)=1.5
MTIME(2)=2.5
R1=   400*EXP(ETA(1))*(1-MPAST(1))
R1=R1+300*EXP(ETA(2))*(MPAST(1)-MPAST(2))
R1=R1+200*EXP(ETA(3))*MPAST(2)
```

Changing a Model Time

This fragment of abbreviated code shows how a model time parameter can be changed. Suppose there are events at times 0 and 10 but one wants to advance in increments of 1 with stops at times 1, 2, 3, ... , 9.

```
IF (TIME.EQ.0) TEMP=0
TEMP=TEMP+1
MTIME(1)=TEMP
MTDIFF=1
```

Changing Absorption Rate

This fragment of abbreviated code shows how the Absorption Rate KA can be changed.
KA is XKA1 from time 0 to THETA(5).
KA is XKA2 after time THETA(5).

```
$PK
MTIME(1)=THETA(5)  ; change point for KA
XKA1=THETA(3)*EXP(ETA(3))
XKA2=THETA(4)*EXP(ETA(4))
KA=XKA1*(1-MPAST(1))+XKA2*(MPAST(1))
```

(See **mtime**).

See also "Cirdadian example: Examples Using MTIME to Model Periodic Discontinuities in $DES".

REFERENCES: None.

MOVING AREA UNDER THE CURVE EXAMPLE

The following set of examples was developed in response to a question to the NONMEM Users about modeling a moving value of the area under the curve.

The original question (from Pavel Belo) asked:

"Efficacy is frequently considered a function of AUC. A disadvantage of this model of efficacy is that the effect is irreversable because AUC of concentration can only increase; it cannot decrease. In many cases, a more meaningful model is one where AUC is calculated from time t -a to t (kind of "moving average"), where t is time in the system of differential equations (variable T in NONMEM)."

DUPLICATE SYSTEM METHOD

Bob Bauer proposed a data set DELAYDATA and control file for computing a moving value of AUC. Bob wrote

"In the following simple absorption model example developed by me and Alison Boeckmann for illustration purposes, compartments 1, 2, and 3 are the "real time" depot, central, auc, and compartments 4,5,6 are the "delayed time" depot, central, auc. So, the base model (non-time delay) system (compartments 1,2,3) is replicated (compartments 4,5,6) for the time delay portion. In addition, the data set duplicates the dose information of compartment 1 into compartment 4, and setting ALAG4 to a non-zero value in the control stream file provides a lag time to any doses inputted into compartment 4 (so this would take care of multiple dose problems as well). This allows for assessment and availability of AUC(t) and AUCT(t-time-delay) at any time t."

DELAYDATA and aucdelay1.ctl

The data for one individual (DELAYDATA) and the control file (aucdelay1.ctl) from the email are provided in the examples directory of NONMEM 7.4 and higher. Note that aucdelay1.ctl and all the examples discussed below may be used with data from more than one individual and different numbers of event records per individual.

In aucdelay1.ctl, the model for time delay is

```
TDY=THETA(1)*EXP(ETA(1))
```

and the variable DAUC ("difference in area under the curve") gives the difference in the area under the curve

```
DAUC=A(3)-A(6) ; AUC(T)-AUC(T-TDY)
```

One significant modification to the control file aucdelay1.ctl was made. The $ERROR model was changed from `Y=F+EPS(1)` to

```
Y=F+DAUC+EPS(1)
```

This demonstrates that DAUC can modify Y (although the relationship would not usually be this simple).

SINGLE SYSTEM METHOD

In aucdelay1.ctl the value of DAUC is available continuously at every time value T during the integration, but in fact it is not used in the differential equations themselves. DAUC is used only in the $ERROR block. In this case, an alternate method is possible. Only the "real time" single system of differential equations is integrated, and the needed values of AUC are captured "on the fly" in the $PK block. This should result in faster run times because fewer differential equations need to be numerically integreated.

The model event time variable (MTIME) is used instead of absorption lag (ALAG) and is modelled with the same time delay variable TDY. For each event record, MTIME is used to interrupt the integration at the time of interest TIME-TDY. Variables called SAVEA are used to capture the value of A(3) at model event times for use in the $ERROR block.

The general algorithm is rather complicated, so a simple algorithm will be described first.

SINGLE SYSTEM METHOD: SIMPLE MODEL

A simple model is possible when TDY is always smaller than the time between any two event records.

aucdelay1S.ctl

This is identical to aucdelay1.ctl except that THETA(1) has initial estimate .5 and is constrained to be between 0 and 1, and TDY is constrained to be less than 1, which is the change in time between any two event records.

aucdelay2S.ctl

This implements the simple model. Only the original (single) set of compartments 1-3 are defined. aucdelay2S.ctl uses the same data file DELAYDATA, but ignores records with CMT=4 (doses into the delayed dose compartment). For each event record, it is necessary to have available NEXTT, the TIME of the next event record. This data item is not recorded in DELAYDATA. Instead, NEXTT is listed in the $INPUT record as an extra data item beyond the data items (columns) listed in the data file, and its entries and are created ("transgenerated") during each run by code in the $INFN block. The control stream has the same constraints on THETA(1) and TDY as aucdelay1S.ctl.

Only one model event time variable, MTIME(1) is needed. Model event time MTIME(1) is modelled as NEXTT-TDY and changes with every event record. SAVEA is a variable that saves the value of the AUC compartment A(3) when $PK is evaluated at the model event time. At such times, MTIME(1) is reset for the next record. The important lines of code are relatively simple, and use TSTATE ("State Time"; the time at which the state-vector A was last computed.)

```
$PK
TDY=THETA(1)*EXP(ETA(1))
MTIME(1)=NEXTT-TDY
IF (TSTATE==TIME-TDY) SAVEA=A(3)
 ...
$ERROR
DAUC=A(3)-SAVEA
 ...
```

The two control files generate the same predictions.

SINGLE SYSTEM METHOD: GENERAL MODEL

The model is more complicated if TDY may be greater than the time between event records, because each value of compartment A(3) must be saved until needed. For the ith. event record, with TIME(i), the value of SAVEAi at TIME(i)-TDY is needed, and is obtained by using a value of MTIME(i) equal to TIME(i)-TDY.

aucdelay2.ctl and aucdelay3.ctl

The general model is implemented in aucdelay2.ctl and aucdelay3.ctl, These versions integrate a single system of differential equations. File aucdelay3.ctl is a short-hand version of aucdelay2.ctl. After processing by the nmtemplate and doexpand utilites, it gives a control stream that is essentially identical to aucdelay2.ctl. It may be easier to look at

the more compact code in aucdelay3.ctl.

The maximum number of records in any individual record is set using nmtemplate. DELAYDATA contains 16 records. Commands to run aucdelay3.ctl are:

```
nmtemplate aucdelay3.ctl temp1 maxrecs=16
doexpand  temp1  temp2
nmfe74 temp2 aucdelay3.res
(The most up-to-date doexpand and nmtemplate utilities are provided in
https://nonmem.iconplc.com/utilities/)
```

Like aucdelay2S.ctl, aucdelay3.ctl uses DELAYDATA, and ignores records with CMT=4. An $INFN block is used to save all values of TIME in an array TIMES(i). There is no need for an explicit NEXTT variable because all values of TIME are available in this array. Note that the TIMES array contains all TIME values in the data set, which typically contains more than one subject's data. In the abbreviated code, reserved variable IRECIDX gives the position in TIMES of the start of the current individual record. (LIREC gives the number of records in the current individual record and is needed if different subjects have different numbers of data records so that maxrecs is too large for some subjects.)

At the start of an individual record, a value of MTIME(i) is computed for every data record as TIMES(KREC)-TDY. (KREC uses reserved variable IRECIDX, the first record in the data set, and hence in TIMES, for the current individual record.) Some values of MTIME(i) may be <=0; such values are ignored by PREDPP.

When PK is called at a model event time MTIME(i) the value of A(3) is stored as SAVEAi. There may be several calls to PK for a given value of TSTATE and MTIME(i) but SAVEAi must be assigned on only the first of these calls because only at this call will A(3) have derivaties with respect to ETA (if MTIME depends on ETA). This is why MTIME(i)=0 is necessary after the value of A(3) (and its ETA derivatives, if any) has been saved.

REMARKS

1. RECURSIVE CODE

Random variables are saved for future use by using with recursive code, such as in aucdelay2.ctl:

```
IF (NEWIND.LE.1.AND.1<=LIREC) THEN
MTIME(1)=TIMES(KREC)-TDY  ; SET NEW VALUE OF MTIME (TDY MAY DEPEND ON ETA)
SAVEA1=0                  ; ERASE OLD VALUE FROM PREVIOUS PASS
ELSE
MTIME(1)=MTIME(1)
SAVEA1=SAVEA1
ENDIF
```

It is instructive to note that there is one warning that may be ignored:

```
 (WARNING  3) THERE MAY BE AN ERROR IN THE ABBREVIATED CODE. THE FOL-
LOWING
 ONE OR MORE RANDOM VARIABLES ARE DEFINED WITH "IF" STATEMENTS THAT
DO NOT
 PROVIDE DEFINITIONS FOR BOTH THE "THEN" AND "ELSE" CASES. IF ALL
 CONDITIONS FAIL, THE VALUES OF THESE VARIABLES WILL BE ZERO.

   SA
```

The code in $ERROR is such that SA is always given a value and SA never defaults to zero. Recursive code for SA could be used to avoid this warning if desired:

```
IF ([I]<=LIREC.AND.TIME==TIMES(KREC)) THEN
SA=SAVEA[I]
ELSE
SA=SA
ENDIF
```

2. SAVEA variables

The variables SAVEAi save the value of compartment 3 (AUC) at TIME(i)-TDY. They must be assigned individually because there is no true array feature in NMTRAN for random variables.

3. SIMULATION

The control streams can be used to simulate values of DV. There will be a difference between simulated values from aucdelay1.ctl vs. aucdelay2.ctl (2 and 3 are the same). The extra record in aucdelay1.ctl's FDATA causes an extra call to SIMEPS (even though it is a dose record), so the simulated etas and eps are not identical. To have identical simulated values, then do not use

```
$DATA DELAYDATA IGNORE=@ IGNORE=(CMT==4)
```

Instead, use a version of DELAYDATA in which the second record has EVID=2 ("other-type event record"), so that it is physically present but does nothing. This adds another record to the data set. Instead of

```
nmtemplate aucdelay3.ctl temp1 maxrecs=16
```

the value maxrecs=17 should be used.

All files may be found in the examples directory:

DELAYDATA
aucdelay1S.ctl
aucdelay2S.ctl
aucdelay1.ctl
aucdelay2.ctl
aucdelay3.ctl

REFERENCES: Guide Introduction_7

MULTIPLE DOSE EXAMPLE

This example is based on work described in Fattinger K. et al., Netilmicin in the neonate - population pharmacokinetic analysis and dosing recommendations, Clinical Pharmacology and Therapeutics, 1991, Jul, V50; N1:55-65. It illustrates a data file with multiple doses of different kinds.

A FORTRAN format statement is needed to describe the data file because each data record spans two FORTRAN records and also because blanks are sometimes used for null data items.

Explanation of data items:

TIME	hours
KG	bodyweight(kg)
LENG	length(cm)
SEX	0 for female and 1 for male
GAGE	gestational age at birth
AGE	actual age (in days)
AGE1	age at start of therapy
IMIV	im or iv dose (1.0 is iv and 0.0 is im)
CREA	creatinine value
TAD	time after last dose (h)

All doses are given as zero order infusions. The RATE data item is -2, indicating that duration D1 is modeled in the $PK block. There are two types of doses: IV (intravenous) and IM (intramuscular). The duration of IV doses is 2 minutes. The duration of IM doses is unknown and is given by theta(7).

The Estimation Step produces a Model Specification File, which may be used in in a later run; (See **$msfi**, **model specification file**).

```
$PROBLEM  NETILMICIN
$INPUT    ID MDV TIME AMT RATE CP=DV KG LENG SEX
          GAGE AGE AGE1 IMIV CREA TAD
$DATA     data (6F7.0/7X,9F7.0)
$SUBROUTINE ADVAN3,TRANS4
$PK
  TVCL1=THETA(1)*KG
  GAGEN=GAGE/39.0
  PAN=(AGE+1.0)/7.0
  IF(GAGE.LT.39.0) THEN
  TVCL=TVCL1*(GAGEN**THETA(3))*(PAN**THETA(2))
  ELSE
  TVCL=TVCL1*(PAN**THETA(2))
  ENDIF
  CL=TVCL*EXP(ETA(1))
  TVV1=THETA(4)*KG
  V1=TVV1*EXP(ETA(2))
  Q=THETA(5)
  V2=THETA(6)*KG
  S1=V1
```

```
  IF(IMIV.GE.1.0) THEN
  D1=.0333333   ; 2./60.
  ELSE
  D1=THETA(7)
  ENDIF
$ERROR
 Y=F*(1+EPS(1))
$THETA (0,0.07) (0,0.1) (0,1.1) (0,0.4) (0,0.07) (0,0.8) (0,.033)
$OMEGA 0.02 0.02
$SIGMA 0.02
$ESTIMATION PRINT=5 MSFO=msfo1
$COVARIANCE
$TABLE MDV ID KG LENG SEX GAGE AGE TAD BY MDV ID NOPRINT FILE=NTABLE.DAT
$SCATTER PRED VS CP  UNIT
```

Data for one subject follow. This subject received both IM and IV doses.

