LINMOD LANGUAGE REFERENCE MANUAL

**David H. Christiansen, James D. Hosking, Ronald W. Helms,
Keith E. Muller, and Kevin B. Hunter**

No warranty of any kind, with respect to accuracy or function, is given for LINMOD.
The code and documentation may be shared freely, but may not be sold for profit.
Copyrighted by the authors.

1. INTRODUCTION 1
1.1 Background and Documentation
1.2 Overview of Code
1.3 User Programming Requirements
1.4 Overview of IML Modules Invoked by User
1.5 Examples of the Use of LINMOD
1.6 Sequential Dependencies in the Code
1.7 References

2. FITMODEL 10
2.1 Function
2.2 Computational Method
2.3 User Inputs
2.4 System Input Matrices
2.5 Options and Defaults
2.6 User Output (Including Matrices)
2.7 System Output Matrices
2.8 Error Conditions
2.9 Examples
2.10 Application Notes
2.11 References

3. GETCORSS 15
3.1 Function
3.2 Computational Method
3.3 User Supplied Inputs
3.4 System Input Matrices
3.5 Options and Defaults
3.6 User Output
3.7 System Outputs
3.8 Errors
3.9 Examples
3.10 Application Notes

4. MAKESS 17
4.1 Function
4.2 Algorithm
4.3 User Inputs
4.4 System Input Matrices
4.5 Options and Defaults
4.6 User Output
4.7 System Output Matrices
4.8 Error Conditions

4.9 Examples
4.10 Application Notes

5. PROCSSCP 20
5.1 Function
5.2 Algorithm (Not applicable)
5.3 User Supplied Inputs
5.4 System Input Matrices (None)
5.5 Options (None)
5.6 User Oriented Output
5.7 System Output Matrices
5.8 Errors
5.9 Examples
5.10 Application Notes

6. SETOPT 22
6.1 Function
6.2 Algorithm (Not Applicable)
6.3 User Supplied Inputs
6.4 System Supplied Inputs
6.5 Options and Defaults
6.6 User Output
6.7 System Output Matrices
6.8 Error Conditions
6.9 Examples
6.10 Application Notes (All LINMOD Options)

7. TESTGLH Module 24
7.1 Purpose
7.2 Algorithm
7.3 User Supplied Matrices
7.4 System Input Matrices
7.5 Options and Defaults
7.6 User Output (Including Matrices)
7.7 System Outputs
7.8 Errors
7.9 Examples
7.10 Application Notes
7.11 References

8. UTILITY Modules (Create Contrast Matrices and Names) 29
8.1 NAMELIST
8.2 UMEAN
8.3 UPOLY1
8.4 UPOLY2
8.5 UTRENDS

APPENDIX A: Notation and Formulae 31

APPENDIX B: Installation Guide 40

1. INTRODUCTION

1.1 Background and Documentation

LINMOD (LINear MODels) performs a wide variety of General Linear Multivariate Model (GLMM) computations in SAS IML (c). LINMOD allows the analyst familiar with matrix algebra notation and IML syntax to efficiently compute tests, estimates, and all associated statistics, for any GLMM. Searle (1971) and Timm (1975) provided thorough treatments of univariate and multivariate versions, respectively, of the models of interest. Muller, LaVange, Ramey, and Ramey (Sections 2.1-2.3, 1992) presented a succinct statement of the model, the general linear hypothesis, and associated statistics.

All of the results can be computed with some combination of PROC GLM and REG in SAS (c) programs. The primary advantage of LINMOD lies in the efficiency which results from the direct relationship between the syntax of the program and the matrix algebra formulation of models and tests. The primary disadvantage of LINMOD lies in the statistical sophistication required to use the program to produce valid results. However, anyone who has spent time trying to deduce exactly how a test or estimate was computed by a particular option in a particular program will welcome the ability to explicitly control the calculations in a formula-based syntax. The additional overhead of LINMOD will usually only be worthwhile for complex designs or tests. The less simple and traditional the design and/or hypothesis, the more likely that LINMOD will appeal to someone able to define and analyze linear models in matrix notation. A further advantage lies in the fact that all of the computed values remain available to the user in matrices. This allows storing all results in a permanent SAS database (as distinct from the LST file), and exercising complete control of formatting.

Comparing LINMOD to PROC REG (or GLM) demonstrates a classic trade-off in program design: increased flexibility and control for the sophisticated user versus friendly interface for the naive user. For example, LINMOD does not add or in any way recognize an intercept term in any model. The user must code one if desired, and may choose to test it, if present. Using LINMOD requires the ability to define and analyze linear models in matrix notation. The user should be familiar with the material presented in such texts as Timm (1975).

Starting from the assumption of the user having matrix knowledge, great effort was expended to give LINMOD an interface with consistent design, and extensive error checking, and informative messages. The wide assortment of matrix operators and functions in PROC IML greatly enhances power and flexibility. Furthermore, the extensive editing, printing, plotting and data management facilities of SAS (c) are available for pre- and post-processing.

Version 3 of LINMOD constitutes at least the fifth generation of such software. In the 1970's, a FORTRAN program (MGLM) was used for multivariate analysis by UNC-CH Biostatistics faculty and students. The second generation was written in PROC MATRIX to take advantage of the functions in the language. The third generation (LINMOD Version 1, in PROC MATRIX) provided many more functions, as well as a better user interface and error checking. Version 2 was a modest revision allowing easier support and greater portability. The creation of PROC REG (and the addition of the REPEATED statement to REG and GLM) reduced the difficulty in producing analyses for which LINMOD was designed. The introduction of IML (and the demise of PROC MATRIX) led, somewhat belatedly, to LINMOD Version 3 in IML.

Version 3 differs from Version 2 in a number of ways. 1) The user interface has changed little except to accommodate the MATRIX/IML conversion. 2) Some modules (CANCORR, PRINCOMP) were deleted because SAS procedures now provide better tools. 3) The ability to store

partially fitted models was eliminated due to limited use history and the modest cost of refitting a model. 4) The ability to reduce the raw data file (with one record per observation) to a permanent file with only one record per variable remains. The reduced file provides the sufficient statistics to allow fitting any GLMM involving the variables included. This approach allows extremely efficient analysis of large set of data. 5) Both uncorrected and Geisser-Greenhouse versions of the "univariate" approach to repeated measures tests are now available. 6) The syntax of IML allows more robust and efficient program design.

In this document notation for matrices and equations is somewhat nonstandard. This was done to facilitate distribution of the documentation for LINMOD. See Appendix A for a summary of the formulae behind the computations.

1.2 Overview of Code

The LINMOD source code directory contains 19 files, in five categories.

1. The file LINMOD.IML contains a block of IML code which causes all of the IML modules in LINMOD to be defined and made available to the user.

2. Each of a collection of 15 files contains a single module internal to LINMOD:
DEFOPT.IML, EREXIT.IML, FITMOD.IML, GETCSS.IML, INITL.IML,
MAKESS.IML, NMLIST.IML, PRPARM.IML, PVTYPE.IML, SCAN.IML,
SETOPT.IML, TSTGLH.IML, UNIREP.IML, UNSCAN.IML, and ZPIVOT.IML.
Module names are file names with an underscore prefixed and post-fixed (*e. g.*, _DEFOPT_).

3. MACROLIB.MAC contains SAS macro statements to simplify invoking LINMOD.

4. ZUSER.IML contains five IML modules invoked by the user to fit and test models:
FITMODEL, GETCORSS, MAKESS, SETOPT, TESTGLH. Each is a dummy module which invokes a corresponding (internal) module. For example, FITMODEL invokes _FITMOD_, contained in the file FITMOD.IML. This approach means that the user specifies all parameters to LINMOD by specifying matrices with particular names. This simplifies the task and improves the robustness of the code.

5. ZUTIL.IML contains six modules to help generate contrast matrices and labels:
NAMELIST, UMEAN, UPOLY1, UPOLY2, and UTRENDS. These modules are not essential to LINMOD.

If this documentation is inadequate, the user may examine the LINMOD source code, provided as ASCII text files. As always, work with a copy and not the original to avoid the risk of corrupting the code.

1.3 User Programming Requirements

A number of design features of LINMOD help protect the LINMOD code from the user's code, and vice versa. All internal (to LINMOD) matrix names, macro variables, and module names begin and end with an underscore (the module _FITMOD_, the matrix _SS_, etc). Only thirteen (external) matrices provide all user input.

Input External Matrices	**Module Using Input**
INDVARS, DEPVARS	FITMODEL
INCLUDE, EXCLUDE	GETCORSS
Z, ZNAMES	MAKESS
OPT_ON, OPT_OFF	SETOPT
C, U, THETA0, THETARNM, THETACNM	TESTGLH

Also note that GETCORSS requires that the file _CORRDS_ exist (in the IML default library, usually WORK).

The programming rules for using LINMOD may be summarized as follows.

1. Avoid using any name that both begins and ends with an underscore.
2. Avoid defining a module with any of the following names:
FITMODEL, GETCORSS, MAKESS, SETOPT, TESTGLH,
NAMELIST, UMEAN, UPOLY1, UPOLY2, or UTRENDS.
3. Use the following matrices only as LINMOD input: INDVARS, DEPVARS,
INCLUDE, EXCLUDE, Z, ZNAMES, OPT_ON, OPT_OFF, C, U,
THETA0, THETARNM, and THETACNM.