```
    6.0    1.0    0.0   13.0   -2.0
    6.0  2.590   47.0    0.0   36.0    0.0    0.0    1.0   56.0           0.0
    6.0    1.0   24.0   10.4   -2.0
    6.0  2.590   47.0    0.0   36.0    1.0    0.0    1.0   56.0           0.0
    6.0    0.0   26.0    0.0    0.0   6.95
    6.0  2.590   47.0    0.0   36.0    1.0    0.0    1.0   56.0    2.0    0.0
    6.0    0.0   32.0    0.0    0.0   3.45
    6.0  2.590   47.0    0.0   36.0    1.0    0.0    1.0   56.0    8.0    0.0
    6.0    0.0   40.0    0.0    0.0    1.8
    6.0  2.590   47.0    0.0   36.0    2.0    0.0    1.0   56.0   16.0    0.0
    6.0    1.0   48.0   10.4   -2.0
    6.0  2.590   47.0    0.0   36.0    2.0    0.0    0.0   56.0           0.0
    6.0    1.0   72.0   10.4   -2.0
    6.0  2.590   47.0    0.0   36.0    3.0    0.0    0.0   56.0           0.0
    6.0    1.0   96.0   10.4   -2.0
    6.0  2.590   47.0    0.0   36.0    4.0    0.0    0.0   56.0           0.0
    6.0    1.0  120.0   10.4   -2.0
    6.0  2.590   47.0    0.0   36.0    5.0    0.0    0.0   56.0           0.0
    6.0    0.0  122.0    0.0    0.0    5.9
    6.0  2.590   47.0    0.0   36.0    5.0    0.0    0.0   56.0    2.0    0.0
    6.0    0.0  128.0    0.0    0.0    2.1
    6.0  2.590   47.0    0.0   36.0    5.0    0.0    0.0   56.0    8.0    0.0
    6.0    0.0  136.0    0.0    0.0   0.75
    6.0  2.590   47.0    0.0   36.0    6.0    0.0    0.0   56.0   16.0    0.0
```

(See **duration**, **$data**).

REFERENCES: Guide VI, section V.D
REFERENCES: Guide V, section 6.8 , 6.10

NONPARAMETRIC EXAMPLE

DISCUSSION:

Three examples are given of the NONPARAMETRIC step in NONMEM. In examples 1 and 2, it is assumed that there only two etas in the problem. With more than two etas, insert the appropriate additional code.

Example 1

This example illustrates how the cumulative density can be plotted versus the conditional etas.

The following code is inserted in the $PRED or $PK block.

```
DN1=CDEN_(1)
DN2=CDEN_(2)
```

The following control records are used. Note that with the $NONPARAMETRIC record, the MARGINALS option is the default and need not be coded explicitly.

```
$ESTIMATION  POSTHOC  PRINT=1
$NONPARAMETRIC MARGINALS
$SCAT DN1 VS ETA1
$SCAT DN2 VS ETA2
```

Example 2

This example illustrates how both the conditional etas and the conditional nonparametric etas (CNPE) can be obtained, and how they can be compared.
(See **comact**).

The following code is inserted in the $PRED or $PK block:

```
;CONDITIONAL PARAMETRIC ETAS
IF (COMACT.EQ.2) ET1A=ETA(1)
IF (COMACT.EQ.2) ET2A=ETA(2)
;CONDITIONAL NONPARAMETRIC ETAS (CNPE)
IF (COMACT.EQ.3) ET1=ETA(1)
IF (COMACT.EQ.3) ET2=ETA(2)
```

The following control statements may be used. The first two are required; the $SCATTER and $TABLE statements are examples of what can be used.

```
$ESTIMATION POSTHOC
$NONP ETAS
; Examine distribution of nonparametric etas
$SCAT ET1 VS ID
$SCAT ET2 VS ID
; Compare parametric and nonparametric etas
$SCAT ET1A VS ET1
$SCAT ET2A VS ET2
$TABLE ID ET1A ET2A ET1 ET2 NOPRINT
```

Example 3

This example shows the specific statements needed to simulate from a nonparametric distribution using a Model Specification file generated from a previous analysis using $NONPARAMETRIC.

```
; Model specification file from a previous analysis using $NONPARAMETRIC
$MSFI nonpfit.msf

; Second seed will be used for simulation using the nonparametric density
; Must include TRUE=FINAL, otherwise initial instead of final parameter
; estimates will be used.

$SIMULATION (3432192) (6348371 NONPARAMETRIC) TRUE=FINAL
```

REFERENCES: None.

NWPRI EXAMPLES

These examples illustrate the use of the NONMEM utility routine NWPRI by the NM-TRAN $PRIOR record, and also by an equivalent user-supplied routine PRIOR. For a discussion of PRIOR: (See **prior**). For a discussion of the NWPRI routine and in particular, a discussion of the $THETA, $OMEGA, $SIGMA records: (See **nwpri**).

Example 1

Use of PRIOR and NWPRI and a control stream for single-subject data:

This example obtains parameter estimates from single subject data, using a two-compartment PK model, and it incorporates a frequentist prior of multivariate normal form for all of the THETA vector.

```
$PROB SINGLE SUBJECT DATA WITH PRIOR ON THETA
$INPUT ID DOSE TIME DV WT
$DATA data1
$PRIOR NWPRI NTHETA=3,NETA=1,NTHP=3,NPEXP=1
$PRED
;
;      THETA(1)=MEAN ABSORPTION RATE CONSTANT
;      THETA(2)=MEAN ELIM. RATE CONSTANT (1/HR)
;      THETA(3)=CLEARANCE (LITERS/HR)
;      DOSE=WEIGHT-ADJUSTED DOSE (MG/KG)
;
       IF (NEWIND.EQ.0) AMT=DOSE*WT
       T1=THETA(1)
       T2=THETA(2)
       T3=THETA(3)
       A=EXP(-T2*TIME)
       B=EXP(-T1*TIME)
       C=T1-T2
       D=A-B
       E=T3*C
       Y=AMT*T1*T2/E*D+ETA(1)

$THETA  (.4,1.7,7) (.025,.102,.4) (.3,3,30)
;mode for prior with THETA:
$THETA 2.77 FIX .0781 FIX 2.63 FIX
$OMEGA .388
;var-cov for prior with THETA:
$OMEGA BLOCK (3) 5.55 .00524 .00024 -.128 .00911 .515 FIX

$EST
```

Instead of a `$PRIOR` record, the following may be used:

```
$SUBROUTINE PRIOR=prior
```

The prior routine is as follows:

```
       SUBROUTINE PRIOR (ICALL,CNT)
       USE SIZES,     ONLY: DPSIZE,ISIZE
       REAL(KIND=DPSIZE) :: CNT
       REAL(KIND=DPSIZE) :: PLEV
       INTEGER(KIND=ISIZE) :: ICALL
       NTHETA=3
       NETA=1
       NTHP=3
       NPEXP=1
```

```
      PLEV=0.
      ITYP=0
      NSAM=0
      ISS=0
      NEPS=0
      NETP=0
      NEPP=0
      CALL NWPRI(NTHETA,NETA,NEPS,NTHP,NETP,NEPP,NPEXP,ITYP,PLEV, &
                 NSAM,ISS,CNT)
      RETURN
      END
```

Example 2

First example of use of PRIOR and NWPRI and a control stream for population data:

This example obtains parameter estimates from population data, using a two-compartment PK model, and it incorporates a frequentist prior of multivariate normal/inverse Wishart form for all of the THETA vector and OMEGA matrix.

```
$PROB POPULATION DATA WITH PRIOR ON THETA AND OMEGA
$INPUT ID DOSE TIME DV WT
$DATA data2
$PRIOR NWPRI NTHETA=3 NETA=3 NTHP=3 NETP=3 NPEXP=1
$PRED
;
;     THETA(1)=MEAN ABSORPTION RATE CONSTANT - MEAN ELIM. RATE CONSTANT (1/HR)
;     THETA(2)=MEAN ELIM. RATE CONSTANT (1/HR)
;     THETA(3)=SLOPE OF CLEARANCE VS WEIGHT RELATIONSHIP (LITERS/HR/KG)
;     DOSE=WEIGHT-ADJUSTED DOSE (MG/KG)
;
      IF (NEWIND.NE.2) THEN
         AMT=DOSE*WT
         W=WT
      ENDIF
      T0=THETA(1)*EXP(ETA(1))
      T2=THETA(2)*EXP(ETA(2))
      T1=T2+T0
      T3=THETA(3)*W*EXP(ETA(3))
      A=EXP(-T2*TIME)
      B=EXP(-T1*TIME)
      C=T1-T2
      D=A-B
      E=T3*C
      Y=AMT*T1*T2/E*D+EPS(1)
$THETA  (0,4,5) (0,.09,.5) (.004,.01,.9)
;mode for prior on THETA:
$THETA 3 FIX .08 FIX .04 FIX
;df for prior on OMEGA:
$THETA 12 FIX
$OMEGA BLOCK (3) .7 .04 .05 .02 .06 .08
;var-cov for prior on THETA:
$OMEGA BLOCK (3) .494 .00207 .0000847 .000692 .0000471 .0000292 FIX
;mode for prior on OMEGA:
$OMEGA BLOCK (3) .7 .04 .05 .02 .06 .08 FIX
$SIGMA  .4

$EST
```

Instead of a `$PRIOR` record, the following may be used:

```
$SUBROUTINE PRIOR=prior
```

The prior routine is as follows:

```
      SUBROUTINE PRIOR (ICALL,CNT)
      USE SIZES,     ONLY: DPSIZE,ISIZE
      REAL(KIND=DPSIZE) :: CNT
      REAL(KIND=DPSIZE) :: PLEV
      INTEGER(KIND=ISIZE) :: ICALL
      NTHETA=3
      NETA=3
      NTHP=3
      NETP=3
      NPEXP=1
      PLEV=0.
      ITYP=0
      NSAM=0
      ISS=0
      NEPS=0
      NEPP=0
      CALL NWPRI(NTHETA,NETA,NEPS,NTHP,NETP,NEPP,NPEXP,ITYP,PLEV, &
                 NSAM,ISS,CNT)
      RETURN
      END
```

Example 3

Second example of use of PRIOR and NWPRI and a control stream for population data:

This example simulates the THETA and OMEGA parameters of the same two-compartment population PK model, as well as the population data from this model. During problem finalization, the entire post-simulation theta vector and omega array in common NMPR16 are written to external files.

```
$PROB POPULATION DATA WITH PRIOR ON THETA AND OMEGA
$INPUT ID DOSE TIME DV WT
$DATA data2
$PRIOR NWPRI
       NTHETA=3 NETA=3 NTHP=3 NETP=3 NPEXP=1 ITYP=0 PLEV=.9999
$PRED
;
;     THETA(1)=MEAN ABSORPTION RATE CONSTANT - MEAN ELIM. RATE CONSTANT (1/HR)
;     THETA(2)=MEAN ELIM. RATE CONSTANT (1/HR)
;     THETA(3)=SLOPE OF CLEARANCE VS WEIGHT RELATIONSHIP (LITERS/HR/KG)
;     DOSE=WEIGHT-ADJUSTED DOSE (MG/KG)
;
      IF (NEWIND.NE.2) THEN
         AMT=DOSE*WT
         W=WT
      ENDIF
      T0=THETA(1)*EXP(ETA(1))
      T2=THETA(2)*EXP(ETA(2))
      T1=T2+T0
      T3=THETA(3)*W*EXP(ETA(3))
      A=EXP(-T2*TIME)
      B=EXP(-T1*TIME)
      C=T1-T2
      D=A-B
      E=T3*C
      Y=AMT*T1*T2/E*D+EPS(1)
      IF (ICALL.EQ.3) THEN
        WRITE (97,*) THSIMP,THSIMPR
```

```
        WRITE (98,*) OMSIMP(BLOCK)
      ENDIF

$THETA  (0,4,5) (0,.09,.5) (.004,.01,.9)
;mode for prior on THETA:
$THETA 3 FIX .08 FIX .04 FIX
;df for prior on OMEGA:
$THETA 12 FIX
$OMEGA BLOCK (3) .7 .04 .05 .02 .06 .08
;var-cov for prior on THETA:
$OMEGA BLOCK (3) .494 .00207 .0000847 .000692 .0000471 .0000292 FIX
;mode for prior on OMEGA:
$OMEGA BLOCK (3) .7 .04 .05 .02 .06 .08 FIX
$SIGMA  .4

$SIM (547676) ONLY
```

Suppose a $TABLE record is added to the above control stream. To obtain values of DV computed with the simlulated values of eta, include the following code:

```
IF (ICALL.EQ.4) THEN
T0=THSIMP(1)*EXP(ETA(1))
T2=THSIMP(2)*EXP(ETA(2))
T3=THSIMP(3)*W*EXP(ETA(3))
ELSE
T0=THETA(1)*EXP(ETA(1))
T2=THETA(2)*EXP(ETA(2))
T3=THETA(3)*W*EXP(ETA(3))
ENDIF
```

Instead of a `$PRIOR` record, the following may be used:

```
$SUBROUTINE PRIOR=prior
```

The prior routine is as follows:

```
SUBROUTINE PRIOR (ICALL,CNT)
USE SIZES,     ONLY: DPSIZE,ISIZE
REAL(KIND=DPSIZE) :: CNT
REAL(KIND=DPSIZE) :: PLEV
INTEGER(KIND=ISIZE) :: ICALL
PLEV=.9999
NTHETA=3
NETA=3
NTHP=3
NETP=3
NPEXP=1
NEPS=0
NEPP=0
ITYP=0
NSAM=0
ISS=0
CALL NWPRI(NTHETA,NETA,NEPS,NTHP,NETP,NEPP,NPEXP,ITYP,PLEV, &
           NSAM,ISS,CNT)
RETURN
END
```

REFERENCES: none.

OBJECTIVE FUNCTION VALUE INDIVIDUAL EXAMPLE

Note: With NONMEM 7, the additional output file root.phi contains the same information.
(See **additional_output_file**).

This example illustrates how individual contributions to the objective function may be obtained.
(See **Objective Function Value Individual**).

This example is a modification of CONTROL4 from the NONMEM distribution media. It is for NONMEM VI 2.0 and later.

```
$PROB  THEOPHYLLINE   POPULATION DATA
$INPUT       ID DOSE TIME CP=DV WT
$DATA        THEO
$PRED
;THETA(1)=MEAN ABSORPTION RATE CONSTANT (1/HR)
;THETA(2)=MEAN ELIMINATION RATE CONSTANT (1/HR)
;THETA(3)=SLOPE OF CLEARANCE VS WEIGHT RELATIONSHIP (LITERS/HR/KG)
;DOSE=WT-ADJUSTED DOSE (MG/KG)
;DS=NON-WT-ADJUSTED DOSE (MG)
   IF (DOSE.NE.0) THEN
      DS=DOSE*WT
      W=WT
   ENDIF
   KA=THETA(1)+ETA(1)
   KE=THETA(2)+ETA(2)
   CL=THETA(3)*W+ETA(3)
   D=EXP(-KE*TIME)-EXP(-KA*TIME)
   E=CL*(KA-KE)
   F=DS*KE*KA/E*D
   Y=F+EPS(1)
   IF (ICALL.EQ.3) THEN
   WRITE (70,*) IIDX,CNTID
   ENDIF
$THETA  (.1,3,5) (.008,.08,.5) (.004,.04,.9)
$OMEGA BLOCK(3)  6 .005 .0002 .3 .006 .4
$SIGMA  .4
$EST     MAXEVAL=450  PRINT=5
```

Pairs of values are written, one pair per line, one pair for each individual record. Each pair contains the ID data item and the contribution the objective function.