1.4 Overview of IML Modules Invoked by User (Defined in ZUSER.IML)

RUN FITMODEL; computes estimates of the parameters of a general linear multivariate model. It requires the LINMOD system matrices as input. FITMODEL does not use the raw data. Instead, it bases calculations on the SS matrix which allows it (and LINMOD) to deal with very large files. Any analysis requires the use of FITMODEL at least once.

RUN GETCORSS; retrieves and expands a SAS file created by &PROCSSCP into the matrices of information needed by LINMOD to fit and test models.

RUN MAKESS; creates a sums of squares and cross products matrix (_SS_) and certain other system matrices. These matrices contain all information necessary to perform linear models computations. MAKESS takes as input the raw data matrix, Z. It is used when the Z and Z'Z matrices are small enough to fit comfortably in memory.

Alternately, invoking &PROCSSCP provides an easy to use, extremely time efficient method for the first computing steps of LINMOD (this replaces the use of MAKESS). &PROCSSCP takes a SAS file as input (with observations as records) and reduces it to a file with variables as records. &PROCSSCP is especially recommended for large numbers of observations. &PROCSSCP creates a SAS file of TYPE=CORR which is input to the GETCORSS module.

RUN SETOPT; is used to set the options for LINMOD modules. Besides simply turning particular options on and off, the module also provides a convenient way to reset all options to default values. Also if desired, SETOPT will give the user a list of all valid options for all modules, and/or a list of output matrices, along with row and column label matrices, available upon completion of LINMOD.

RUN TESTGLH; provides tests of the general linear hypothesis and estimation of the contrasts (secondary parameters). The secondary parameters are linear combinations of the primary parameter matrix. An important feature of LINMOD, provided in TESTGLH, is easy to tests and estimation for contrasts involving linear combinations of response variables.

1.5 Examples of the Use of LINMOD

Although this manual is not a tutorial document, examples of LINMOD analyses demonstrate the ways in which the LINMOD modules can be combined. The examples, while representative of many of the more common sequences of analyses which will be performed, are by no means exhaustive in demonstrating the ways in which the LINMOD modules can be combined. Section 1.6 contains a more complete discussion. Also, see Appendix B for dicussion of using stored code (which runs

faster). Substituting STORED for SOURCE in %LET LMDIRECT= ; would suffice for the directory arrangement used for the examples.

```
TITLE1 "EXAMPLE1.SAS--Demonstrate simple LINMOD use";

* Data are from Danford, Hughes, and McNee,                *;
* "On the Analysis of Repeated Measurement Experiments," *;
* Biometrics, 16, 1960, pp. 547-565.                      *;

* 0. Define raw data file;
FILENAME IN01 "..\EXAMPLES\PAYNE.DAT";
DATA PAYNE;
     INFILE IN01;
     INPUT @7 GROUP $ INIT SCORE2 SCORE4 SCORE6 SCORE8 SCORE10;
* Create indicator variables ;
CONTROL = (GROUP="1");
LOW     = (GROUP="2");
MODERATE= (GROUP="3");
HIGH    = (GROUP="4");

PROC PRINT DATA=PAYNE;

* 1. Define directory in which LINMOD source code has been stored;
%LET LMDIRECT = ..\SOURCE\    ;

* 2. Define SAS macro code needed;
%INCLUDE "&LMDIRECT.MACROLIB.MAC" / NOSOURCE2 ;

* 3. Reduce raw data to a TYPE=CORR file named _CORRDS_
     with characteristics required by LINMOD;
&PROCSSCP DATA=PAYNE ;
          VAR CONTROL LOW MODERATE HIGH INIT
          SCORE2 SCORE4 SCORE6 SCORE8 SCORE10;

* 4.1 Start IML with at least minimum space needed;
*     Use SHOW SPACE statement and larger WORKSIZE, SYMSIZE values
      to tune performance with bigger problems;
PROC IML WORKSIZE=1000 SYMSIZE=1000;

* 4.2 Initialize LINMOD IML code;
&LINMOD ;

* 5. Retrieve the file _CORRDS_ created in Step 3 ;
RUN GETCORSS;

* 6. Define the model and estimate primary parameters;
INDVARS = { "CONTROL" "LOW" "MODERATE" "HIGH" "INIT" };
DEPVARS=NAMELIST("SCORE",2,10,2);
*Last statement creates following matrix--
*DEPVARS = { "SCORE2" "SCORE4" "SCORE6" "SCORE8" "SCORE10" };
RUN FITMODEL;
```

```
* 7.1 Conduct a test (and estimation) step;
NOTE={"* MANOVA Test of Main Effect of Treatment,            *"
    , "* comparing each treatment group to the control group *" } ;
PRINT ,NOTE ;
C = { 1 -1  0  0   0 ,
      1  0 -1  0   0 ,
      1  0  0 -1   0 } ;
THETARNM= { "C-LOW"  "C-MOD"  "C-HIGH" };
*U defaults to Identity matrix, if NROW(U)=0;
RUN TESTGLH;

* 7.2 Conduct another test (and estimation) step;
NOTE={"* Trends by Treatment Interaction,                    *"
    , "* comparing each treatment group to the control group *" } ;
PRINT , NOTE ;
*Re-use the same C matrix;
*Note that user inputs (C, U, THETA0, THETARNM, THETACNM, etc.)
 are never changed by LINMOD, and hence remain in existence;
*May FREE THETACNM, for example, to avoid conflict
 between successive invocations of TSTGLH;
TIMES ={2 4 6 8 10};
RUN UPOLY1(TIMES  ,"Time",
           USCORE , SCORENM );
U = USCORE;
THETACNM = SCORENM ;
RUN TESTGLH;
```

The example demonstrates using LINMOD to fit a multivariate general linear model, then estimate and test secondary parameters. Both the program and data are included in the EXAMPLES directory provided with LINMOD. The data being analyzed are the "Payne data" described by Danford, Hughes, and McNee (1960). Data were collected on 45 individuals suffering from cancerous lesions. Of these subjects, 39 were assigned to one of 3 treatment groups and exposed to different amounts of whole-body X-irradiation, while 6 were used as controls (no irradiation). The number of subjects in each group, and the amounts of radiation used were:

Group	Radiation Amount	# Subjects
CONTROL	0	6
LOW	25r-50r	14
MODERATE	75r-100r	15
HIGH	125r-200r	10

After an initial training period the psychomotor test was administered to each subject four times a day. The average of the four trials for each subject each day was taken as the basic score for each day. Immediately following the training, each subject was given four pretreatment tests on the device, to obtain a baseline score, and then the indicated dose of radiation was administered. Observations were made on each of 10 consecutive days following irradiation; for the controls, observations were taken for 10 consecutive days after completion of initial training and baseline measurements. The

purpose of the experiment was to ascertain the extent to which whole-body X-irradiation affected psychomotor performance as measured by the device and whether the effect varied with dose. For this example, only the baseline measurement and measurements on even-numbered days were used.

At each time (j=2, 4, 6, 8, or 10), the model has four intercepts, one for each radiation group (g=1, 2, 3, or 4), and one covariate, the baseline measurement for a subject (i(g)=1, 2, ... N(g)):

E(Y(i(g), g, j)) = beta0(g, j) + beta1(j)*INIT(i) .

Each subject's data is assumed independent of each other subject's. Within a subject, the covariance matrix is allowed to be unstructured, but assumed the same for all subjects (homogeneous between subjects):

cov(Y(i(g), g, j) , Y(i(g), g, j')) = { sigma(j, j') } (5 x 5 for the example) .

For the example, N(1)=6, N(2)=14, N(3)=15, and N(4)=10.

In matrix form the model is: E(Y) = X*BETA .

The five columns of Y (45 x 5) are the scores on days 2, 4, 6, 8, and 10 (labeled SCORE2, SCORE4,..., SCORE10 in the SAS program). The first four columns of X (45 x 5) are indicator variables for the four groups (corresponding to the variables named CONTROL, LOW, MODERATE, HIGH in the SAS program), and the fifth column of X is the initial score (INIT). BETA (5 x 5) has the structure:

beta0(Ctl, 2)	beta0(Ctl, 4)	beta0(Ctl, 6)	beta0(Ctl, 8)	beta0(Ctl, 10)
beta0(Low, 2)	beta0(Low, 4)	beta0(Low, 6)	beta0(Low, 8)	beta0(Low, 10)
beta0(Med, 2)	beta0(Med, 4)	beta0(Med, 6)	beta0(Med, 8)	beta0(Med, 10)
beta0(Hi , 2)	beta0(Hi , 4)	beta0(Hi , 6)	beta0(Hi , 8)	beta0(Hi , 10)
beta1(2)	beta1(4)	beta1(6)	beta1(8)	beta1(10)

LINMOD stores the parameter estimates in the matrices _BETA_ and _SIGMA_ (available after RUN FITMODEL). Secondary parameters and corresponding tests of hypotheses of interest are described in the example program.

The second example demonstrates using LINMOD by reading the data into IML. This approach should only be used with small sets of data.

```
TITLE1 "EXAMPLE2.SAS--Demonstrate LINMOD using MAKESS";
* Data are from Danford, Hughes, and McNee,                *;
* "On the Analysis of Repeated Measurement Experiments," *;
* Biometrics, 16, 1960, pp. 547-565.                      *;

* 0. Define raw data file;
FILENAME IN01 "..\EXAMPLES\PAYNE.DAT";
DATA PAYNE;
     INFILE IN01;
     INPUT @7 GROUP $ INIT SCORE2 SCORE4 SCORE6 SCORE8 SCORE10;
* Create indicator variables ;
CONTROL =(GROUP="1");
LOW     =(GROUP="2");
MODERATE=(GROUP="3");
HIGH    =(GROUP="4");

PROC PRINT DATA=PAYNE;

*1. Define directory in which LINMOD source code has been stored;
%LET LMDIRECT = ..\SOURCE\    ;

*2. Define SAS macro's needed;
```

```
%INCLUDE "&LMDIRECT.MACROLIB.MAC" / NOSOURCE2 ;

* 3. Reduce raw data to a TYPE=CORR file.  See Example 1;
*    Not done in this example due to using MAKESS;

* 4.1 Start IML with at least minimum space needed;
*     Use SHOW SPACE statement and larger WORKSIZE, SYMSIZE values
*     to tune performance with bigger problems;
PROC IML WORKSIZE=1000 SYMSIZE=1000;


*4.2 Initialize LINMOD code;
&LINMOD ;

*5. Change and list options;
OPT_OFF = { "MSH" };
OPT_ON  = { "LISTINFO" "AVAILOPT" };
RUN SETOPT;

*6.1 Read raw data into IML;
*    Use this approach only for small files;
*    Use PROCSSCP macro otherwise;
USE PAYNE;
READ ALL VAR{"GROUP"} INTO GROUP;
READ ALL VAR{"INIT"} INTO INIT;
READ ALL VAR{"SCORE2" "SCORE4" "SCORE6" "SCORE8" "SCORE10"} INTO Y;
CLOSE PAYNE;

*6.2 Use IML functions to create indicator variables for design
matrix;
N=NROW(Y); * # observations in sample;
CONSTANT=J(N,1,1); *Column of 1's for intercept, etc;
CELLMEAN=DESIGN(GROUP);                          *Cell mean coding;
EFFECT  =CONSTANT||DESIGNF(GROUP);               *Effect coding;
REFERENC=CONSTANT||CELLMEAN(|*,2:NCOL(CELLMEAN)|); *Reference
coding;
*Refer to IML manual for explanation of functions;
*Use PROC PRINT and FREQ on GROUP to see interpretation of BETA;

*6.3 Assemble all predictors and responses into a single matrix,
   which must be called Z (to use MAKESS).  ZNAMES must also exist,
   with 1 row, with values labeling the columns of Z ;
Z = CELLMEAN || INIT || Y;
ZNAMES = { "CONTROL"  "LOW"  "MODERATE"  "HIGH"  "INIT" }
        || { "SCORE2" "SCORE4" "SCORE6" "SCORE8" "SCORE10" } ;
FREE GROUP INIT Y ;

*7. Create SSCP matrix and associated parameters;
RUN MAKESS;
FREE Z; *Saves space, but if helpful, use PROCSSCP macro, not
MAKESS;
```

```
*8. Fit a model;
INDVARS = { "CONTROL" "LOW" "MODERATE" "HIGH" "INIT" };
DEPVARS = { "SCORE2" "SCORE4" "SCORE6" "SCORE8" "SCORE10" };
RUN FITMODEL;

*9. Conduct any testing and estimation desired;
*C= ---- ;
*U= ---- ;
*RUN TESTGLH;
```

1.6 Sequential Dependencies in the Code

The LINMOD modules must be executed in the sequence demonstrated in the examples. For example, a secondary hypothesis cannot be tested without first fitting a model, nor can a model be fitted without first creating the proper sums of squares and cross products matrix. Consider the list of steps in the following table, with numbering consistent with the examples.

LINMOD Steps

Step	Task	&PROCSSCP (Example 1)	MAKESS (Example 2)
0	Define raw file of data	LIBNAME or DATA step	LIBNAME or DATA step
1	Define source file path	%LET LMDIRECT=	%LET LMDIRECT=
2	Define macro variables	%INCLUDE MACROLIB	%INCLUDE MACROLIB
*3	Reduce data to SS etc.	&PROCSSCP	-(not done here)-
4	Initialize IML code	PROC IML -; &LINMOD	PROC IML -; &LINMOD
5	Change or list options	SETOPT	SETOPT
*6	Read raw data into IML	-(not done here)-	USE -;READ -;CLOSE;
*7	Create SS etc. in IML	GETCORSS	MAKESS
8	Fit a GLMM	FITMODEL	FITMODEL
9	Conduct tests, estimation	TESTGLH	TESTGLH

The steps marked with an asterisk vary between the two possible paths (&PROCSSCP and MAKESS). Within this required order, the user has great flexibility. Options can be changed by invoking SETOPT at any time after Step 4. Any number of models may be fitted, after Step 3. After a model has been fitted, any number of tests and estimates can be computed with TESTGLH.

In general, moving to back to an earlier step will clear errors detected, at least in terms of matrices and results internal to LINMOD. None of the code ever creates, modifies, or FREEs any matrix whose name does not start and end with an underscore. Therefore any error in a matrix such as INDVARS or THETA0 will not be fixed. The general philosophy regarding error handling was to be conservative and avoid potentially incorrect calculations. Furthermore a significant amount of code is devoted to detecting errors and reporting sufficient information to allow the user to fix the problem in one try. Overall, the goal was to provide a quick, but graceful "death."

Example 1 uses &PROCSSCP to create a TYPE=CORR file from the raw data as the input. The output is stored as a temporary SAS data set named _CORRDS_. For greater efficiency, such as with very large files, or the desire to conduct many sets of LINMOD calculations on a single set of data, the file may be stored by saying

PROC COPY IN=_CORRDS_ OUT=permanentlib.filename;.

In turn, subsequent runs need to include (before the PROC IML step)

PROC COPY IN=permanentlib.filename OUT=_CORRDS_ .

This modest inefficiency stems from the inability of the IML READ statement to accept a matrix name as a file name (and limitations associated with using macro's in this context). Note that the VAR statement with &PROCSSCP causes only the listed variables to be used in creating the output data set, and causes them to appear in the order listed.

1.7 References

Muller, K. E., LaVange, L. M., Ramey, S. L., and Ramey, C. T. (1992). Power calculations for general linear multivariate models including repeated measures applications. Journal of the American Statistical Association, **87**, 1209-1226.

Searle, S. R. (1971). Linear Models. New York: John Wiley.

Timm, N. H. (1975). Multivariate Analysis. Monterey, California: Brooks/Cole.

2. FITMODEL

2.1 Function

FITMODEL performs the task of fitting the desired model. The model is defined by the user specifying the independent and dependent variables by name. The sweep operation (Goodnight, 1978) is then performed on the sums of square and cross products (_SS_) elements corresponding to the independent variables specified by the user. The resulting _SS_ matrix can then be partitioned into B, the model parameter estimates; (X'X)**(-) the estimated parameter covariance; and S, the estimated error covariance matrix. These matrices can be displayed in various forms and significance tests can be performed on the primary parameters as described later in the Options Section.

Subsequent links to FITMODEL to fit different models will first check the current status of the variables and sweep only those not already included in the model. Since the sweep operator is reversible, independent variables to be removed are simply "swept out" of the model by the same operator that "sweeps in" new independent variables. This method results in significant computational savings when fitting many different models, for example, in a stepwise regression procedure. Both full rank (FR) and less than full rank (LTFR) models may be fitted, with full rank being the default option. If a singularity occurs when a full rank model is desired, an error message is displayed and the structure of the linear dependency is printed.

2.2 Computational Method

The model fitting process consists of the following four steps.

2.2.1 Define the Model. The user specifies the names of dependent (Y) and independent (X) variables by defining the character row matrices DEPVARS and INDVARS respectively. The variable names must match those given to the variables when the _SS_ matrix was formed. If either DEPVARS or INDVARS is not specified by the user, FITMODEL will check the current status of all variables in _SS_. If any dependent or independent variables are already defined, they are used in the model. If not, an error message is printed and the model cannot be fitted.

2.2.2 Determine which variables need to be swept. The model defined by the user is compared with the current model defined by the status of the variables in _SS_, and a row matrix _SWEEP1_ is created to indicate which variables need to be swept. A value of 1 indicates that an independent variable is to be added to the model, while -1 indicates an independent variable is to be removed, and 0 indicates no change. Note that a change from a dependent (Y) variable to a variable not in the model (W) requires no change in the sweep status.

2.2.3 Perform the Sweep. Since the sweep operator does not prescale and involves division by the pivot (Barr, Goodnight, Sall, and Helwig, 1976), some checking must be done in order to avoid division by zero or a number computationally near to zero. A row matrix, _DELTA_= _SUMSQ_*_EPSL_, is created to provide tolerances for each variable in _SS_. _SUMSQ_ is a row matrix of the raw sums of squares of all variables in _SS_, and _EPSL_ = 10**(-12) unless otherwise specified by the user. (See application note 2.10.1).