```
 1.000000000000000E+00  1.328986294492194E+01
 2.000000000000000E+00  2.071754196741930E+01
 ...
```

They may also be displayed in a table, using

```
$ABBR COMRES=2
```

and code such as the following in the $ERROR or $PK block:

```
IF (COMACT.EQ.1) THEN
  COM(1)=IIDX(NIREC)
```

```
  COM(2)=CNTID(NIREC)
ENDIF
```

The following, for example, will produced a separate table for the values:

```
$TABLE IID=COM(1) CNT=COM(2) FILE=comvals NOAPPEND NOPRINT FIRSTONLY
```

Note: With earlier versions than NONMEM 7.3, verbatim code is needed:

```
"  COM(1)=IIDX(NIREC)
"  COM(2)=CNTID(NIREC)
```

REFERENCES: None.

PHENOBARB EXAMPLE

This example illustrates simple kinetics.

```
$PROBLEM PHENOBARB  SIMPLE MODEL
$INPUT   ID TIME AMT WGT APGR DV
$DATA    PHENO
$SUBR   ADVAN1 TRANS2
$PK
      TVCL=THETA(1)
      TVV=THETA(2)
      CL=TVCL*EXP(ETA(1))
      V=TVV*EXP(ETA(2))
      S1=V
$THETA  (0,.0105)  (0,1.05)
$OMEGA  .25   .25
$ERROR
      W=F
      Y=F+W*EPS(1)

      IPRED=F         ;  individual-specific prediction
      IRES=DV-IPRED   ;  individual-specific residual
      IWRES=IRES/W    ;  individual-specific weighted residual
$SIGMA  .04
$ESTIMATION   SIGDIGITS=4 POSTHOC
$COVARIANCE
$TABLE      ID WGT APGR TIME AMT CL V ETA1 ETA2 IPRED IWRES
            NOPRINT ONEHEADER FILE=TABLE1
$SCATTER    DV VS PRED UNIT
$SCATTER    (RES,WRES) VS PRED
$SCATTER    (CL,V) VS (WGT,APGR)
$SCATTER    IWRES VS IPRED BY MDV ORDZERO
```

Data for the first individual follow.

```
      1    0.    25.0    1.4      7      .
      1    2.0      .    1.4      7   17.3
      1   12.5    3.5    1.4      7      .
      1   24.5    3.5    1.4      7      .
      1   37.0    3.5    1.4      7      .
      1   48.0    3.5    1.4      7      .
      1   60.5    3.5    1.4      7      .
      1   72.5    3.5    1.4      7      .
      1   85.3    3.5    1.4      7      .
      1   96.5    3.5    1.4      7      .
      1  108.5    3.5    1.4      7      .
      1  112.5      .    1.4      7   31.0
```

REFERENCES: Guide III, section VI.2
REFERENCES: Guide V, section 2.3

PK PD SEQUENTIAL 1 EXAMPLE

This is an example for analysis of direct PD using previously estimated individuals' PK parameters. (See **PK_PD_sequential_2** example).

```
$PROBLEM   EMAX MODEL APPLIED TO EFFECT SITE CONCENTRATION
; NOTE: INDIVIDUAL K, V ESTIMATED FROM PREVIOUS PK FIT
;       AND RECORDED IN DATA
;
; THE DATA FILE CONTAINS *ONLY* EFFECT OBSERVATIONS,
; CMT = 2  NEEDED TO TELL PREDPP TO SET F = CE
$DATA   data
$INPUT  ID TIME DV AMT=DOSE CMT  KK V1
;                                |---|
;                              KK AND V1 ARE ESTIMATES OF K AND V
$SUBROUTINES ADVAN3
$PK
   K  = KK
   K12= .001*K                      ; TRIVIAL LOSS TO EFFECT COMPT.
   K21= THETA(1)*EXP(ETA(1))
   EMAX=THETA(2)*EXP(ETA(2))
   C50= THETA(3)*EXP(ETA(3))
   S2=  V1*K12/K21                  ; SO THAT CESS = CPSS
$ERROR
   Y  =EMAX*F/(C50+F)*EXP(ERR(1))
; ADD APPROPRIATE $THETA, $OMEGA, $SIGMA, $ESTIMATION, ETC.
```

If, moreover, an absorption lag (ALAG1) parameter is modeled in $PK with parameters estimated from the PD data only, then this introduces a lag in the dose - Ce relationship. Since this lag was not present in the dose - Cp relationship, it may be interpreted as a lag in the Cp - Ce relationship.

REFERENCES: Guide IV, section V.C.6
REFERENCES: Guide VI, section IV.B.2 , VII.C.3

PK PD SEQUENTIAL 2 EXAMPLE

This is an example for analysis of direct PD using previously estimated population PK parameters. (See **PK_PD_sequential_1** example).

```
$PROBLEM   EMAX MODEL APPLIED TO EFFECT SITE CONCENTRATION
; THE DATA FILE CONTAINS *ONLY* EFFECT OBSERVATIONS,
; POP PK PARAMS ESTIMATED IN PREVIOUS RUN & FIXED HERE.
; CMT = 2  NEEDED TO TELL PREDPP TO SET F = CE
;
$DATA   data
$INPUT  ID TIME DV AMT=DOSE CMT
$SUBROUTINES ADVAN3
$PK
   K  = THETA(1)*EXP(ETA(1))
   K12= .001*K                        ; TRIVIAL LOSS TO EFFECT COMPT.
   K21= THETA(2)*EXP(ETA(2))
   V1=  THETA(3)*EXP(ETA(3))
   EMAX=THETA(4)*EXP(ETA(4))
   C50= THETA(5)*EXP(ETA(5))
   S2=  V1*K12/K21                  ;  SO THAT CESS = CPSS
$ERROR
   Y=   EMAX*F/(C50+F)*EXP(ERR(1))
$THETA
  .124 FIXED                        ; K   - FIXED TO PREV EST
  .1                                ; KEO - ESTIMATED
  28.6 FIXED                        ; V1  - FIXED TO PREV EST
 100                                ; EMAX- ESTIMATED
  50                                ; C50 - ESTIMATED
$OMEGA
  .113 FIXED                        ; OMEGA(K) - FIXED TO PREV EST
  .04
  .627 FIXED                        ; OMEGA(V1)- FIXED TO PREV EST
  .04
  .04
 $SIGMA .16
; ADD $ESTIMATION, ETC, AS DESIRED.
```

If, moreover, an absorption lag (ALAG1) parameter is modeled in $PK with parameters estimated from the PD data only, then this introduces a lag in the dose - Ce relationship. Since this lag was not present in the dose - Cp relationship, it may be interpreted as a lag in the Cp - Ce relationship.

REFERENCES: Guide IV, section V.C.6
REFERENCES: Guide VI, section IV.B.2 , VII.C.3

PK PD SIMULTANEOUS 1 EXAMPLE

This is an example for simultaneous analysis of PK and direct PD. Kinetics are one compartment. (See **PK_PD_simultaneous_2**, **PK_PD_simultaneous_3**).

```
$PROBLEM   SIMULT PK/PD - ADVAN3
; THE DATA FILE CONTAINS BOTH CP AND EFFECT OBSERVATIONS.
; WHEN DV IS A CP OBSERVATION, CMT = 1 (OR 0)
; WHEN DV IS AN EFFECT OBSERVATION, CMT =2
$DATA   data
$INPUT  ID TIME DV AMT=DOSE CMT
$SUBROUTINES ADVAN3
$PK
   K  = THETA(1)*EXP(ETA(1))
   K12= .001*K                    ; TRIVIAL LOSS TO EFFECT CMPT
   K21= THETA(2)*EXP(ETA(2))      ; KEO
   S1=  THETA(3)*EXP(ETA(3))      ; V1 FOR DRUG
   EMAX=THETA(4)*EXP(ETA(4))
   C50= THETA(5)*EXP(ETA(5))
   S2=  S1*K12/K21                ; SO THAT CESS = CPSS
$ERROR
   IF (CMT.EQ.2) THEN
   Y = EMAX*F/(C50+F)*EXP(ERR(2))   ; DV = EFFECT IF CMT=2
   ELSE
   Y = F*EXP(ERR(1))                ; DV = CP IF CMT=1 (OR 0)
   ENDIF
; ADD APPROPRIATE $THETA, $OMEGA, $SIGMA, $ESTIMATION, ETC.
```

REFERENCES: Guide IV, section V.C.6
REFERENCES: Guide VI, section IV.B.2 , VII.C.3

PK PD SIMULTANEOUS 2 EXAMPLE

This is an example for simultaneous analysis of PK and PD. Kinetics are one-compartment. (See **PK_PD_simultaneous_1**, **PK_PD_simultaneous_3**).

```
$PROBLEM   SIMULTANEOUS PK/PD - ADVAN6
; NOTE: EXAMPLE USES 1-COMPT PK;
;       MORE COMPLEX COMPARTMENTAL MODEL CAN BE USED
; CMT = 1 (OR 0) FOR DV = CP; CMT = 2 FOR DV = EFFECT
$DATA   data
$INPUT  ID TIME DV AMT=DOSE CMT
$SUBROUTINE ADVAN6 TOL=5
$MODEL
     COMP=(CENTRAL,DEFDOSE,DEFOBS)
     COMP=EFFECT
$PK
     K10= THETA(1)*EXP(ETA(1))
     V1=  THETA(2)*EXP(ETA(2))
     S1=  V1
     K20= THETA(3)*EXP(ETA(3))
     C50= THETA(4)*EXP(ETA(4))
     EMAX=THETA(5)*EXP(ETA(5))
$DES
     DADT(1)=-K10*A(1)                ;NOTE NO LOSS TO A(2)
     DADT(2)= K20*(A(1)/V1-A(2))      ;A(2) IS CE
$ERROR
   IF (CMT.EQ.2) THEN
   Y = EMAX*F/(C50+F)*EXP(ERR(2))   ; DV = EFFECT IF CMT=2
   ELSE
   Y = F*EXP(ERR(1))                ; DV = CP IF CMT=1 (OR 0)
   ENDIF
; ADD APPROPRIATE $THETA, $OMEGA, $SIGMA, $ESTIMATION, ETC.
```

REFERENCES: Guide IV, section V.C.6 , V.C.7
REFERENCES: Guide VI, section IV.B.2 , VII.C.6

PK PD SIMULTANEOUS 3 EXAMPLE

This is an example for simultaneous analysis of PK and direct PD. Kinetics are two-compartment. Either ADVAN5 or ADVAN7 could be used. (See **PK_PD_simultaneous_1**, **PK_PD_simultaneous_2**).

```
$PROBLEM   SIMULTANEOUS PK/PD - ADVAN7
; NOTE: MODEL FOR CP CAN BE MORE COMPLEX BY ADDING CMPTS
;
; THE DATA FILE CONTAINS BOTH CP AND EFFECT OBSERVATIONS.
; WHEN DV IS A CP OBSERVATION, CMT = 1 (OR 0),
; WHEN DV IS AN EFFECT OBSERVATION, CMT =3.
$DATA   data
$INPUT  ID TIME DV AMT=DOSE CMT
$SUBROUTINES ADVAN7
$MODEL
  COMP=(CENTRAL,DEFDOSE,DEFOBS)
  COMP=PERIPH
  COMP=EFFECT
$PK
  K10= THETA(1)*EXP(ETA(1))
  K12= THETA(2)*EXP(ETA(2))
  K13= .001*K10                    ; TRIVIAL LOSS TO EFFECT COMPT
  K21= THETA(3)*EXP(ETA(3))
  S1=  THETA(4)*EXP(ETA(4))        ; V1 FOR DRUG
  K30= THETA(5)*EXP(ETA(5))        ; KEO
  EMAX=THETA(6)*EXP(ETA(6))
  C50= THETA(7)*EXP(ETA(7))
  S3=S1*K13/K30                    ; SO THAT CESS = CPSS
$ERROR
   IF (CMT.EQ.3) THEN
    Y =EMAX*F/(C50+F)*EXP(ERR(2)) ; CMT = 3; DV = EFFECT
   ELSE
    Y =  F*EXP(ERR(1))             ; CMT = 1 (OR 0); DV = CP
   ENDIF
; ADD APPROPRIATE $THETA, $OMEGA, $SIGMA, $ESTIMATION, ETC.
```

REFERENCES: Guide IV, section V.C.6
REFERENCES: Guide VI, section IV.B.2 , VII.C.7

PK PD SIMULTANEOUS 4 EXAMPLE

This is an example for simultaneous analysis of PK and indirect PD, where the PREDPP PK prediction is modified by $ERROR code.

```
$PROB WARFARIN
$INPUT ID TIME DV AMT CMT
; DV = DRUG CONC OR  EFFECT (PROTHROMBIN TIME)
$DATA data
$SUBROUTINE ADVAN6 TOL=5
$MODEL
     COMP=(CENTRAL,DEFDOSE,DEFOBS)
     ; CAN ADD ADD'L COMPARTMENTS IF PK MULTICOMPARTMENTAL
     COMP=(CLOTFAC)
$PK
     K10=  THETA(1)*EXP(ETA(1))       ;LOSS RATE DRUG
     S1=   THETA(2)*EXP(ETA(2))       ;VD DRUG
     SYNTH=THETA(3)*EXP(ETA(3))       ;SYNTH RATE OF CLOTING FACTOR
     LOSS= SYNTH
     C50=  THETA(4)*EXP(ETA(4))
$DES
     DADT(1)=-K10*A(1)
     EFF    = C50/(C50+A(1)/S1)
     DADT(2)= SYNTH*EFF - LOSS*A(2)   ;CF LEVEL
$ERROR
     PT = THETA(5)/(THETA(6)+F)       ; HYPERBOLIC MODEL FOR PRO-TIME
     IF (CMT.EQ.2) THEN
       Y = PT*EXP(EPS(2))             ; OBS IS PRO-TIME
     ELSE
       Y = F*EXP(EPS(1))              ; OBS IS DRUG CONCENTRATION
     ENDIF
; ADD APPROPRIATE $THETA, $OMEGA, $SIGMA, $ESTIMATION, ETC.
```

The system should be initialized with a bolus dose of AMT=1 into compartment 2 (CLOTFAC) at time zero to establish the initial condition A(1)=0, A(2)=1 at time zero. This is appropriate because, in the absence of drug, the system is assumed to be at steady-state (DADT(2)=0).

REFERENCES: Guide IV, section V.C.6 , V.C.7
REFERENCES: Guide VI, section IV.B.2 , VII.C.6

PLASMA URINE EXAMPLE

Here are several examples of simultaneous modelling of plasma and urine data. The examples are adapted from a model for the analysis of Dextromethorphan data provided by Khaled Mohammed Abduljalil. The first involves data from the parent drug. The second involves both parent and metabolite data. In the control streams of these examples plasma and urine data are simulated. The same control streams could also be used for analysis of the data. MU_ modelling is used for the simulation to maintain consistency with subsequent analysis runs but MU_ modelling has no effect on simulation. If MU_ variables are renamed to avoid the character "_" (e.g., rename MU_1 as MU1), the same control streams may be run with NONMEM VI.

EXAMPLE 1 sim_parent_a2.ctl

The data dextroparent.dat contains data from only the parent drug. The parent drug is elimated directly to urine, and also via conversion to metabolite, but there are no measurements of metabolite in plasma or urine. The fraction F0 of parent drug in the urine is computed as in Guide V, Chapter 4, Section 3.1 equation (4.6) and Section 3.4.1 equation (4.7).
(See **Guide V, Chapter 4.3**)

The L2 data item is not required, but is used because multiple observations at the same TIME are in effect multivariate observations. Values of SIGMA corresponding to unused elements EPS(3) and EPS(4) are defined so that the EPS vector has the same length in both examples. See Remarks below.

Although ADVAN2 is used, the same control stream could be implemented with ADVAN5 or ADVAN6 or ADVAN7, with minor changes.

```
$PROBLEM  Parent drug, using ADVAN2
$INPUT ID TIME AMT UVOL DV CMT MDV EVID L2
$DATA dextroparent.dat IGNORE=#
$SUBROUTINES ADVAN2 TRANS1
$PK
  K12=THETA(1)
  KA=K12
  MU_1=LOG(THETA(2))
  V2=EXP(MU_1+ETA(1))
  MU_2=LOG(THETA(3))
  CLP=EXP(MU_2+ETA(2)) ; RENAL CL FOR PARENT
  CLB=THETA(4)         ; METABOLIC CL FOR METABOLIC
  K23=CLP/V2+CLB/V2
  K=K23
  F0=CLP/(CLP+CLB)
  S2=V2
  S3=UVOL
$ERROR (EVERY EVENT)
  ACMT=ABS(CMT) ; output compt. may have neg. value of CMT
  IF(ACMT.EQ.2) Y=F*(1+EPS(1))
  IF(ACMT.EQ.3) Y=F*(1+EPS(2))
$THETA
  (0.01,0.8,6) ;KA
  (0.01,43,1000);V2
```

```
  (0.0001,20,190);CLP
  (0.01,15,90);CLB
$OMEGA
  0.05  0.05
$SIGMA
  .01 .01
  .01 .01 ; eps(3), eps(4) for consistent EPS with sim_metab_a6 example
$SIM (111111) ONLYSIM
$TABLE ID TIME AMT UVOL DV SIMP=PRED CMT MDV
       EVID L2 NOAPPEND FILE=sim_parent_a2.tab
```

A fragment of the data file is as follows:

```
#  ID TIME  AMT        UVOL   DV    CMT    MDV   EVID  L2
2.000 0.000 30000.000  0.000  0.000  1.000 1.000 1.000 1
2.000 0.171     0.000   0.000 0.000  3.000 1.000 2.000 1
2.000 2.000     0.000   0.000 0.000  2.000 0.000 0.000 1
2.000 2.000     0.000  93.100 0.000 -3.000 0.000 0.000 1
2.000 2.000     0.000   0.000 0.000  3.000 1.000 2.000 2
2.000 3.150     0.000   0.000 0.000  2.000 0.000 0.000 2
2.000 3.150     0.000 133.600 0.000 -3.000 0.000 0.000 2
```

There is a dose at TIME=0. At TIME=.171, the urine compartment (which is the default compartment for output) is turned on. At TIME=2, there are observations of both plasma and urine. The urine amount is reset to 0 and the compartment turned back on for the start of a new urine collection. At TIME 3.15 there are observations of plasma and urine.

In the model, different EPS variables are used for the two types of observations. In the table, values of PRED (SIMP) are computed with ETA and EPS equal to 0. The DV values are computed with simulated ETA and EPS, and would be used as the DV in subsequent analysis runs.

EXAMPLE 2 sim_metab_a6.ctl

The data dextrometab.dat includes measurements of metabolite in plasma and urine. The model also include predictions of the metabolite in both plasma (CMT=4) and urine (CMT=5; the default compartment for output). Different compartments are used for parent and metabolite in urine, but the same value of UVOL applies to both. The F0 parameter is not needed because the model for K23 now uses only CLP, the clearance of the parent drug. No analytic ADVAN model has two output compartments, so a general ADVAN (ADVAN6 or ADVAN5 or ADVAN7) must be used.

Note that Compartment 3 is no longer the default for output. Instead of being computed by PREDPP using mass balance, it is computed explicitly by the ADVAN routine itself (e.g., using a differential equation when ADVAN6 is used). If compartment 3 is defined in $MODEL with default attributes, e.g.,

```
COMP=(DEXURIN)
```

there is an error messsage from PREDPP:

```
SPECIFIED COMPARTMENT MAY NOT BE TURNED OFF WITH AN OBSERVATION RECORD
```

Instead, the compartment may be defined as an output-type compartment:

```
COMP=(DEXURIN INITIALOFF NODOSE)
```

Now observations with CMT=-3 are permitted and no change to the original data records is needed.

```
$PROBLEM  Parent drug and metabolite, using ADVAN6
$INPUT ID TIME AMT UVOL DV CMT MDV EVID L2
$DATA dextrometab.dat IGNORE=#
$SUBROUTINES ADVAN6 TRANS1 TOL=4
$MODEL
  COMP=(DEPOT)
  COMP=(PLASMA DEFOBS) ; PARENT IN PLASMA
  COMP=(DEXURIN INITIALOFF NODOSE)  ;PARENT IN URINE
  COMP=METAB    ;METABOLITE IN PLASMA
$PK
  K12=THETA(1)
  MU_1=LOG(THETA(2))
  V2=EXP(MU_1+ETA(1))
  MU_2=LOG(THETA(3))
  CLP=EXP(MU_2+ETA(2)) ; RENAL CL FOR PARENT
  CLB=THETA(4)         ; METABOLIC CL FOR METABOLIC
  CLMR=THETA(5)        ; RENAL CL FOR METABOLITE
  V4=1
  K24=CLB/V2
  K23=CLP/V2
  ; F0=CLP/(CLP+CLB) Omit F0 because parent and
  ; metab have different urine compts.
  K45=CLMR/V4
  S2=V2
  S4=V4
  S3=UVOL
  S5=UVOL
$ERROR (EVERY EVENT)
  ACMT=ABS(CMT) ; output compart. may have neg. value of CMT
  IF(ACMT.EQ.2) Y=F*(1+EPS(1))
  IF(ACMT.EQ.3) Y=F*(1+EPS(2))
  IF(ACMT.EQ.4) Y=F*(1+EPS(3))
  IF(ACMT.EQ.5) Y=F*(1+EPS(4))
$DES
  DADT(1)=-K12*A(1)
  DADT(2)=K12*A(1)-K23*A(2)-K24*A(2)
  DADT(3)=K23*A(2)
  DADT(4)=K24*A(2)-K45*A(4)
$THETA
  (0.01,0.8,6) ;KA
  (0.01,43,1000);V2
  (0.0001,20,190);CLP
  (0.01,15,90);CLB
  (0.0001,5,90);CLMR
$OMEGA
  0.05  0.05
$SIGMA
  .01 .01 .01 .01
$SIM (111111) ONLYSIM
$TABLE ID TIME AMT UVOL DV SIMP=PRED CMT MDV EVID L2 NOAPPEND
```

```
        FILE=sim_metab_a6.tab
```

A fragment of the data dextrometab.dat is:

```
#  ID  TIME  AMT          UVOL  DV    CMT    MDV   EVID  L2
2.000  0.000 30000.000   0.000 0.000  1.000 1.000 1.000 1
2.000  0.171     0.000   0.000 0.000  3.000 1.000 2.000 1
2.000  0.171     0.000   0.000 0.000  5.000 1.000 2.000 1
2.000  2.000     0.000   0.000 0.000  2.000 0.000 0.000 1
2.000  2.000     0.000   0.000 0.000  4.000 0.000 0.000 1
2.000  2.000     0.000  93.100 0.000 -3.000 0.000 0.000 1
2.000  2.000     0.000  93.100 0.000 -5.000 0.000 0.000 1
2.000  2.000     0.000   0.000 0.000  3.000 1.000 2.000 2
2.000  2.000     0.000   0.000 0.000  5.000 1.000 2.000 2
2.000  3.150     0.000   0.000 0.000  2.000 0.000 0.000 2
2.000  3.150     0.000   0.000 0.000  4.000 0.000 0.000 2
2.000  3.150     0.000 133.600 0.000 -3.000 0.000 0.000 2
2.000  3.150     0.000 133.600 0.000 -5.000 0.000 0.000 2
```

Observations of metabolite in plasma (CMT=4) and metabolite in urine (CMT=5) are present at the same values of TIME as observations of parent in plasma (CMT=2) and parent in urine (CMT=3), respectively, and have the same values of L2.

Remarks:

After the control streams are run, the predictions SIMP for compartments 2 and 3 are the same in the two table files sim_parent_a2.tab and sim_metab_a6.tab, as expected, because they are computed with EPS=0, i.e., they are not simulated values. Because of the use of L2 and the same-length EPS vector in both examples, the simulated DV's for compartments 2 and 3 are also the same. This is useful for illustrative and debugging purposes, but is not usually the case when the EPS structure is different, or the L2 data item is not used.
To summarize:
With L2, during simulation SIMEPS generates a new set of values for EPS only when L2 changes value.
Without L2, during simulation SIMEPS generates a new set of values for EPS with every record, even when MDV is 1 or the PREDPP item EVID is not 0.

(See **simeps**).

All control streams and data files are found in the NONMEM 7.4 examples\dextro directory.

dextroparent.dat (also displayed here)
sim_parent_a2.ctl (also displayed here)
sim_parent_a5.ctl
sim_parent_a6.ctl
dextrometab.dat (also displayed here)
sim_metab_a5.ctl
sim_metab_a6.ctl (also displayed here)

REFERENCES: None.

RECORDS=ID EXAMPLE

These examples illustrate the use of the RECORDS=ID option of the $DATA record. (See **$data**).

Example 1 - Data checkout

During a checkout phase, one might want to see only the predictions for the first individual record, and therefore, use a data set comprised only of the data records therein. With RECS=ID, it is not necessary to count the number of records in the fist individual record.

```
$DATA filename RECS=ID
```

Example 2 - Data Analysis

One might want to analyze each subject's data independently of the others. One way this can be done is by creating a single control stream with as many problems as there are individual records in the data, and including one each of the following series of $DATA records in the series of problem specifications:

```
$DATA filename RECS=n1
$DATA filename RECS=n2 NOREWIND
$DATA filename RECS=n3 NOREWIND
etc.
```

where "nk" is the number of records in the kth individual record. It would be necessary to count the exact number of records. Now, these records can simply be coded

```
$DATA filename RECS=ID
$DATA filename RECS=ID NOREWIND
$DATA filename RECS=ID NOREWIND
etc.
```

Example 3 - Use of the INCLUDE record

With the INCLUDE record, the filename may now be followed by an integer n, whose default value is 1. Then NM-TRAN reads n copies of the named file.

In the example above if the problem specification for all subjects after the first are completely identical, a compact way of writing the control stream is possible. The control stream would contain the problem specification for the first individual, including the $SUBROUTINES record, abbreviated code, and the $DATA record for the first subject, and would end with (if there are e.g. 12 subjects in all)

```
INCLUDE ctlfile2 11
```

The file ctlfile2 would contain one problem specification (including $PROBLEM) for one subject, with no $SUBROUTINES record or abbreviated code, but would include

```
$DATA filename RECS=ID NOREWIND
```

A fully worked-out example is supplied as recid2.exa (See **recid2.exa**).

An alternate approach for single-subject data is available with NONMEM 7. OMEGA diagonal values are fixed to a special value 1.0E+06.
See Guide Introduction_7 "Single-Subject Analysis using Population with Unconstrained ETAs ".

REFERENCES: Guide Introduction_7

RECORDS=ID EXAMPLE 2

This fully-worked out example shows how the RECORDS=ID option of $DATA and the INCLUDE statement may be used to analyze the Theophyline data set THEOPP as separate individuals. The NONMEM control file is as follows.

```
$PROB  THEOPHYLLINE POPULATION DATA; Analysis of Individuals
; Modification of CONTROL5 control steam
$INPUT      ID DOSE=AMT TIME CP=DV WT
$DATA       THEOPP RECS=ID
;RECS=ID:  Data set will be read until ID changes or end-of-file

$SUBROUTINES  ADVAN2

$PK
;THETA(1)=MEAN ABSORPTION RATE CONSTANT (1/HR)
;THETA(2)=MEAN ELIMINATION RATE CONSTANT (1/HR)
;THETA(3)=SLOPE OF CLEARANCE VS WEIGHT RELATIONSHIP (LITERS/HR/KG)
;SCALING PARAMETER=VOLUME/WT SINCE DOSE IS WEIGHT-ADJUSTED
   CALLFL=1
   KA=THETA(1)
   K=THETA(2)
   CL=THETA(3)*WT
   SC=CL/K/WT

$THETA  (.1,3,5) (.008,.08,.5) (.004,.04,.9)
$OMEGA .4
;For single subject data OMEGA is residual variance.