The diagonal elements (pivots) of all variables to be swept are ranked, and the largest is chosen to be swept first. Before the sweep is performed, two computational accuracy checks must be passed. First the pivot must be greater than the proper element of _DELTA_. Second, the resulting sweep must not increase the approximate condition number more than a specified amount. If the pivot fails either of these checks, it is set to zero and the model is assumed to be LTFR. The row and column elements of a zero pivot are also set to zero if they correspond to an unswept variable, or if they are within +/- _DELTA_ of zero for a swept variable.

After the selected pivot is swept, the matrix values indicating variable status, degrees of freedom and variables left to be swept are all updated. The largest remaining pivot is selected and the process is repeated until desired variables are either swept or have zero pivots. If a zero pivot exists in the swept (X) variables, a note is printed and the conditions of the LTFR option is checked. If a LTFR model is not allowed, an appropriate error condition is set. A zero pivot in the dependent (Y) variables causes a note and sets the error code to a warning level.

2.2.4 Extract Parameters and Perform Tests. The successfully swept _SS_ can now be partitioned into _BETA_, _XPXINV_ and _SIGMA_, which can later be printed and tested as described in the Options Section. It should be noted that the _XPXINV_ printed in the LTFR case is not identical to the corresponding submatrix of _SS_. The (X'X)**(-) generalized inverse must allow non-zero elements in the row and column of a zero pivot in order to preserve the reversibility of the sweeping operation. The inverse printed and used in testing, _XPXINV_ has these rows and columns all set to zero. It has the desirable property of satisfying the equation _BETA_=_XPXINV_*X'Y.

2.3 User Inputs

2.3.1 Required

2.3.1.1 DEPVARS. A character matrix of one row of dependent (Y) variable names, which are character values. This vector is required when fitting a new model. If a previous model has been fitted and DEPVARS is omitted, all previously defined dependent variables will be included in the model. The order of the variable names in DEPVARS does not change the order of the variables in _SS_. Thus, when specifying a U matrix for TESTGLH, use the variable order reflected in _SS_, not the order in DEPVARS.

2.3.1.2 INDVARS. A character matrix with one row of independent (X) variable names. This vector is required when fitting a new model. If a previous model has been fitted and INDVARS is omitted, all previously defined independent variables will be included in the model. The order of the variable names in INDVARS does not change the order of the variables in _SS_. Thus, when specifying a C matrix for TESTGLH, use the variable order reflected in _SS_, not the order in DEPVARS.

2.3.2 Optional

2.3.2.1 _EPSL_. The constant used in calculating the tolerance value for each variable. The tolerance is the raw sum of squares times _EPSL_. Default value is 10**(-12). This value is computer system dependent, and should reflect the floating point accuracy of the platform.

2.4 System Input Matrices

The following system matrices are required by this module:

OPT - the option status matrix.

ECODE - the error condition matrix.

_SS_ - the sums of squares and cross products matrix.

MODNMS - character matrix of module names.

PARM, _SUMSQ_, _VTYPE_, _VNAME_ - the _SS_ descriptors. See SETOPT and MAKESS for descriptions of these matrices.

2.5 Options and Defaults

The option names used to change the options status and the default values are shown below. See SETOPT for specific details.

2.5.1 "PARMIN" (ON) Prints the status of the variables (_VTYPE_) and parameter values (_PARM_) for input _SS_.

2.5.2 "SSIN" (OFF) Prints the input _SS_.

2.5.3 "BETA" (ON) Calculates and prints _BETA_ in the submatrix of _SS_ corresponding to the matrix of parameter estimates.

2.5.4 "XPXINV" (OFF) Extracts and prints _XPXINV_, the submatrix of _SS_ corresponding to the covariance matrix of the model.

2.5.5 "UNIBETA" (OFF) Calculates and prints model parameters, F tests and R**(2). Prints the number of observations, degrees of freedom, value of the F statistic and its p value, and the multiple correlation R**(2) for each dependent variable. WARNING! This R**(2) value includes the intercept and thus is not the traditional R**(2) value, which can be computed with TESTGLH.

2.5.6 "EXBETA" (ON) Prints the expanded form of _BETA_, a column at a time. It gives the standard error, t statistic and two-tailed p value for each parameter. These matrices are also accessible in matrix form as _BSE_, _BT_, and _BPVAL_ respectively.

2.5.7 "COVBETA" (OFF) Prints the covariance structure of _BETA_.

2.5.7 "SIGMA" (ON) Calculates and prints _SIGMA_. _SIGMA_ is the submatrix of _SS_ corresponding to the estimated error covariance matrix printed with ROWNAME=_YNAME_, COLNAME=_YNAME_.

2.5.9 "SCORR" (ON) Correlation matrix of _SIGMA_.

2.5.10 "SSSTEP" (OFF) Prints the _SS_ matrix after each sweep.

2.5.11 "SSFIT" (OFF) Prints the fitted _SS_ matrix.

2.5.l2 "LTFR" (OFF) Allows less than full rank models.

2.5.13 "LINDEP" (ON) Prints the linear dependencies among the independent (X) variables.

2.5.14 "PARMOUT" (ON) Specifies the model that has been fitted. It will give the names of the X and Y variables, the number of observations, the rank of X, and the degrees of freedom.

2.6 User Output

2.6.1 Printed Output. All printed output is described in the Options Section above.

2.6.2 User Accessible Matrices. Any of the optional matrices described in Section 2.5 can be accessed by the user directly after a link to FITMODEL. This must be done before linking any other module since some of the matrices may be changed by another module. The following table gives the matrix name, option and row and column labels where appropriate.

Matrices Available from FITMODEL

Matrix	Option	ROWNAME	COLNAME
BETA	BETA,EXBETA	_XNAME_	_YNAME_
BPVAL	EXBETA	_XNAME_	_YNAME_
BSE	EXBETA	_XNAME_	_YNAME_
_BT_	EXBETA	_XNAME_	_YNAME_
LINDEP	LINDEP	_XNAME_	_VNAME_
PARM1	UNIBETA	----	_PM1CNM_
SCORR	SCORR	_YNAME_	_YNAME_
SIGMA	SIGMA	_YNAME_	_YNAME_
_SS_	SSFIT	_VNAME_	_VNAME_
STAT	UNIBETA	_YNAME_	_STRNM_
XPXINV	XPXINV	_XNAME_	_XNAME_

2.7 System Output Matrices

The following system matrices may be modified by this module:

ECODE - the error condition matrix.

_SS_ - the sums of squares and cross products matrix.

PARM, _VTYPE_, _VNAME_ - the _SS descriptors.

See MAKESS for a description of these matrices.

2.8 Error Conditions

5001 (SEVERE) No DEPVARS specified and no dependent (Y) variables exist from a previously fitted model, or one or more variables specified in DEPVARS do not exist.

5002 (SEVERE) No INDVARS specified and no independent (X) variables exist from a previously fitted model, or one or more variables specified in INDVARS do not exist.

5003 (SEVERE) A valid _SS_ matrix has not been created.

5004 (SEVERE) A singularity in the X variables has been found and the less than full rank option (LTFR) was OFF.

5005 (SEVERE) A variable was declared in both INDVARS and DEPVARS.

5006 (SEVERE) Error in sweep status.

5007 (SEVERE) Error degrees of freedom was less than one: sample size is too small.

5010 (WARNING) A singularity exists in the Y variables.

1010 (WARNING) A singularity exists in the X variables. LTFR is allowed and has been fitted.

1110 (WARNING) A LTFR model fitted and a singularity exists in Y variables.

2.9 Examples

```
*FIT THE MODEL Y1 = (X1 X2 X3)*BETA;
INDVARS = { "X1" "X2" "X3" };
DEPVARS = { "Y1" };
RUN FITMODEL;

*OMIT X3, MODEL IS NOW Y1 = X1 + X2;
INDVARS = { "X1" "X2" }; RUN FITMODEL;
```

2.10 Application Notes

10.1 Modifying the Tolerance Limits. The tolerance for defining a zero sum of squares can be changed by the user. The current default is _EPSL_ = 10**(- 12). The user can reduce (or increase) the tolerance, thus reducing (or increasing) _EPSL_. This may be done by including, for example, the statement

EPSL=10E-16;

before linking FITMODEL. The value of _EPSL_ must reflect the number of digits of accuracy provided by SAS in the particular operating system in which LINMOD is run.

2.11 References

Goodnight, J. H. The Sweep Operator: Its Importance in Statistical Computing. Proceedings of the Computer Science and Statistics: Eleventh Annual Symposium on the INTERFACE, 1978, pp. 218-229.

Barr, A. J., Goodnight, J. H., Sall, J. P., and Helwig, J. T. A User's Guide to SAS 76, SAS Institute, Inc., Raleigh, NC, l976.

3. GETCORSS

3.1 Function

The GETCORSS module creates the basic LINMOD system matrices (_SS_, _PARM_, _VTYPE_, _SUMSQ_, _VNAME_, and _ECODE_) from the SSCP file _CORRDS_ and matrix of column names, _CORNM_. _CORRDS_ is created by the execution of &PROCSSCP. The use of a particular particular file name (expected by the GETCORSS module) is a mandatory restriction of IML due to limitations in the usefulness of macro variables inside the IML environment. If only a subset of variables in _SSCOR_ are of interest for a particular analysis, the INCLUDE or EXCLUDE statements may be used to create system matrices containing only that subset.