$ERROR
   Y=F+ERR(1)
;ERR must be used instead of EPS.

$EST MAXEVAL=450  PRINT=5

$COV SPECIAL
;SPECIAL is required to obtain the variance-covariance matrix for
;single-subject data.

$TABLE ID DOSE WT TIME NOPRINT ONEHEADER FILE=indest.tab

$TABLE ID KA K CL SC NOPRINT FIRSTONLY ONEHEADER NOAPPEND
       FILE=indest.par
INCLUDE indest.txt 11
; INCLUDE: Inserts copies of the file named indest.txt for each
; additional individual.
```

The contents of file indest.txt are:

```
$PROB  THEOPHYLLINE POPULATION DATA; Analysis of Individuals
$INPUT      ID DOSE=AMT TIME CP=DV WT
```

```
$DATA       THEOPP RECS=ID NOREWIND
;NOREWIND: data set will be read starting after the previous individual

$THETA  (.1,3,5) (.008,.08,.5) (.004,.04,.9)

$OMEGA .4
;For single subject data OMEGA is residual variance

$EST MAXEVAL=450  PRINT=5

$COV SPECIAL
;SPECIAL is required to obtain the variance-covariance matrix for
;single-subject data

$TABLE ID DOSE WT TIME NOPRINT ONEHEADER FORWARD FILE=indest.tab

$TABLE ID KA K CL SC NOPRINT FIRSTONLY ONEHEADER FORWARD NOAPPEND
  FILE=indest.par
```

An alternate approach for single-subject data is available with NONMEM 7. OMEGA diagonal values are fixed to a special value 1.0E+06.
See Guide Introduction_7 "Single-Subject Analysis using Population with Unconstrained ETAs ".

REFERENCES: Guide Introduction_7

REPETITION 1 EXAMPLE

This example illustrates the use of the repetition feature, and in particular, the use of the variables RPTI and RPTO. For a discussion of these reserved variables: (See **Repetition Variables**).

We suppose that a compartment amount A2 at time t1 is given by a convolution, i.e. an integral with respect to s, of the product D(t1-s) and r(s), where D is a unit distribution function, and r is an input function. The integral is taken from 0 to t1. The amount in question can be computed by integrating the differential equation

dA2/ds=D(t1-s)r(s)

from time 0 to time t1. For A2 at a later time t2, one integrates a different differential equation:

dA2/ds=D(t2-s)r(s)

from time 0 to time t2. This equation differs from the first in so far as the constant t2, rather than t1, is used in D.

Ordinarily, PREDPP will use a system of differential equations

dA2/ds=f(A2,t)

to obtain the value of a vector A2 of compartment amounts at times t1 and t2, by first integrating the system to time t1, and then by integrating the same or different system from time t1 to time t2. This scheme cannot be used with the two equations described above, because the second equation - with the constant t2 - applies throughout the time interval 0 to t1, as well as throughout the interval t1 to t2.

PREDPP requires the event records to be ordered chronologically in the data set, and with its ordinary usage with differential equations (described above), these records are processed sequentially, starting with the first event record (which, for the purposes of this discussion, can be thought to have time 0). The records are passed sequentially to the PK routine.

The differential equations used for the convolution requires something special. During a regular sequential pass of the event records to the PK routine, when a record R with time t - at which compartment amounts are desired - is initially processed and passed to the PK routine for the first time, the routine should set the output variable RPTO to -1. This signals to NONMEM that the regular pass should be interrupted and a second pass of the records should be initiated, starting with the first event record once again and ending with record R (thus the pass of *these* records is "repeated"). Only at the end of the second pass, after *all* event records up to and including record R have once again been passed, will the value for A2(t) be established. The amounts at times on preceeding records have already been established, and they remain unchanged as the pass of these records is repeated. After R has been passed for the second time, the first pass will be continued, the next event record will be passed for the first time, and RPTO can again be set to -1. Whenever an event record is passed for the first time, f(s) should be set to D(t-s)r(s), where t is the time on the record, so that during the second pass, the differential equation that is used is the desired one. When an event record is passed, if the input variable to the

PK routine, RPTI, has the value 0, this signals that the record is not being passed as a part of a repeated pass, and so this can be used as a signal that the record is being passed for the first time.

Suppose the kinetic system is a two-compartment system where drug absorption from a depot compartment into a sampling compartment occurs in a first-order manner, and drug elimination from the sampling compartment occurs according to a Weibull unit disposition function D(s)=exp(-K(s**c)). Suppose too that the superposition principle is assumed to hold. So drug amount is given by a convolution. Suppose the event records for an individual are simply structured, e.g.

```
# TIME AMT DV     EVID
  .0    10   .      1              rec 1
  .25   .  2.44     0              rec 2
  .5    .  5.24     0              rec 3
  .75   .  5.57     0              rec 4
  1     .  5.85     0              rec 5
```

Then the control stream should look like:

```
$PROB
$INPT TIME AMT DV EVID
$DATA data
$SUB ADVAN6 TOL=5
$MODEL COMP=(DEPOT DEFDOSE) COMP=(CENTRAL DEFOBS)

$INFN
IF (ICALL.EQ.0) RPTO=1      ;enables use of repetition feature
$PK
KA=THETA(1)*EXP(ETA(1))
K =THETA(2)*EXP(ETA(2))
C =THETA(3)
V =THETA(4)*EXP(ETA(4))
S2=V
IF (RPTI.EQ.0) TI=TIME
IF (NEWIND.EQ.2) RPTO=-1

$DES
DADT(1)=-KA*A(1)
D=EXP(-K*(TI-T)**C)
DADT(2)=D*KA*A(1)

etc.
```

Note that when TIME= 0.25, RPTO need not have been set to -1; after the second event record has been passed for the first time, the kinetics are advanced from time 0 to time 0.25 using TI=0.25.

(1) The above is supplied as repeat1.ctl and repeat1.dat in the examples directory

The code

```
IF (ICALL.EQ.0) RPTO=1      ;enables use of repetition fea-
ture
```

is needed to inform NONMEM that the repetition feature will be used.

The code

```
IF (NEWIND.EQ.2) RPTO=-1
```

initiates a pass starting with the first event record, which is the default with RPTO=-1.

(2) Another example, repeat1t.ctl, demonstates that a repeated pass may start with any designated record, not necessarily the first record. The statement

```
IF (NEWIND.NE.2) RPTO=1
```

is used to indicate explicitly that the first record is a repetition base.

(3) Example repeat1s.ctl uses the RPT_ data item for the same purpose. This is repeat1s.ctl and repeat1s.dat in the examples directory. A non-zero value of RPT_ marks a record as a repetition base. A statement setting RPTO=-n (where n is the value of the RPT_ data item) requests repetition starting with the marked record. When RPT_ is used, it automatically enables use of the repetition feature and the $INFN block may be omitted.

The data is augmented with the RPT_ item, as in this fragment:

```
# TIME AMT DV     EVID RPT_
1  .0    10   .       1    1          rec 1
1  .25   .  2.44     0    .          rec 2
```

and the control stream contains

```
 $INPT ID TIME AMT DV EVID RPT_
```

(4) A different example of the use of RPT_ is given in repeatf.ctl and repeatf.dat. Reserved variables RPTO=n and RPTO=-n are intended to mark and repeat a sequence of more than 1 data records. If only one record is to be repeated, RPT_ must be used. The example computes the factorial of a value supplied as KRPT in the data record. A fragment of the data is

```
ID TIME DV  MDV RPT_ KRPT
1  1    1     0    .    .
1  2    2     0    1    4
```

The code sets

```
   IF (RPT_.EQ.1) RPTO=-1
   RPTON=KRPT
```

NONMEM calls PRED KRPT+1 times for each record with RPT_=1. This allows the factorial to be computed. This is intended as an illustration of what can be done using RPT_ data item. With NONMEM 7.3, factorials can be computed more quickly using the DOWHILE feature and the GAMLN function.

REFERENCES: none.

REPETITION 2 EXAMPLE

This example illustrates the use of the repetition feature, and in particular, the use of the PRED output control flag PRDFL. For a discussion of the reserved variables RPTO, RPTI and PRDFL: (See **Repetition Variables**).

This example computes a variable RS to be output in a table, defined as the difference between the observation in the data record and the average observation in the individual record. User-defined tabled quantities are computed during a copying pass through the data set - signaled by COMACT=1 (See **comact**). In this example, during the copying pass, the data records in each individual record are actually passed twice; the average is computed during a first pass through the individual record - signaled by RPTI=0, and the difference is computed during a second (repeated) pass - signaled by RPTO=1.

```
$PRED
Y=...
IF (COMACT.EQ.0) PRDFL=1
IF (ICALL.EQ.0) THEN
   RPTO=1      ;initiates use of repetition feature
   PRDFL=1     ;initiates use of the PRED output control flag
   N=0
   AV=0
ENDIF
IF (COMACT.EQ.1) THEN
   IF (RPTI.EQ.0.AND.NEWIND.LE.1) THEN
      N=0
      AV=0
   ENDIF
   IF (RPTI.EQ.0.AND.MDV.EQ.0) THEN
      N=N+1
      AV=AV+DV
   ENDIF
   IF (RPTI.EQ.1) THEN
      RS=0
      PRDFL=1
   ENDIF
   IF (RPTI.EQ.1.AND.MDV.EQ.0) RS=DV-AV
   IF (NDREC.EQ.LIREC) THEN
      RPTO=-1          ;RPTO=-1 is a signal initiating
                       ;a second pass. As RPTON isn't also set,
                       ;there are no further passes.
      AV=AV/N
   ENDIF
ENDIF

$TABLE RS NOAPPEND
```

The variable NDREC contains the index of the current data record within the current individual record, and LIREC contains the index of the last data record within the current individual record. (See **Record Counters: NIREC,NDREC**)
(See **Size of Individual Record**).
If a data record does not contain an observation (i.e. MDV=1), this code sets RS to 0.

If a variable is to be used only during a copying pass, but recursively across data records (such as N and AV), then it should be defined in an initialization block (e.g. at ICALL.EQ.0). Setting PRDFL=1 during a noncopying pass, assures that during such passes, PRED output will be accepted in the usual way: data record by data record.

Setting PRDFL=1 during a copying pass, but only with the second pass through the data records, assures that PRED output (which includes the RS variable) will be accepted data record by data record, but only during the repeated pass.

REFERENCES: none.

SIMULATION 1 EXAMPLE

This is an example of simulation with population data. The presence of the $SIMULATION record causes NONMEM to generate DV data items. The data set in file SIMORIG may contain nulls in the DV column.

Table file SIMDATA1 is created. It is similar to SIMORIG, except that the DV column contains the simulated observations. Because of the option NOHEADER of $TABLE, table header lines are not present, and, with the option NOPRINT, the table is not included in the NONMEM output report. SIMDATA1 can be used in subsequent runs as an input file for NONMEM analysis without further modification. These runs can involve different initial estimates, different models, etc.

```
$PROBLEM Simulation of population data
$INPUT ID TIME WT AMT DV
$DATA SIMORIG IGNORE=#
$SUBROUTINE ADVAN1
$PK
     CL=THETA(1)*EXP(ETA(1))
     V=THETA(2)*EXP(ETA(2))
     K=CL/V
     S1=V
$ERROR
     Y=F+F*EPS(1)
$THETA  .0625  10
$OMEGA .09 .05
$SIGMA .01
$SIMULATION (9215690) ; seed 1-7 digits
$TABLE ID TIME WT AMT NOPRINT FILE=SIMDATA1 NOHEADER
```

Contents of SIMORIG (for the first two individuals) follow.

```
#  ID TIME WT AMT DV
    1   0  80 100  .
    1   1  80   .  .
    2   0  75 100  .
    2   2  75   .  .
```

Contents of SIMDATA1 follow, with floating-point notation converted to fixed-point notation for the convenience of the reader. With versions of NONMEM prior to 7.4, this help item contained an incorrect set of values. The correct values are as follows.

```
1.    0.   80.  100.    0.000   10.000    0.000    0.000
1.    1.   80.    0.   13.646    9.938    3.708    1.531
2.    0.   75.  100.    0.000   10.000    0.000    0.000
2.    2.   75.    0.    8.379    9.876   -1.496   -0.625
```

The last four columns are automatically included by NONMEM in every table (unless the NOAPPEND option is used). They are: DV (simulated values), PRED (calculated using zeros for etas and epsilons), RES, and WRES. The DV, RES, and WRES colunns are zero for dose event records. The last three columns may be ignored (or the DROP option of $INPUT may be used) when the file is subsequently used. Alternately, list DV on the $TABLE record, and include the option NOAPPEND.

It is possible to display simulated values of PRED-defined variables. With NM-TRAN, simulated etas are automatically displayable.

Add the following to the $ERROR block to capture simulated epsilons as ERROR-defined items:

```
  EP1=EPS(1)
```

One may change the $SIMULATION and $TABLE records as follows.

```
$SIMULATION (9215690) ONLYSIM
$TABLE ID TIME WT AMT CL V ETA1 ETA2 EP1
       NOPRINT FILE=SIMDATA1 NOHEADER
```

Displayed values of CL, V, ETA1, ETA2, and EP1 are simulated values. Without the ONLYSIM option of $SIMULATION, the displayed values are typical values.

(See **Simulation example 2**, **Simulation example 3**).

REFERENCES: Guide IV, section III.B.13 , IV.I
REFERENCES: Guide V, section 12.4.8
REFERENCES: Guide VI, section III.C , III.E , IV.B , IV.G.1

SIMULATION 2 EXAMPLE

This is an example of simulation with population data. It is a continuation of Simulation Example 1.

First, during the Simulation Step, values of the covariate WT are simulated, as well as DV values, as PK-defined items SIMWT. Normally-distributed values of ETA(3) are used. Values of CL are calculated based on SIMWT. SIMWT replaces WT in the table file, and, because the ONLYSIM option is used, tabled values of SIMWT are simulated values.

Second, this process is repeated 10 times, using the same model, by means of the SUBPROBLEMS option. The table file contains the concatenated output of all the subproblems, and is the simulated data for 20 individuals.