3.2 Algorithm (Not applicable)

3.3 User Supplied Inputs

3.3.1 Required

The user must have created the file _CORRDS_ in the default library (usually WORK) with &PROCSSCP. If the file has been stored as a permanent file with a different name by using PROC COPY, it must be copied back to the default library with the name _CORRDS_. See PROCSSCP documentation for details.

3.3.2 Optional

3.3.2.l INCLUDE. A character matrix of one row of variable names may be specified in order to create the system matrices for a subset of the variables in _SSCOR_. When it is specified, only the variables named will be present in _SS_ and its associated matrices.

3.3.2.2 EXCLUDE. A character matrix of one row of variable names may be used to specify a list of variables to be excluded from the system matrices. All of the variables in _SSCOR_ not named in EXCLUDE will be included. INCLUDE and EXCLUDE may not both be specified.

3.4 System Input Matrices

OPT, _ECODE_.

3.5 Options and Defaults

5.1 "CPARMS" (OFF) Prints the row matrix, _PARM_, containing the number of observations, number of columns of Y, rank of X, and number of columns of W.

5.2 "CSS" (OFF) Prints the _SS_ matrix created by GETCORSS.

3.6 User Output (None)

3.7 System Outputs

SSCOR contains the data as read directly from _CORRDS_.

CORNM is a character matrix of one row containing the names of the (numeric) variables in _SSCOR_.

ECODE - the error condition matrix.

_SS_ - the sums of squares and cross products matrix.

PARM, _SUMSQ_, _VNAME_, and _VTYPE_, the _SS_ descriptors.

See MAKESS for a description of these matrices.

3.8 Errors

9001 (FATAL) _SSCOR_ or _CORNM_ not defined.

9002 (FATAL) One or more variables in INCLUDE do not exist in _SS_.

9003 (FATAL) INCLUDE and EXCLUDE have both been specified.

9004 (FATAL) One or more variables in EXCLUDE do not exist in _SS_.

9005 (FATAL) Sample size not the same for all variables

3.9 Examples

This example shows using GETCORSS after &PROCSSCP, excluding two variables, A and B, from _SS_.

```
&LET LMDIRECT= ..\SOURCE ;
%INCLUDE "&LMDIRECT.MACROLIB.IML" / NOSOURCE2;
&PROCSSCP DATA=ZIN;
VAR A B X1-X3 Y1-Y2;
PROC IML WORKSIZE=1000 SYMSIZE=1000;
&LINMOD;
EXCLUDE = { "A" "B" };
RUN GETCORSS;
INDVARS = { "X1" "X2" "X3" };
DEPVARS = { "Y1" "Y2" };
RUN FITMODEL;
```

3.10 Application Notes

10.1 Excluding Swept Variables. It is permissible to specify a subset of variables (using INCLUDE or EXCLUDE) which excludes one or more independent variables which are currently in the model (have been swept by FITMODEL). This does not remove those variables from the model (in the statistical sense). Any such excluded variables are included in any further analyses. For example, if TESTGLH is then used to estimate and test hypotheses about secondary parameters, excluding some swept variables from _SS_ is statistically identical to having those variables in _SS_ and setting the columns of C which correspond to those variables to zeros for every such secondary parameter. This capability was included in LINMOD specifically to allow the user to remove the effects of a number of nuisance variables, and then exclude them from _SS_ for convenience in constructing C matrices.

4. MAKESS

4.1 Function

The MAKESS module is used to create _SS_, the uncorrected sums of squares and cross products matrix, and associated parameter values, when Z, the raw data matrix, is READ into memory in PROC IML. It will optionally check Z for missing values, and delete those rows containing one or more missing values. If Z is N x NV, then approximately (2*N*NV)+NV**(2) bytes of workspace are required for computing _SS_ when no missing value checks are done. If missing value checks are done, and NM rows are found to have missing values, then approximately [(N*NV)+(2*(N-NM)*NV)+NV**(2)] bytes of workspace will be needed. If not enough workspace is available, and for any large problem, &PROCSSCP and GETCORSS module should be used rather than MAKESS; see PROCSSCP.

4.2 Algorithm

If no missing value checks are performed, or if no missing values are found, _SS_ is computed as Z'*Z. When rows containing missing values are found, a temporary matrix _Z_ is formed containing the rows of Z without missing values and _SS_ is computed as _Z_'*_Z_. The various associated parameter values, described below, are then created. This corresponds to list-wise deletion.

The order of the variables in _SS_ is the same as the order of the variables in the SAS file that is READ. If the user wants to change this order, the KEEP option can be used on the READ statement.

4.3 User Inputs

4.3.1 Required.

4.3.1.1 Z, the data matrix containing both X's and Y's, with columns as variables.

4.3.l.2 ZNAMES, a character matrix of one row of names for the columns of Z must be defined. It must conform to Z.

4.4 System Input Matrices

The following system matrices are required by this module:

OPT - the option status matrix.

ECODE - the error condition matrix.

4.5 Options and Defaults

The option names used to change the options status and the default values are shown below. See SETOPT for specific details.

4.5.1 "CHKMISS" (ON) If this option is on, MAKESS will use only those rows with no missing values in computing _SS_. If this option is off, and missing values are present in Z , they will be treated as zeros in computing _SS_.

4.5.2 "MPARM" (OFF) This option, if enabled, causes MAKESS to print the parameters in _PARM_ described below.

4.5.3 "MSS" (OFF) When this option is enabled, MAKESS will compute and print _SS_, the sum of squares and cross products matrix. Note that this matrix is NV x NV and can require a number of pages to print.

4.6 User Output

A one line message is printed stating whether or not missing value checks were performed. If missing values were found, the number of observations deleted and the row numbers of the deleted observations are printed. The user may specify that _PARM_ and/or _SS_ be printed.

On completion of the execution of the MAKESS module, three matrices of potential interest to the user will have been created if missing values are found.

4.6.2.1 _Z_ A matrix containing those rows of Z having no missing values.

4.6.2.2 _INDX_ A row matrix containing the row numbers of the complete observations in Z.

4.6.2.3 _MINDX_ A row matrix containing the row numbers of those rows of Z containing missing values.

4.7 System Output Matrices

The MAKESS module creates or modifies six system matrices.

4.7.1 _SS_ The matrix of sums of squares and cross products.

4.7.2 _PARM_ A row matrix containing N, NCOL(Y), RANK(X), NCOL(W), and TOLR.

4.7.3 _SUMSQ_ A row matrix containing the diagonal elements (sums of squares) of _SS_. This matrix is not modified by subsequent SWEEPS in FITMODEL.

4.7.4 _VNAME_ A row matrix created from ZNAMES which contains the names of the columns (and rows) of _SS_.

4.7.5 _VTYPE_ A row matrix whose i-th element is -1 if the i-th variable is a dependent variable (Y), 0 if the i-th variable is not currently in the model (W) and 1 if the i-th element is an independent variable (X). Since independent and dependent variables are not defined until the FITMODEL module is used, at the end of MAKESS execution none of the variables are currently in the model.

4.7.6 _ECODE_ The vector (column) of system error and warning codes. An element will be set to reflect any problems encountered.

4.8 Error Conditions

7001 (FATAL) Z or ZNAMES not defined, or do not conform.

7002 (FATAL) Z (_Z_ if missing value checks performed) has fewer than two rows.

4.9 Examples

4.9.1 Fetching Z into memory. This fragment assumes that Z has already been created in a SAS file named DATAIN.

```
&LET LMDIRECT= ..\SOURCE ;
%INCLUDE "&LMDIRECT.MACROLIB.IML" / NOSOURCE2;

PROC IML WORKSIZE=1000 SYMSIZE=1000;
&LINMOD;

USE DATAIN;
READ ALL VAR _ALL_ INTO Z;
COLNAME = ZNAMES;

RUN MAKESS;
```

4.9.2 Creating Z in PROC IML. This example assumes that the dependent variables (Y1-Y5) and a CLASSES variable (GROUP) are in the SAS data set DATAIN.

```
&LET LMDIRECT= ..\SOURCE ;
%INCLUDE "&LMDIRECT.MACROLIB.IML" / NOSOURCE2;
```

```
PROC IML WORKSIZE=1000 SYMSIZE=1000;

&LINMOD;

USE DATAIN;
READ ALL VAR Y1-Y5 INTO Y COLNAME=YNAMES;
READ ALL VAR GROUP INTO GROUP;
X = DESIGN (GROUP);
XNAMES = NAMELIST("GROUP1", 1 , NCOL(X));
Z = X || Y;
ZNAMES = XNAMES || YNAMES;
FREE X Y XNAMES YNAMES GROUP;

RUN MAKESS;
```

4.10 Application Notes

4.10.1 MAKESS is significantly less efficient than &PROCSSCP in computing Z'Z. Furthermore, &PROCSSCP followed by using GETCORSS will almost always require less memory than using MAKESS. The MAKESS module is provided only for convenience with small data sets.