```
$PROBLEM Simulation of population data and weight
$INPUT ID TIME WT AMT DV
$DATA SIMORIG IGNORE=#
$SUBROUTINE ADVAN1
$PK
     SIMWT=70+70*ETA(3)
     CL=THETA(1)*SIMWT*EXP(ETA(1))
     V=THETA(2)*EXP(ETA(2))
     K=CL/V
     S1=V
$ERROR
     Y=F+F*EPS(1)
$THETA  .0625  10
$OMEGA .09 .05 .04
$SIGMA .01
$SIMULATION (9215690) ONLYSIM SUBPROBLEMS=10
$TABLE ID TIME SIMWT AMT NOPRINT FILE=SIMDATA2 NOHEADER
```

SIMORIG is the same as with Simulation Example 1. Contents of SIMORIG (for the first two individuals) follow.

```
#  ID TIME WT AMT DV
    1   0  80 100  .
    1   1  80   .  .
    2   0  75 100  .
    2   2  75   .  .
```

Contents of SIMDATA2 follow, with floating-point notation converted to fixed-point for the convenience of the reader.

```
1.    0.   77.486  100.    0.000   10.000    0.000    0.000
1.    1.   77.486    0.    6.825    6.457    0.369    0.000
2.    0.   72.882  100.    0.000   10.000    0.000    0.000
2.    2.   72.882    0.    3.515    4.169   -0.653    0.000
```

REFERENCES: Guide IV, section III.B.13 , IV.I
REFERENCES: Guide V, section 12.4.8
REFERENCES: Guide VI, section III.C , III.E , IV.B , IV.G.1

SIMULATION 3 EXAMPLE

This is an example of simulation and estimation within the same NONMEM run. It is a continuation of Simulation Example 2.

As in Simulation Example 2, values of the covariate WT are simulated during the Simulation Step. They also replace the original WT values in NONMEM's internal copy of the data set. This transgeneration makes them available in the Estimation and Table Steps.

There are two models for V, illustrating the use of diferent models in the Simulation and Estimation Steps:

(simulated V) `VS = THETA(2) * EXP(ETA(3))`
(non-simulated V) `VN = THETA(3) + ETA(4)`

There are two epsilons in the $ERROR block, illustrating the use of an of using an initial estimate for estimation different from the value used to simulate.

Because theta(2), eta(2), eta(3) and eps(1) are used only in the Simulation Step, the FIXED attribute is specified for theta(2), omega(2,2), omega(3,3) and sigma(1,1) during the Estimation Step.

```
$PROBLEM Simulation of population data w/ covariates
$INPUT ID TIME WT AMT DV
$DATA SIMORIG IGNORE=#
$SUBROUTINE ADVAN1
$PK
      IF (ICALL.EQ.4) THEN
      WT=70+70*ETA(3)
      ENDIF
      CL=THETA(1)*WT*EXP(ETA(1))
      IF (ICALL.EQ.4) THEN
       V=THETA(2)*EXP(ETA(2))  ; Simulated V
      ELSE
       V=THETA(3)+ETA(4)       ; Non-simulated V
      ENDIF
      K=CL/V
      S1=V
$ERROR
      IF (ICALL.EQ.4) THEN
       Y=F+F*EPS(1)
      ELSE
       Y=F+F*EPS(2)
      ENDIF
$THETA  (0,.0625)  (10 FIX) (0,12)
$OMEGA .09 (.05 FIX) (.04 FIX) 4
$SIGMA (.01 FIX) .03
$SIMULATION (9215690) SUBPROBLEMS=10
$ESTIMATION
$TABLE ID TIME WT AMT NOPRINT FILE=SIMDATA3 NOHEADER
```

REFERENCES: Guide IV, section III.B.13 , IV.I
REFERENCES: Guide V, section 12.4.8

REFERENCES: Guide VI, section III.C , III.E , IV.B , IV.G.1

STIELTJES EXAMPLE

The following are examples of the specification of the Stieltjes method.

```
$EST PRINT=1 MET=1 STIELTJES
$EST MAX=9999 METH=1 LAPLACE LIKE PRINT=1 STIELTJES
$EST MAX=9999 METH=1 LAPLACE LIKE PRINT=1 STIELTJES GRID=(9,10,.1,.9)
$EST MAX=500 MET=COND LAPLACE LIKE STIELTJES GRID=(20,2,.1,.9)
```

REFERENCES: Guide Introduction_7

STOCHASTIC DIFFERENTIAL EQUATION EXAMPLES

A number of examples are given of the use of Stochastic Differential Equations.

sde6
: Example problem without sde

sde7
: Example problem with SDE, using expanded data set that have repeated lines to re-invoke PRED several times for each data record.

sde8
: Use of XVID to re-invoke PRED several times per data record wthout requiring the redundancy of sde7.csv.

sde9
: Alternative method of sde programming, using add-on sde.f90

They, along with the data files, can be found in the examples directory of the NONMEM 7 distribution medium.

See **INTRODUCTION TO NONMEM 7, Repeated Observation Records**
See **INTRODUCTION TO NONMEM 7, Stochastic Differential Equation Plug-In**

REFERENCES: Guide Introduction_7

SUPERPROBLEM 1 EXAMPLE

There are four problems:

Problem 1: The (true) probability that with an individual chosen at random, after a 200 mg dose, the true concentration at 5 hrs is above 5, is computed using simulation.
Problem 2: A population data set is simulated.
Problem 3: Parameter estimates are obtained.
Problem 4: The probability in question is estimated, using simulation and the parameter estimates obtained in problem 3.

Problems 2-4 comprise a superproblem. The bias in the estimator defined by problem 4 is estimated by iterating this superproblem.

```
$PROB    COMPUTE PERCENTAGE OF POP. WITH CP>5 AT DOSE=200, TIME=5
$INPUT ID TIME DOSE DV
$DATA dataB
$PRED
IF (ICALL.EQ.0) THEN
   BIAS=0           ; initialize
   TRUE=0
   OPEN (50,FILE='BIAS') ; Give the output file a name
ENDIF
IF (ICALL.EQ.1) THEN
   IF (IPROB.EQ.1.OR.IPROB.EQ.4) N=0   ; see rocm14
   CALL SUPP (1,1)
ENDIF
      KA=THETA(1)*EXP(ETA(1))
      KE=THETA(2)*EXP(ETA(2))
      V=THETA(3)*EXP(ETA(3))
      A=EXP(-KE*TIME)
      B=EXP(-KA*TIME)
      C=KA-KE
      D=A-B
      E=KA*DOSE/(V*C)
      F=E*D
      Y=F+ERR(1)
IF (ICALL.EQ.4) THEN
   IF (IPROB.EQ.1.OR.IPROB.EQ.4) THEN
      IF (F.GT.5) N=N+1
      IF (IREP.EQ.NREP) THEN           ; see rocm10
         PER=100.*N/NREP
         IF (IPROB.EQ.1) TRUE=PER
         IF (IPROB.EQ.4) THEN
            BIAS=BIAS+(PER-TRUE)/TRUE
            IF (S1IT.EQ.S1NIT) THEN
               BIAS=BIAS/S1NIT
               WRITE (50,*) BIAS
            ENDIF
         ENDIF
      ENDIF
   ENDIF
ENDIF
$THTA (.4,1.7,7) (.025,.102,.4) (10,29,80)
$OMEGA .04 .04 .04
$SIGMA 1.5
$SIM (5566898) ONLY SUB=1000
```

```
;
$SUPER  SCOPE=3  ITERATIONS=10
$PROB    SIMULATION
$INPUT ID TIME DOSE DV
$DATA dataA
$THTA (.4,1.7,7) (.025,.102,.4) (10,29,80)
$OMEGA .04 .04 .04
$SIGMA 1.5
$SIM (-1) ONLY
$TABLE ID TIME DOSE DV FILE=simulation NOHEADER NOPRINT NOFORWARD
;
$PROB    ESTIMATION
$INPUT ID TIME DOSE DV
$DATA simulation (4E12.0) NRECS=500 NOOPEN
$THTA (.4,1.7,7) (.025,.102,.4) (10,29,80)
$OMEGA .04 .04 .04
$SIGMA 1.5
$EST PRINT=0 MSFO=msf
;
$PROB    ESTIMATE PERCENTAGE OF POP. WITH CP>5 AT DOSE=200, TIME=5
$INPUT ID TIME DOSE DV
$DATA dataB
$MSFI msf
$SIM (-1) ONLY SUB=100 TRUE=FINAL
```

Comments:

(1) Variables whose values are to be retained across problems (e.g. BIAS and TRUE) should be defined in initialization blocks. In this instance these variables are defined at ICALL=0, before any of the problems are implemented

(2) IREP and NREP are the number of the current replication and number of requested replications for the Simulation Step, respectively.

(3) S1IT and S1NIT are the number of the current iteration and number of requested iterations for the (level 1) superproblem, respectively.

(4) Problems 2 & 3 can, of course, be combined, making the control stream much simpler. But separation can sometimes be useful, and here we illustrate how this can be done.

(5) The data set of problem 3 can be taken to be the internal data set created with problem 2 simply by removing both the $TABLE record of problem 2 and the $DATA record of problem 3. But these records are included here to illustrate the ability to use a table file as a data file in a subsequent problem of the same run. When so doing, all the items in the $INPUT record of the subsequent problem should be included in the $INPUT record of the problem with the $TABLE record. (If a PRED-item in the table is to be used in the subsequent problem as a data item, the PRED-item name can be different from the data item name.) The FORMAT and NOOPEN options should be included in the $DATA record of the subsequent problem. The NRECS option is not needed, but if it is also included, it should be set to the number of records in the table file, i.e. to the number of data records in data set A (assumed here to be 500).

Contents of dataA (for individual 1 only) follow:

```
1    .27   320.   .
1    .52   320.   .
1   1.     320.   .
1   1.92   320.   .
```

```
1   3.5    320.   .
1   5.02   320.   .
1   7.03   320.   .
1   9.     320.   .
1  12.     320.   .
1  24.3    320.   .
```

This data set cannot be run as-is. File dataA must include more subjects. If not, the estimation step in problem 3 will not terminate correctly. The final problem will produce the message:

```
THE SIMULATION TASK REQUIRES THAT A BETTER FINAL ESTIMATE
 BE AVAILABLE IN THE MODEL SPECIFICATION FILE
OR
MULTIPLE SUB_PROBLEMS IS INCOMPATIBLE WITH OTHER INFORMATION
 IN THE CONTROL STREAM AND INPUT MSF
```

This error message is related to an earlier message

```
PARAMETER ESTIMATE IS NEAR ITS BOUNDARY
THIS MUST BE ADDRESSED BEFORE THE COVARIANCE STEP CAN BE IMPLEMENTED
```

A larger version of dataA can be found in the examples directory of the NONMEM distribution medium. The content of dataA above is replicated 28 times. ID values 1 and 2 alternate so that there are 28 subjects.

Contents of dataB follow:

```
1   5.0    200.   .
```

Comments on NONMEM 7 changes:

In this example as distributed with earlier versions of NONMEM, the WRITE statement specified unit 42:

```
WRITE (42,*) BIAS
```

With NONMEM 7, this must be changed to

```
WRITE (50,*) BIAS
```

This avoids a possible NONMEM error message

```
FORTRAN UNIT BEING OPENED OR CLOSED IN PROBLEM        1 IS OUTSIDE
ACCEPTABLE RANGE (50-2000)
```

A new statement is added:

```
OPEN (50,FILE='BIAS')
```

This causes the output to be written to a file named BIAS rather than fort.50 or the like.

REFERENCES: None.

SUPERPROBLEM 2 EXAMPLE

Two problems, in order to effect a standard two-stage analysis:

Problem 1: First stage analysis
Problem 2: Obtain simple population average of PK parameter.

Problem 2 comprises a superproblem. It is implemented once for CL and once for V, and can be implemented more times when there are more parameters of interest.

```
$PROBLEM TWO-STAGE  STAGE 1
$DATA dataD
$INPUT ID DOSE TIME DV=CP
$PRED
IF (IPROB.EQ.1.AND.NEWIND.LE.1) D=DOSE
IF (IPROB.EQ.1) THEN
   CL=THETA(1)*EXP(ETA(1))
   V =THETA(2)*EXP(ETA(2))
   K=CL/V
   F=D*EXP(-K*TIME)/V
   Y=F+F*EXP(ETA(3))*ERR(1)
ENDIF
IF (ICALL.EQ.4) THEN
; if S1IT >= 1 then problem 2 is being implemented.
; see comment 1 below.
   IF (S1IT.GE.1) THEN
"       IF (S1IT.EQ.1) DPV=DATREC(2)
"       IF (S1IT.EQ.2) DPV=DATREC(3)
        Y=DPV
        RETURN
   ENDIF
ENDIF
IF (S1IT.EQ.1) Y=THETA(1)+ERR(1)
IF (S1IT.EQ.2) Y=THETA(2)+ERR(2)
$THETA 5 50
$OMEGA 100 100 100
$SIGMA 1 FIX 0 FIX
$ESTIMATION MAXEVAL=0 POSTHOC METH=COND INTERACTION
$TABLE ID CL V  FILE=stage2 NOHEADER NOPRINT FIRST NOFORWARD
;
$SUPER  SCOPE=1 ITERATION=2
$PROBLEM TWO-STAGE  STAGE 2
$INPT ID CLI VI DV
$DATA stage2 (4F12.0) NRECS=12 REWIND NOOPEN
$THETA 5 50
$OMEGA 0 FIX 0 FIX 0 FIX
$SIGMA 1 1
$SIMULATION (1)
$ESTIMATION   ;population estimates are obtained
```

Comment:

(1) The simulation step in problem 2 is used to select the DV data item from one of the several possibilities (CL or V) output by problem 1.

(2) Using generated code, data item names that do not appear in the $INPUT record of problem 1, such as CLI and VI, cannot be used in abbreviated or verbatim code. DATREC is a vector-array of data items that can be referenced in verbatim code.

Contents of dataD (for individual 1 only) follow.

```
1.000 4.020 0.000 0.091
1.000 0.000 0.250 0.084
1.000 0.000 0.570 0.105
1.000 0.000 1.120 0.057
1.000 0.000 2.020 0.038
1.000 0.000 3.820 0.060
1.000 0.000 5.100 0.046
1.000 0.000 9.050 0.017
1.000 0.000 7.030 0.024
1.000 0.000 12.120 0.018
1.000 0.000 24.370 0.002
```

REFERENCES: None.

TDM EXAMPLE

This is an example of how F1 can be used to simplify the preparation of a data file when the dose is changed often, e.g., for Therapeutic drug monitoring (TDM).

Suppose the dosing pattern is consistent (e.g., every 12 hours) but the amount changes often. Suppose also there are observation events at various times.

In the following data set, there is only one dose record having AMT=1. ADDL and II are used to specify an (effectively infinite) number of additional doses, every 12 hours. When the dose amount changes, it is sufficient to supply a record giving the new dose amount in the DOSE data item. Such records have EVID=2. There is no need to interrupt the dosing pattern for an observation event, or to count the number of doses between observation events.

Suppose a fragment of the dose records looks like this:

ID	DAT1=DROP	TIME	EVID	DV	DOSE	AMT	II	ADDL
1	01.01.2008	20	1	0	4	1	12	99999
1	03.01.2008	8	0	6.1	0	0	0	0
1	04.01.2008	8	0	7.6	0	0	0	0
1	04.01.2008	20	2	0	3	0	0	0
1	06.01.2008	8	0	5.4	0	0	0	0

Here is a fragment of the control stream:

```
$INPUT ID DAT1=DROP TIME EVID DV DOSE AMT II ADDL
 ...
$PK (ADDITIONAL OR LAGGED)
 ...
IF (DOSE> 0) SAVEDOSE=DOSE
F1=SAVEDOSE
```

In $PK, the value of DOSE is saved, and used on subsequent calls to PK as the value of F1. Although F1 is usually used to model the bio-availability of doses to compartment 1, in this example it is the actual dose amount.

REFERENCES: Guide V, section 7.4.4

THREE COMPARTMENT EXAMPLE

This example shows how to implement a three Compartment Linear Model with First-order Absorption using PREDPP, where elimination occurs from a central compartment *and* one of two peripheral compartments.

As with any ADVAN, one or more doses could be input to any of the compartments. Steady state doses are also possible.

The use of the DEFDOSE attribute on the following $MODEL record specifies that if the CMT data item is null on a given dose event record, then the dose(s) specified by that record are to be placed in the depot compartment. The use of the INITIALOFF attribute specifies that the depot compartment is to be off until a dose is placed in it.

The use of the DEFOBS attribute on the $MODEL record specifies that if the CMT data item is null on a given observation event record, then the quantity F which is available as a right-hand quantity in $ERROR is to be the scaled drug amount in the central compartment. (See **CMT PCMT data item**).

```
$PROBLEM EXAMPLE OF A THREE COMPARTMENT MODEL WITH ABSORPTION
$INPUT   ID TIME AMT DV CMT
$DATA   datafile
$SUBROUTINES  ADVAN7
$MODEL COMP = (DEPOT,DEFDOSE,INITIALOFF)
       COMP = (CENTRAL,DEFOBS)
       COMP = PERIPH1,  COMP = PERIPH2
$PK
   K12 = THETA(1)*EXP(ETA(1))  ; depot to central
   K23 = THETA(2)              ; central to periph1
   K32 = THETA(3)              ; periph1 to central
   K30 = THETA(4)*EXP(ETA(2))
   K24 = THETA(5)              ; central to periph2
   K42 = THETA(6)              ; periph2 to central
   K20 = THETA(7)*EXP(ETA(3))  ; elimination from central
   S2  = THETA(8)*EXP(ETA(4))  ; scale for central
;There are other reasonable ways that etas can be assigned.
;Elaborate these expressions for the K's as is appropriate.
;Add $ERROR, parameter and task records as is necessary/desired.
```

Other parameterizations are possible, as long as the above set of micro-constants is defined. E.g., the definitions of K20 and S2 can be replaced by the following.

```
   CL  = THETA(7)*EXP(ETA(3))
   V   = THETA(8)*EXP(ETA(4))
   K20 = CL/V
   S2  = V
```

If the depot compartment is not needed, it can be omitted. The numberings of the compartments, the thetas and etas must then be adjusted.

REFERENCES: Guide VI, section VII.C.5 , VII.C.7

TIME AFTER DOSE (TAD) EXAMPLE

This fully-worked out example shows how Time After Dose (TAD) may be computed in $PK abbreviated code. It works with transient and steady-state doses, and also with additional and lagged doses.

It is based on suggestions from the NONMEM tips directory but is not identical to any of them. See Note 8 for a discussion of the tips.

Two versions of the code are given.

VERSION I

Code to compute TADE ("TAD Effective"). The time that the dose enters the system is used, which is the time the dose was administered plus ALAG.

VERSION II

Code to compute TADA ("TAD Administered"). The time that the dose was administered is used, and ALAG is ignored.

If PDxPop is used or any other software that looks for TAD in the table, specify TAD=TADA or TAD=TADE in the $TABLE record.

```
$PROB  TIME AFTER DOSE (Based on RUN# 705 nonmem coding challenge #1)
; This example shows how Time After Dose can be computed.
; It contains two independent versions of the computation.
; Version I computes TADE (Time After Dose Effective)
; If there is absorption lag, TADE is the Time after the lagged dose,
; that is, the time the dose actually enters the sysem.
; Version II computes TADA (Time After Dose Administered)
; Absorption lag is ignored.
;
; As of Dec. 17, 2018
;
$INPUT C ID TIME DV AMT WT AGE CRCL SMK ADDL II EVID DROP
$DATA 706.csv IGNORE=C
$SUBROUTINE ADVAN2 TRANS2
$PK
;
ALAG1=0  ;  ALAG1 is 0 for this example, but could be set to some other
;           value. With ALAG1=0, either version can be used.
;           However, VERSION I IS simpler and is preferred.
;
; VERSION I:  CODE TO COMPUTE TADE = TAD EFFECTIVE (USE ALAG)
;  PREDPP KEEPS TRACK INITIATING AND IMPLIED DOSES, AND COMPUTES DOSTIM
   IF (NEWIND.LT.2.OR.EVID.EQ.3) THEN
      TDOSE=-999 ; TIME OF MOST RECENT DOSE. -999 IF NO PREVIOUS DOSE.
      TADE=0.0   ; TIME AFTER DOSE EFFECTIVE
   ENDIF
   IF (EVID.EQ.1.AND.ALAG1.EQ.0.OR.EVID.EQ.4.AND.ALAG1.EQ.0) TDOSE=TIME
   IF (DOSTIM.GT.0) TDOSE=DOSTIM
   IF (TDOSE.GT.-999) TADE=TIME-TDOSE

; If no ALAG and no ADDL doses, the above three lines become:
;    IF (EVID.EQ.1.OR.EVID.EQ.4) TDOSE=TIME
;    IF (TDOSE.GT.-999) TADE=TIME-TDOSE
;
; End of VERSION I

; VERSION II: CODE TO COMPUTE TADA = TAD ADMINISTERED (IGNORES ALAG)
;  IF ALAG>0,  PREDPP's DOSTIM CANNOT BE USED FOR TADA
```

```
;  THE ABBREVIATED CODE MUST KEEP TRACK OF INITIATING AND IMPLIED DOSES.
;  THIS CODE IS SIMILAR TO tip4-new-general-01-31-18.txt
   IF (NEWIND.LT.2.OR.EVID.EQ.3) THEN
      TDOSA=-999 ; TIME OF MOST RECENT DOSE. -999 IF NO PREVIOUS DOSE.
      TADA=0.0   ; TIME AFTER DOSE ADMINISTERED
    ENDIF
  IF (DOSTIM==0) THEN ; IGNORE NON-EVENT DOSE TIMES (DOSTIM>0)
       IF(EVID.EQ.1.OR.EVID.EQ.4) THEN ; NEW DOSE EVENT RECORD
          TADA=0.0
          DIV=II
          TDOSA=TIME
          TLAST=TDOSA+ADDL*II ; TLAST IS THE TIME OF THE FINAL IMPLIED DOSE
       ENDIF
       IF (TDOSA.GE.0.AND.EVID.NE.1.AND.EVID.NE.4) THEN ; THERE WAS AN EARLIER DOSE
          IF (TIME>TLAST) THEN
             TADA=TIME-TLAST ; CURRENT TIME IS PAST THE TIME OF THE FINAL IMPLIED DO
          ELSE
             DIFF=TIME-TDOSA
             TADA=MOD(DIFF,DIV) ; COMPUTES TIME OF THE MOST RECENT IMPLIED DOSE
          ENDIF
       ENDIF ; NOT A DOSE
  ENDIF ; END OF DOSTIM==0
;
; End of VERSION II

  CL=THETA(1)*EXP(ETA(1))
  TVV=THETA(2)
  V=TVV*EXP(ETA(2))
  TVKA=THETA(3)
  KA=TVKA*EXP(ETA(3))
  S2=V
$THETA
  (0, 10)
  (0, 100)
  (0, 0.5)
$ERROR
  Y=F+ERR(1)
  IPRED=F
$OMEGA
  0.04 ;[P] INTERIND VAR IN CL
  0.04 ;[P] INTERIND VAR IN V
  0.04 ;[P] INTERIND VAR IN KA
$SIGMA
  0.2 ;[A] ADDITIVE COMPONENT
$EST MAXEVAL=0 PRINT=2 NOABORT ;POSTHOC
;
; If PDxPop is used or any other software that looks for TAD in the table, specify
; TAD=TADA or TAD=TADE in the $TABLE record
$TABLE ID TIME EVID TDOSA TADA TDOSE TADE IPRED NOAPPEND NOPRINT ONEHEADER
FILE=tadexa.tab FORMAT=SF11.7
```

The data for the first subject in 706.csv is:

```
C,Data Desc: test of TAD for ADDL,,,,,,,,,
C,ID,TIME,DV,AMT,WT,AGE,CRCL,SMK,ADDL,II,EVID,TAD
0,1,0,0,0,58.4,51,4.49,1,0,0,2,0
0,1,0.1,0,500,58.4,51,4.49,1,1,24,1,0
0,1,1.1,1.67538,0,58.4,51,4.49,1,0,0,0,0
0,1,4.1,2.79283,0,58.4,51,4.49,1,0,0,0,0
```

```
0,1,10.1,0.583263,0,58.4,51,4.49,1,0,0,0,0
0,1,24.1,0.146255,0,58.4,51,4.49,1,0,0,0,0
0,1,25.1,1.67538,0,58.4,51,4.49,1,0,0,0,0
```

The original data set 704.csv is embedded in tip3-2-25-02.txt. Data set 706.csv can be found in the tips directory. It was created from 704.csv by appending an extra column, TAD, whose values are zero. $INFN code in the tips assign values to the TAD data item (see Note 8). The code in the present example does not use the TAD data item (DROP is listed in $INPUT). Instead, TAD is a user-defined variable listed in MODULE NMPRD4 from which its values may be displayed in a table file.

Note 1.

It is assumed that TAD is displayed in tables for the purpose of graphical display of the data. It is not a random variable, and is not intended to be part of the prediction model.

Note 2.

If there are non-dose records prior to the first dose record of the Individual record, TAD is set to zero for these records. This is an arbitrary convention; the user could use different values, especially if these conditions might signal an error in the data set.

Note 3.

ALAG1 is used in the example, but a different dosing compartment could be used, e.g., ALAG2 for doses into compartment 2. The code would need modification to compute TAD for doses into two or more compartments.

Note 4.

This code can handle random DOSTIM (Eta on ALAG or ETA on modelled duration or rate.)

Note 5.

The code does not use INFN. The information is developed with passes through the data set at ICALL=2, in which NONMEM calls PREDPP. During a pass in which INFN is called, PREDPP is not called. The abbreviated code would have to account for additional and lagged dose times.

Note 6.

TDOS=-999 until the first dose. TDOS=0 when the first dose is at TIME 0. This makes it possible to dispense with a flag variable whose only function is to say "a dose has occurred". If there are negative time values in the data set and any time value is smaller than -999, then

Note 7.

The code is intended to compute TAD for multiple individual doses or one initiating dose. (An initiating dose has ADDL>0,II>0).

Note 8.

Changes were made to tip3, tip4, and tip4update in the NONMEM tips directory. See tips in

https://nonmem.iconplc.com/nonmem/tips

In each case, the name of the revised tip contains 01-31-18 and supercedes the original. Although the tips were revised as part of the nonmem 7.4.2 release they apply to previous versions of nonmem.

tip3-revised-01-31-18.txt

The original version, tip3-2-25-02.txt, provides a data file 704.csv and a simple

model. The revised version provides improved code, similar to the code in tadexa.exa.

tip4-revised-01-31-18.txt
: Uses Fortran suboutine INFN and equivalent $INFN code. The datafile is 706.csv.

tip4update-revised-01-31-18.txt
: Uses DOSTIM.

The original versions are still in the tips directory with new names that include the original dates.

tip3-2-25-02.txt
tip4-3-04-02.txt
tip4update-10-11-13.txt

REFERENCES: None

TNPRI EXAMPLES

These examples illustrate the use of the NONMEM utility routine TNPRI by the NM-TRAN $PRIOR record, and also by an equivalent user-supplied routine PRIOR. For a discussion of PRIOR: (See **prior**). For a discussion of the TNPRI: (See **tnpri**).

Example 1

Use of PRIOR and TNPRI and a control stream for population data:

This example obtains parameter estimates from population data, using a two-compartment PK model, and it incorporates a frequentist prior of transformed normal form for all of THETA, OMEGA and SIGMA. The prior information is found in model specification file msf1.