5. PROCSSCP

5.1 Function

Creates a SSCP matrix from Z. By invoking PROC CORR, &PROCSSCP creates a SAS file containing the uncorrected sums of squares and cross products for a list of variables in the input data set. This data set is READ into memory in PROC IML and the GETCORSS module is used to create _SS_, _PARM_, _VNAME_, _VTYPE_, _SUMSQ_, and _ECODE_ from it. At this point, other LINMOD modules can be used just as if Z, the original data matrix, had been READ into memory and the MAKESS module had been used to create _SS_, etc. &PROCSSCP has several important advantages in comparison to the MAKESS module for creating these matrices. First, &PROCSSCP reads only one observation from the input data set into memory at a time; therefore the number of observations in the input data set has no effect on the memory requirements of &PROCSSCP. In contrast, the MAKESS module processes the entire Z matrix in memory. Thus &PROCSSCP is strongly preferred, and sometimes necessary, for the analysis of large files. In addition, &PROCSSCP has other advantages in efficiency and memory in comparison with MAKESS.

5.2 Algorithm (Not applicable)

5.3 User Supplied Inputs

5.3.1 Input_data_set_name. The DATA= --- parameter on the &PROCSSCP statement specifies the name of the SAS file containing the data matrix Z. If the parameter is omitted, the most recently created file will be used, as for any SAS procedure.

5.3.2 VAR statement. The syntax follows that for any SAS procedure. The statement specifies those variables to be included in the SSCP file. If it is omitted, all numeric variables in the input file will be used. The order of the variables in the VAR statement determines the resulting order of the variables in _SS_, and hence in LINMOD. This characteristic is important in choosing contrast matrices, C, U, THETA0, and associated labels, for TESTGLH.

5.4 System Input Matrices (None)

5.5 Options (None)

5.6 User Oriented Output

Although no printed output is produced by &PROCSSCP,

PROC PRINT DATA=_CORRDS_; may be used to list the entire output file.

5.7 System Output

5.7.1 _CORRDS_ is a particular version of a TYPE=CORR file created by PROC CORR. It consists of _SS_, with three rows appended from containing the means, standard deviations, and sample sizes. Listwise deletion (elimination of the entire observation, a row of data) is invoked so that sample size is the same for all variables.

5.8 Errors

ECODE is not set by this macro; if an error occurs the SAS system error switch is set to 1 and no output data set is created.

5.9 Examples

```
* Input dependent variables (Y1-Y5) and Group code (1,2,3);
DATA Z;
        INFILE RAWDATA;
        INPUT GROUP Y1-Y5;
```

```
X1 = (GROUP=1);
X2 = (GROUP=2);
X3 = (GROUP=3);

&LET LMDIRECT= ..\SOURCE ;
%INCLUDE "&LMDIRECT.MACROLIB.IML" / NOSOURCE2;


* Create _SS_ and related information as a SAS data set;
&PROCSSCP DATA=Z ;
VAR X1-X3 Y1-Y5;
* Read the data and analyze it;
PROC IML WORKSIZE=1000 SYMSIZE=1000;
&LINMOD;
* Read the _SS_ file;
RUN GETCORSS;
* Fit a model;
INDVARS = { "X1" "X2" "X3" };
DEPVARS = { "Y1" "Y2" "Y3" "Y4" "Y5" };
RUN FITMODEL;
```

5.10 Application Notes

&PROCSSCP uses listwise deletion of observations containing one or more missing values. Using any other approach for missing data is strongly discouraged as being of unknown or poor statistical quality.

6. SETOPT

6.1 Function

The SETOPT module is used to change the values of LINMOD system options. These options are initialized to their default values when the INITIAL module is INCLUDEd. The options may be changed at anytime during the LINMOD analysis.

6.2 Algorithm (Not Applicable)

6.3 User Supplied Inputs

6.3.1 Required: none. If neither OPT_ON nor OPT_OFF is specified, the default options are restored.

6.3.2 Optional

OPT_ON A character row matrix of names of options to be turned on.

OPT_OFF A character row matrix of names of options to be turned off.

6.4 System Supplied Inputs

6.4.1 _ECODE_ The vector of LINMOD system error conditions.

6.4.2 _MNEMON_ A matrix which contains the option names.

6.4.3 _MODNMS_ Character matrix containing module names.

6.4.4 _OPT_ A binary matrix with one row per LINMOD module. If the jth element in a module's row is set to one, the corresponding option is on.

6.4.5 _PNAME_ A matrix which contains the option names.

6.5 Options and Defaults

The option name used to change an option's status and the default values are shown.

6.5.1 "AVAILOPT" (OFF) If this option is on, a printout of all available options in LINMOD will be printed in a format similar to the table in Section 6.10.

6.5.2 "LISTINFO" (OFF) If this option is on then list all of the matrices that are produced by LINMOD.

6.5.3 "CURROPTS" (OFF) If this option is on, the options currently turned on will be printed.

6.5.4 "NEWOPTS" (OFF) If this option is on, OPT_ON and OPT_OFF specified by the user will be printed.

6.5.5 "COMPRESS" (ON) If this option is on, certain titles, headings, and blank lines will be removed so as to present a more compact printout.

6.6 User Output

Nothing is printed by default. Setting options listed in 6.5 to ON prints information.

6.7 System Output Matrices

6.7.1 _OPT_ Changed to reflect the user's request.

6.7.2 _ECODE_ Will be set to reflect any errors encountered.

6.8 Error Conditions

9001 (FATAL) Bad option name specified.

6.9 Example

```
PROC IML WORKSIZE=1000 SYMSIZE=1000;
&LINMOD ;
OPT_ON = { "LTFR" "ECORR" };
OPT_OFF = { "SIGMA" };
RUN SETOPT;
```

6.10 Application Notes

LINMOD Options and Default Values

SETOPT / General	MAKESS	GETCORSS	FITMODEL	TESTGLH	
AVAILOPT	CHKMISS (ON)	CPARMS	PARMIN	C	(ON)
LISTINFO	MPARMS	CSS	SSIN	U	(ON)
CURROPTS	MSS		BETA (ON)	THETA0	(ON)
NEWOPTS			XPXINV	THETA	(ON)
COMPRESS (ON)		UNIBETA	MID		
			CHECK	EXTHETA	(ON)
			EXBETA (ON)	MATTHETA	
			COVBETA	UNITHETA	(ON)
			SIGMA (ON)	MSH	
			SCORR (ON)	MSE	
			SSSTEP	ECORR	(ON)
			SSFIT	HEIVAL	
			LTFR	CANVEC	
			LINDEP (ON)	CANRSQ	(ON)
			PARMOUT (ON)		
				MULTTEST	(ON)
				RSQUARED	(ON)
				UNIREP	(ON)
				UNIRPRNT	(ON)
				UNIRWARN	
				UNIRFORC	

Note: (ON) indicates the default for an option is ON.
The absence of (ON) indicates the default is OFF.

7. TESTGLH

7.1 Function

Estimate and test secondary parameters. The TESTGLH module computes THETA, the estimate of CBU, and performs tests of Ho: CBU=THETHA0. C, U, and THETA0 are matrices of constants provided by the user. A variety of statistics associated with THETA may also be computed and printed.

7.2 Algorithm

The estimates of the primary parameters have been formed by FITMODEL and are contained in _SS_. _THETA_ is computed as THETA = C*BETA*U-THETA0. C (a x q), U (p x b), and THETA0 (a x b) are matrices of constants, with C required and U and THETA0 optional. In most cases, THETA0 is chosen to be the zero matrix. THETA is checked for estimability, using the technique described in Helms, Hosking, and Christiansen (1979). Provided THETA is estimable, the standard errors of each element of THETA, along with t-tests, and p-values for testing each element for departure from 0 are computed. Next, if univariate F tests (for the columns of THETA) or multivariate tests (for THETA as a whole) have been requested, (C(X'X)**(-)C')**(-1) is computed using the sweep operator (Goodnight, l978). If multivariate tests have been specified, four test criteria are computed with corresponding significance tests. All are functions of the two b x b matrices H and E defined by: H = THETA'(C(X'X)**(-)C')**(-1)THETA and E = U'SU. The four statistics computed are:

1) Roy's largest root (Roy, 1939, 1945). Charts of the distribution of this statistic are available in many standard linear models texts (e.g. Morrison, l976; Timm, 1975). Note that alternate definitions of the statistics are available. LINOD reports the largest eigenvalue of H*(H+E)**-1, the largest squared canonical correlation, while SAS GLM reports the largest eigenvalue of H*(E)**-1.

2) The likelihood ratio criterion (Wilks, 1936). Whenever MIN(a, b) < 3, an exact F test is computed based on Rao (1973, p. 555). In other cases, an asymptotic chi-squared approximation (Bartlett, 1947) and an asymptotic F approximation (Rao, 1951) are computed.

3) The Hotelling-Lawley trace (Lawley, 1938; Hotelling, 1947, 1951). An asymptotic F-approximation due to Pillai (1960) is computed whenever MIN(a, b)>1.

4) Pillai's trace (Pillai, 1955). An approximate F statistic for the significance of V is computed using Pillai's (1960) approximation, whenever MIN(a, b)>1.

The formulas for these criteria and approximations are presented in Appendix A. When THETA has only one row or one column, these four criteria are equivalent and the one (exact) F-test is computed.

7.3 User Supplied Matrices

7.3.1 Required

C, a matrix defining linear combinations of the independent variables. Each row of C defines one linear combination of the independent variables (columns of X and rows of B) which will form the corresponding row of THETA. When specifying C, the order of the independent variables must be assumed to be as they were in _SS_, not as they were specified in INDVARS.