```
$PROB   READ THE MODEL SPECIFICATION FILE
$DATA data
$INPUT ID DOSE TIME DV WT
$PRIOR TNPRI (PROBLEM 2)
$MSFI msf1 ONLYREAD
$PRED
;
;      THETA(1)=MEAN ABSORPTION RATE CONSTANT - MEAN ELIM. RATE CONSTANT (1/HR)
;      THETA(2)=MEAN ELIM. RATE CONSTANT (1/HR)
;      THETA(3)=SLOPE OF CLEARANCE VS WEIGHT RELATIONSHIP (LITERS/HR/KG)
;      DOSE=WEIGHT-ADJUSTED DOSE (MG/KG)
;
       IF (NEWIND.NE.2) THEN
          AMT=DOSE*WT
          W=WT
       ENDIF
       T0=THETA(1)*EXP(ETA(1))
       T2=THETA(2)*EXP(ETA(2))
       T1=T2+T0
       T3=THETA(3)*W*EXP(ETA(3))
       A=EXP(-T2*TIME)
       B=EXP(-T1*TIME)
       C=T1-T2
       D=A-B
       E=T3*C
       Y=AMT*T1*T2/E*D+EPS(1)

$PROB POPULATION DATA WITH PRIOR ON THETA, OMEGA AND SIGMA
$INPUT ID DOSE TIME DV WT
$DATA data  REWIND

$THETA  (0,4,5) (0,.09,.5) (.004,.01,.9)
$OMEGA BLOCK (3) .7 .04 .05 .02 .06 .08
$SIGMA  .4

$EST
```

Instead of a $PRIOR record, the following may be used:

```
$SUBROUTINE PRIOR=prior
```

The prior routine is as follows:

```
      SUBROUTINE PRIOR (ICALL,CNT)
      USE SIZES,     ONLY: DPSIZE,ISIZE
```

```
      USE NMPRD_INT, ONLY: IPROB
      REAL(KIND=DPSIZE) :: CNT
      REAL(KIND=DPSIZE) :: PLEV
      INTEGER(KIND=ISIZE) :: ICALL
      IF (IPROB==2) THEN
      PLEV=0.
      ITYP=0
      NSAM=0
      ISS=0
      IFND=0
      MODE=0
      IVAR=0
      CALL TNPRI(IFND,MODE,ITYP,PLEV,NSAM,ISS,IVAR,CNT)
      ENDIF
      RETURN
      END
```

Example 2

Use of PRIOR and TNPRI and a control stream for population data:

This example obtains parameter estimates from population data, using a two-compartment PK model, and it incorporates a frequentist prior of transformed normal form for all of THETA and OMEGA, but for only the first element of a diagonal SIGMA, which is a prior-specific parameter. Prior information for these parameters is found in model specification file msf1.

```
$PROB   READ THE MODEL SPECIFICATION FILE
$DATA data
$INPUT ID DOSE TIME DV WT
$PRIOR TNPRI (PROBLEM 2)
$MSFI msf1 ONLYREAD
$PRED
;
;     THETA(1)=MEAN ABSORPTION RATE CONSTANT - MEAN ELIM. RATE CONSTANT (1/HR)
;     THETA(2)=MEAN ELIM. RATE CONSTANT (1/HR)
;     THETA(3)=SLOPE OF CLEARANCE VS WEIGHT RELATIONSHIP (LITERS/HR/KG)
;     DOSE=WEIGHT-ADJUSTED DOSE (MG/KG)
;
      IF (NEWIND.NE.2) THEN
         AMT=DOSE*WT
         W=WT
      ENDIF
      T0=THETA(1)*EXP(ETA(1))
      T2=THETA(2)*EXP(ETA(2))
      T1=T2+T0
      T3=THETA(3)*W*EXP(ETA(3))
      A=EXP(-T2*TIME)
      B=EXP(-T1*TIME)
      C=T1-T2
      D=A-B
      E=T3*C
      Y=AMT*T1*T2/E*D+EPS(2)

$PROB POPULATION DATA WITH PRIOR ON THETA AND OMEGA
$INPUT ID DOSE TIME DV WT
$DATA data REWIND
$THETA  (0,4,5) (0,.09,.5) (.004,.01,.9)
$OMEGA BLOCK (3) .7 .04 .05 .02 .06 .08
$SIGMA  .4 FIX .5
```

```
$EST
```

Instead of a $PRIOR record, use the same $SUBROUTINE record and the same subroutine prior as in Example 1.

Example 3

Use of PRIOR and TNPRI and a control stream for population data:

This example simulates the THETA and OMEGA parameters of the same two-compartment population PK model, as well as the population data from this model. As with Example 2, the first element of SIGMA is prior-specific, and so its value need not be simulated. The value .5 is used for the value of the second element of SIGMA. The default variance-covariance matrix from the prior problem is used for the hyperparameter variance-covariance matrix (IVAR=1).

```
$PROB   READ THE MODEL SPECIFICATION FILE
$DATA data
$INPUT ID DOSE TIME DV WT
$PRIOR TNPRI (PROBLEM 2) PLEV=.9999 IVAR=1
$MSFI msf1 ONLYREAD
$PRED
;
;      THETA(1)=MEAN ABSORPTION RATE CONSTANT - MEAN ELIM. RATE CONSTANT (1/HR)
;      THETA(2)=MEAN ELIM. RATE CONSTANT (1/HR)
;      THETA(3)=SLOPE OF CLEARANCE VS WEIGHT RELATIONSHIP (LITERS/HR/KG)
;      DOSE=WEIGHT-ADJUSTED DOSE (MG/KG)
;
       IF (NEWIND.NE.2) THEN
          AMT=DOSE*WT
          W=WT
       ENDIF
       T0=THETA(1)*EXP(ETA(1))
       T2=THETA(2)*EXP(ETA(2))
       T1=T2+T0
       T3=THETA(3)*W*EXP(ETA(3))
       A=EXP(-T2*TIME)
       B=EXP(-T1*TIME)
       C=T1-T2
       D=A-B
       E=T3*C
       Y=AMT*T1*T2/E*D+EPS(2)

$PROB POPULATION DATA WITH PRIOR ON THETA AND OMEGA
$INPUT ID DOSE TIME DV WT
$DATA data REWIND
$THETA  (0,4,5) (0,.09,.5) (.004,.01,.9)
$OMEGA BLOCK (3) .7 .04 .05 .02 .06 .08
$SIGMA  .4 FIX .5

$SIM (547676) ONLY TRUE=PRIOR
```

Instead of a $PRIOR record, the following may be used:

```
$SUBROUTINE PRIOR=prior
```

The prior routine is as follows:

```
       SUBROUTINE PRIOR (ICALL,CNT)
       USE SIZES,     ONLY: DPSIZE,ISIZE
       USE NMPRD_INT, ONLY: IPROB
       REAL(KIND=DPSIZE) :: CNT
       REAL(KIND=DPSIZE) :: PLEV
```

```
      INTEGER(KIND=ISIZE) :: ICALL
      IF (IPROB==2) THEN
      PLEV=.9999
      IVAR=1
      ITYP=0
      NSAM=0
      ISS=0
      IFND=0
      MODE=0
      CALL TNPRI(IFND,MODE,ITYP,PLEV,NSAM,ISS,IVAR,CNT)
      ENDIF
      RETURN
      END
```

REFERENCES: none.

TRANSIT COMPARTMENT EXAMPLES

A number of examples are provided for the implementation of dose superimposition into a transit compartment.

BACKGROUND:

These examples are based on the following two papers:

Implementation of a transit compartment model for describing drug absorption in pharmacokinetic studies
J Pharmacokinet Pharmacodyn (2007) 34:711-726 DOI 10.1007/s10928-007-9066-0
Radojka M. Savic, Daniel M. Jonker, Thomas Kerbusch, Mats. O. Karlsson

Implementation of dose superimposition to introduce multiple doses for a mathematical absorption model (transit compartment model)
J Pharmacokinet Pharmacodyn (2012) 39:251-262 DOI 10.1007/s10928-012-9247
Jun Shen, Alison Boeckmann & Andrew Vick

The first paper defines what a transit compartment is, and gives a model for a single dose into such a compartment. The second paper gives several ways that multiple dosing with dose superimposition can be modelled with NONMEM 7.

Files on ftp site and NONMEM 7

The files for the second paper are provided on the IDS ftp site:

ftp://nonmem.iconplc.com/Public/nonmem/transit_compartment/

The same files are also provided with NONMEM 7 in directory

examples\transit_compartment

The general solution (Appendix 4; sumdosetf.ctl) is of particular interest in that it gives an example of the use of vectors (e.g., `VECTRA(n)`, `VECTRB(n)`) and user-written functions (`FUNCA`, `FUNCB`) in abbreviated code.

NEW FILES FOR NONMEM 7:

The following files may be found in the examples directory.

sumdosetn.ctl, sumdosetf.dat

With NONMEM 7, it is also possible to code the general solution directly in abbreviated code, using the DO WHILE statement, user-declared arrays, and subscripted variables.

Note that data file sumdosetf.csv is identical to data file sall6.csv on the ftp site.

As noted in the INTRODUCTION TO NONMEM 7, this includes abbreviated code for summation such as:

```
$abbr declare dosetime(100),dose(100)
$abbr declare dowhile i

$DES
INPT=0
I=1
DOWHILE (I<=NDOSE)
IPT=0
IF (T>=dosetime(I)) &
   IPT=DOSE(I)*(T-dosetime(I))**NN*EXP(-KTR*(T-dosetime(I)))
INPT=INPT+IPT
```

```
I=I+1
ENDDO
```

ssmultidose.ctl, ssmultidose.dat

This is an example of a steady state reached using multiple transient bolus doses. The abbreviated code is the same as sumdosetn.ctl, but the data file ssmultidose.dat has nine identical bolus doses, at 12 hour intervals, leading to steady state (to the number of significant digits printed).

ssonedose.ctl, ssonedose.dat

This is an example of a steady state computed with a single dose record having SS=1 and II=12. The data file is ssonedose.dat. The $PK block saves the values of AMT and II from the dose record. Abbreviated code in the $DES block contains a DO WHILE loop that adds drug from each implied dose, going back in time till the amount from the implied dose is negligible.

REFERENCES: Guide Introduction_7

USER-DEFINED RESERVED FUNCTION EXAMPLE

If you wish to define your own function, and have the information about its proper use of arguments be conveyed upon its execution, so the compiler may detect errors, then one method is to package the definition of the function in a USE module. The function must be listed in an include file whose name starts with the characters
`nonmem_reserved`

Caution: NMTRAN will permit the use of such a function but will not compute eta deriatives with respect to the arguments. Do not use such functions to compute any variable that affects the objective function!

Here is an example.

Myfuncmodule.f90 defines the functions mymin and mymax:

```
MODULE MYFUNCS
contains
function mymin(a,b,c,d,e)
integer mymin
integer a,b,c,d,e
mymin=min(a,b,c,d,e)
end function
function mymax(a,b,c,d,e)
integer mymax
integer a,b,c,d,e
mymax=max(a,b,c,d,e)
end function
END MODULE MYFUNCS
```

Nonmem_reserved_myfunc is the include file that declares its use:

```
" USE myfuncs, only: mymin,mymax
```

The following control stream file uses the function:

```
$PROB  THEOPHYLLINE   POPULATION DATA
$INPUT      ID DOSE=AMT TIME CP=DV WT
$DATA       THEOPP

$SUBROUTINES  ADVAN2 OTHER=myfuncmodule

$PK
;THETA(1)=MEAN ABSORPTION RATE CONSTANT (1/HR)
;THETA(2)=MEAN ELIMINATION RATE CONSTANT (1/HR)
;THETA(3)=SLOPE OF CLEARANCE VS WEIGHT RELATIONSHIP (LITERS/HR/KG)
;SCALING PARAMETER=VOLUME/WT SINCE DOSE IS WEIGHT-ADJUSTED
include "nonmem_reserved_myfunc"
   CALLFL=1
   KA=THETA(1)+ETA(1)
   K=THETA(2)+ETA(2)
   CL=THETA(3)*WT+ETA(3)
   SC=CL/K/WT
I=mymin(1,2,3,4,5.0)
print *,'I ',I
```

```
$THETA  (.1,3,5) (.008,.08,.5) (.004,.04,.9)
$OMEGA BLOCK(3)  6 .005 .0002 .3 .006 .4

$ERROR
   Y=F+EPS(1)

$SIGMA  .4
```

If you use the wrong argument type (real instead of integer), or perhaps use the wrong number of arguments, the compiler will readily flag this.

REFERENCES: Guide Introduction_7

YLO EXAMPLE

In this example, blood concentrations below a known quantification limit (BQL) have been eliminated from the analysis. A one-sided interval with lower boundary equal to the quantification limit is defined. The conditional likelihood for an observation is "conditioned" on values of the population parameters and also on the condition that the observation is inside the one-sided interval.
(See **pr_y**, **YLO YUP**)
(See **YLO YUP Probability: PR_Y**).

Here is an NM-TRAN control stream.

```
$PROB YLO example
$INPUT ID TIME DV AMT EVID MDV
$DATA ylow.dat IGNORE=C
$SUBROUTINES ADVAN=1 TRANS=2
$PK
CL=THETA(1)*EXP(ETA(1))
V=THETA(2)*EXP(ETA(2))
K=CL/V
S1=V

$ERROR
; Limit of Quantification (LOQ) is 1
; DV's below LOQ are given MDV=1 in dataset
; PR_Y (PRB) is probability that DV is > LOQ
; PR_Y (PRB) is provided even if MDV is 1

    YLO=1
    PRB=PR_Y
    Y = F+ERR(1)

$THETA
(0,0.693); CL
(0,24); V
$OMEGA
0.09
0.09
$SIGMA
 .5
$ESTIMATION
      MAXEVAL=9999
      NSIG=4
      METHOD=COND INTERACTION LAPLACIAN
      PRINT=5
$TABLE ID TIME AMT MDV EVID PRB
  NOPRINT
  ONEHEADER
  FILE=ylo.tab
```

Here is a segment of the dataset for a subject with last DV < LOQ

```
      ID     TIME      DV       AMT    EVID      MDV
       .        .       .         .       .        .
       .        .       .         .       .        .
       .        .       .         .       .        .
       2        0       0       100       1        1
       2    0.333    2.47         0       0        0
       2        2    3.26         0       0        0
       2        6    2.32         0       0        0
       2   15.917    2.46         0       0        0
       2       24       0       100       1        1
       2       48       0       100       1        1
       2       72       0       100       1        1
       2   95.917    4.04         0       0        0
       2       96       0       100       1        1
       2    96.08    8.85         0       0        0
       2      120    3.16         0       0        0
C Last observation record (below) has DV < LOQ (LOQ = 1) and MDV = 1
C This observation does not contribute to the -2LL estimate. However,
C probability that DV is > LOQ can be tabled using PRB=PR_Y.
       2      168   0.163         0       0        1
```

REFERENCES: None.