7.3.2 Optional

U, a matrix defining linear combinations of the dependent variables. U is a p x b matrix of constants. Each column of U defines one linear combination of the dependent variables (columns of Y and columns of B) which will form the corresponding column of THETA. If U is not specified, a p x p identity matrix is used for U (i.e. no linear combinations of the dependent variables are

formed). When specifying U, the order of the dependent variables must be assumed to be as they were in _SS_, not as they were specified in DEPVARS.

THETA0, a matrix of constants to be subtracted from C*B*U to form THETA. THETA0 is an a x b matrix whose (i,j) element is subtracted from the (i,j) element of C*B*U to form THETA. If THETA0 is not specified, then an a x b zero matrix is used for THETA0 (e. g., nothing is subtracted from C*B*U). When specifying THETA0, the order of the independent and dependent variables must be assumed to be as they were in _SS_, not as they were specified in INDVARS and DEPVARS.

THETARNM, a (1 x a) row matrix of row names for THETA. If it is omitted, then TESTGLH will generate default names.

THETACNM, a (1 x b) row matrix of column names for THETA. If it is omitted, then TESTGLH will generate default names.

7.4 System Input Matrices

The following system matrices are required by this module:

OPT - the option status matrix.

ECODE - the error condition matrix.

EPSL - value used to test for numeric zero.

TGLHP1 - indicator of first invocation of TESTGLH.

_SS_ - the sums of squares and cross products matrix.

PARM, _SUMSQ_, _VTYPE_, _VNAME_ - the descriptors of _SS_.

See MAKESS and SETOPT for descriptions of these matrices.

7.5 Options and Defaults

The option names used to change the options status and the default values are shown below. See SETOPT for specific details.

7.5.1 "C" (ON) If this option is on, the C matrix specified by the user will be printed.

7.5.2 "U" (ON) If this option is on, then any user-defined U matrix will be printed. If U has not been defined, then this option has no effect.

7.5.3 "THETA0" (ON) If this option is on, then any user-defined THETA0 matrix will be printed. If THETA0 has not been defined, then this option has no effect.

7.5.4 "THETA" (ON) This option causes _THETA_, the estimate of THETA, to be printed in matrix form.

7.5.5 "MID" (OFF) When this option is on, MID = C(X'X)**(-)C' is printed.

7.5.6 "EXTHETA" (ON) This option, if on, causes THETA to be printed in "expanded form". For each column of THETA, a matrix _STAT1_ is computed and printed. The first column of _STAT1_ contains the column of THETA being expanded. The second column contains the estimated standard errors of the elements of the first column. The third column contains t-statistics for testing the corresponding element for departure from zero. The fourth column contains the degrees of freedom for the t-statistics and the fifth column contains two tailed P-values for the tests.

7.5.7 "MATTHETA" (OFF) When this option is on, TESTGLH creates three matrices containing statistics concerning the elements of THETA, which the user can then print or use in further computations. Each of these matrices is the same size as THETA (axb columns). Each element of _SDTHTA_ is the estimated standard error of the corresponding element of THETA. Each element

of _TTHTA_ is the t-statistic for the corresponding element of THETA. Each element of _PVTHTA_ contains the two tailed P-value for the corresponding element of _TTHTA_.

7.5.8 "UNITHETA" (ON) If this option is on, TESTGLH computes and prints univariate F tests, with associated degrees of freedom and two tailed P-values for each column of THETA.

7.5.9 "MSH" (OFF) This option causes H/a, the mean square hypothesis and cross products matrix to be printed.

7.5.10 "MSE" (OFF) This option causes E/(N-r), the mean square error and cross products matrix to be printed. This is the residual covariance matrix, unbiased estimate, for the transformed scores, Y*U.

7.5.11 "ECORR" (ON) This option causes the error (residual) correlation matrix (based on E) to be computed and printed.

7.5.12 "HEIVAL" (OFF) This option prints the eigenvalues of HE**(-1).

7.5.13 "CANVEC" (OFF) If this option is on, the left eigenvectors of H*INV(E) which equal the left eigenvectors of H*INV(H+E) are printed. These eigenvectors, in matrix _A_, are normalized so that _A_'*E*_A_ = I. The j-th column of _A_ is the j-th eigenvector, and its elements are proportional to the j-th canonical variate raw weights for the dependent variables.

7.5.14 "CANRSQ" (ON) This option causes the eigenvalues of H*INV(H+E) to be computed and printed. These are the generalized squared canonical correlations.

7.5.15 "MULTTEST" (ON) If this option is specified, the four standard multivariate test criteria defined in Section 7.2 (this chapter) and their associated significance tests are computed and printed.

7.5.16 "RSQUARED" (ON) This option adds to the information printed by the options "EXTHETA", "UNITHETA", and "MULTTEST", and thus one or more of these three options must be on for "RSQUARED" to have any effect. For "EXTHETA", "RSQUARED" causes the simple correlation coefficient associated with each t test to be printed. For "UNITHETA", "RSQUARED" causes the multiple R**(2) associated with each univariate F test to be printed. For "MULTTEST", it causes a multivariate measure of association to be printed for each of the four multivariate test statistics.

7.5.17 "UNIREP" (ON) If this option is specified, the uncorrected and the Geisser-Greenhouse corrected test for the univariate approach to repeated measures are computed.

7.5.18 "UNIRPRNT" (ON) This option causes the results computed if UNIREP is on to be printed.

7.5.19 "UNIRWARN" (OFF) If this option is on then whenever UNIREP is on and U'*U is not proportional to an identity matrix then a warning is printed.

7.5.20 "UNIRFORC" (OFF) If this option and UNIREP are on then the tests are computed even if U'*U is not proportional to an identity matrix, or ncol(U)>nrow(U).

7.6 User Oriented Output

7.6.l Printed Output

Printed output of TESTGLH is controlled by the options described in Section 7.5.

7.6.2 User Accessible Matrices

Any of the matrices described in Section 7.5 can be accessed by the user directly after a link to TESTGLH. This must be done before linking another module since all matrices are FREEd at the beginning of the next module. The following table gives the matrix name, option used to create it, and the chapter section in which it is described.

Matrix	Option	Rowname	Colname	Section
THETA	THETA	_THRNM_	_THCNM_	7.2.2, 7.5.4
MID	MID	_THRNM_	_THRNM_	7.2.2, 7.5.5
SDTHTA	MATTHETA	_THRNM_	_THCNM_	7.5.7
TTHTA	MATTHETA	_THRNM_	_THCNM_	7.5.7
PVTHTA	MATTHETA	_THRNM_	_THCNM_	7.5.7
MSH	MSH	_THCNM_	_THCNM_	7.2.2, 7.5.9
MSE	MSE	_THCNM_	_THCNM_	7.2.2, 7.5.10
ECORR	ECORR	_THCNM_	_THCNM_	7.5.11
HEIVAL	HEIVAL	_CANNM_	_NONM_	7.5.12
CANVEC	CANVEC	_THCNM_	_CANNM_	7.5.13
CANRSQ	CANRSQ	_CANNM_	_NONM_	7.5.14
FSTATS	UNITHETA	_THCNM_	_FSTRNM_	7.5.6,7.5.17
STMAT1	MULTTEST	_STMRNM_	_STMCNM_	7.5.16,7.5.17
TPARM1	MULTTEST	_NONM_	_TPCNM1_	7.5.16
URESUL	UNIREP	_UCOLNM_	_UROWNM_	7.5.18-7.5.21

Note that if THETARNM exists then _THTRNM_=THETARNM,
and if THETACNM exists then _THTCNM_=THETACNM.

7.7 System Outputs

TESTGLH may modify _ECODE_, the error condition matrix.

7.8 Errors

5001 (SEVERE) Last invocation of FITMODEL created an error of level 5000 or more.

3002 (TEMPORARY) C is not defined.

3003 (TEMPORARY) C and B do not conform.

3004 (TEMPORARY) U is specified and does not conform to B.

3005 (TEMPORARY) THETA0 is specified and does not conform to C*B*U.

3006 (TEMPORARY) THETA is not estimable.

3007 (TEMPORARY) TESTGLH was invoked without any model having been fitted.

3008 (TEMPORARY) TESTGLH is used, but no dependent variables have been specified in _SS_.

3009 (TEMPORARY) THETACNM and THETA do not conform.

3010 (TEMPORARY) THETARNM and THETA do not conform.

3011 (TEMPORARY) Denominator degrees of freedom are less than 1 for at least one of the four multivariate tests.

2001 (WARNING) THETA is singular, therefore estimable, but no multivariate tests can be performed.

7.9 Examples

See Examples 1 and 2 in Section 1.5.

7.10 Application Notes (None)

7.11 References

Bartlett, M. S. Multivariate analysis. Journal of the Royal Statistical Society Supplement, l947, 9(B), 176-197.

Goodnight, J. H. The Sweep Operator: its Importance in Statistical Computing. Proceedings of the Computer Science and Statistics: Eleventh Annual Symposium on the INTERFACE, 1978, pp. 218-229.

Helms, R. W., Christiansen, D. H., and Hosking, J. H. The LINMOD Algorithms: Their Rhyme and Reason. The American Statistical Association, 1979, Proceedings of Statistical Computing Section.

Hotelling, H. Multivariate Quality Control, Illustrated by the Air Testing of Sample Bombsights. In C. Eisenhart et al (eds.). Selected Techniques of Statistical Analysis. McGraw-Hill, New York, 1947, 111-184.

Hotelling, H. A Generalized T Test and Measure of Multivariate Dispersion. Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley, l95l, 23-41.

Lawley, D. N. A Generalization of Fisher's Z-test. Biometrika, 1939, 30, 180-187.

Pillai, K. C. S. Some New Test Criteria in Multivariate Analysis. Annals of Mathematical Statistics, l955, 26, 117-121.

Pillai, K. C. S. Statistical Tables for Tests of Multivariate Hypotheses. Manila, University of the Philippines Statistical Center, 1960.

Rao, C. R. An Asymptotic Expansion of the Distribution of Wilks' Criterion. Bulletin of the Institute of International Statistics, 1951, XXXIII(c), 177-180.

Roy, S. N. p-Statistics or Some Generalizations in Analysis of Variance Appropriate to Multivariate Problems. Sankhya, 1939, 4, 381-396.

Roy, S. N. The Individual Sampling Distribution of the Maximum, the Minimum, and Any Intermediate One of the p-Statistics on the Null Hypothesis. Sankhya, l945, 7, 133-158.

Rao, C. R. Linear Statistical Inference and its Applications (2nd ed.). Wiley, New York, l973, 555.

Timm, N. H. Multivariate Analysis with Applications in Education and Psychology, Wadsworth, Belmont, California, l975.

Wilks, S. S. Certain Generalizations in the Analysis of Variance. Biometrika, l932, 24, 471-494.

8. UTILITY IML MODULES

This section describes a set of utility modules designed to aid the use of LINMOD, although not a part of LINMOD. Only limited error checking is done by these modules. Note that the U matrices produced by the modules can be used as C matrices (for cell mean coding schemes) simply by transposing the results.

8.1 NAMELIST (A Function)

Usage: yourlist = NAMELIST(STEM,LOW,HIGH,BY);

This function generates a row of names, STEMlow to STEMhigh, by..., a character matrix. STEM is a character string (1x1). Require 0<=LOW<=HIGH, and 1<=BY, integers (1x1). See the MAKESS example in 4.9 for an example use. Example 1 in Section 1.5, includes the following code:
DEPVARS=NAMELIST("SCORE",2,10,2);

This statement creates the following matrix:

*DEPVARS = { "SCORE2" "SCORE4" "SCORE6" "SCORE8" "SCORE10" };

8.2 UMEAN (A Function)

Usage: UAVE = UMEAN(P); This function returns J(P,1,1/P), which provides a column (vector) to use as an averaging U matrix. UAVE=UMEAN(4); yields the column [.25 .25 .25 .25]';

8.3 UPOLY1 (A Module)

This module produces a U matrix and associated names for a test of trends. The names can be used as THETACNM. Centering and scaling are used to increase numerical accuracy.

Usage: RUN UPOLY1(VALUES, NAME, U, NMOUT);

Inputs. VALUES, a set of numeric treatment levels (values), forming a matrix with one row or column.

NAME, a character string providing stem of names for trends.

Outputs. U, a matrix with columns orthonormal polynomial coefficients

(excludes zero order).

NMOUT, a character matrix with one row of names.

Example 1 (Section 1.5) uses this module.

8.4 UPOLY2 (A Module)

This module produces a U matrices and associated names for a tests of trends and interaction for a two within-subject factors. The names can be used as THETACNM. Assume Factor 1, with levels VALUES1, varies slowly,

and that Factor 2, with levels VALUES2, varies rapidly.

Centering and scaling are used to increase numerical accuracy.

Usage: RUN UPOLY2(VALUES1,NAME1, VALUES2,NAME2,
U1,NMOUT1, U2,NMOUT2, U12 ,NMOUT12);

```
Inputs  VALUES1=1st set of numeric treatment levels/values
        NAME1  = 1st character string providing stem of names;
        VALUES2=2nd set of numeric treatment levels/values
        NAME2  = 2nd character string providing stem of names;
Outputs U1     =matrix of orthonormal polynomial coefficient
                 columns for 1st factor (excludes zero order)
        NMOUT1 =list (1 row matrix) of 1st factor names (charactr);
        U2     =matrix of orthonormal polynomial coefficient
                  columns for 2nd factor (excludes zero order)
        NMOUT2 =list (1 row matrix) of 2nd factor names (charactr);
        U12    =matrix of orthonormal polynomial coefficient
                columns for interaction (excludes zero order)
        NMOUT12=list (1 row matrix) of interaction names(charactr);
```

Example 1 (Section 1.5) uses UPOLY1, which treats the special case of one factor.

8.5 UTRENDS (A Function)

Create polynomial trends matrix (columnwise), excluding the zero order. Centering and scaling is used to help numerical accuracy.

Usage: UPOLY = UTREND(VALUES);

For example, TIME= {2 4 6}; and UTIME=UTREND(TIME); yields the 3 x 2 matrix

```
[ -1/sqrt(2)    1/sqrt(6) ]
[  0           -2/sqrt(6) ]
[  1/sqrt(2)    1/sqrt(6) ]
```

APPENDIX A: LINMOD Notation

Tables A.1 - A.8 summarize the notation used throughout the LINMOD documentation and, to the extent possible, notation used in the LINMOD code as well. The matrix _SS_ is a dynamic matrix which changes throughout computations (by modules MAKESS, GETCORSS, FITMODEL). The structure of this matrix at various places in the computations is discussed elsewhere in the LINMOD documentation.

Table A.1 Notation for Data Matrices

Description	Mathematical Notation	LINMOD Primer Notation
Data matrix	Z (N x NV)	Z
Number of observations (rows of Z)	N	N
Number of variables (columns) in Z	NV	NV
Matrix of dependent variable values [subset of the columns (variables) of Z]	Y (N x p)	Y
Number of dependent variables (columns of Y)	p	p
Matrix of independent variables values [subset of columns (variables) of Z]	X (N x q)	X
Number of independent variables (columns of X)	q	q
Rank(X)	r = Rank(X)	r
Matrix of values of "other" variables which are in Z but are not in X or Y	W	W
Number of variables (columns) of W	NV-(p+q)	NV-(p+q)
Matrix of (raw) sums of squares and cross products	Z'Z = [X'X X'Y X'W] [Y'X Y'Y Y'W] [W'X W'Y W'W]	SSCP,_SS_

Table A.2 Linear Model Parameters and Assumptions

Description	Mathematical Notation	LINMOD Primer Notation
Model Equation	E(Y) = X*BETA	E(Y)=X*BETA
Matrix of primary parameters ("regression coefficients")	BETA (q x p)	BETA (q x p)
Covariance assumptions		
(a) Univariate model (p = 1)	V(Y)=sigma**(2)I(N)	V(Y)=sigma**(2)I(N)
(b) Multivariate model (p > 1)	V[Y(,j)]=SIGMA(j,j)I(N)	SIGMA(J,J)*I(N)
	Cov[Y(,j),Y(,k)] = SIGMA(j,k)I(N)	SIGMA(J,K)*I(J)
	V[Y(i,)] = SIGMA (pxp)	SIGMA
	Cov[Y(i,),Y(k,)]=0,i=k	
Secondary parameters (See Note 1)		
(a) Univariate model	C B − THETA0 (ax1)	THETA=C*BETA−THETA0
(b) Multivariate model	C B U − THETA0 (axb)	THETA=C*BETA*U−THETA0
where the matrices are		
C, independent variable "contrast" matrix	C (axq),Rank(C)= a <= q	C
U, dependent variable "contrast" matrix	U (pxb),Rank(U)= b <= p	U

Note 1. Secondary parameters from Less Than Full Rank models are assumed to be estimable and/or "testable"; LINMOD checks these assumptions.

Table A.3 Linear Model Parameter Estimates and Related Matrices

Parameter or Matrix	Mathematical Formula for Estimator, or Description	LINMOD Primer Notation
BETA	$B = (X'X)^{**}(-1)X'Y$ [FR] $(X'X)^{**}(-)X'Y$ [LTFR]	B
SIGMA (multivariate model)	$S = \frac{(Y - XB)'(Y - XB)}{N - Rank(X)}$	S
sigma**(2)	$s^{**}(2) = \frac{(Y - XB)'(Y - XB)}{N - Rank(X)}$	s**(2)
THETA	CB - THETA0 (univariate) CBU - THETA0 (multivariate)	T

Table A.4 Some Variances and Covariances of Estimators in General Linear Models

---

a. Univariate Model (B is qx1, T is ax1)

V(B) = sigma**(2)(X'X)**(-1)

V(B(i,)) = sigma**(2)[(i,i) - elt. of (X'X)**(-1)]

V(T) = sigma**(2)C(X'X)**(-1)C'

V(T(i,)) = sigma**(2)[(i,i) - elt. of C(X'X)**(-1)C']

b. Multivariate Model (B is qxp, T is axb)

V(B(i,j)) = SIGMA(j,j)*[(i,i) - elt. of (X'X)**(-1)]

V[B(i,)] = SIGMA*[(i,i) - elt. of (X'X)**(-1)]

V[B(,j)] = SIGMA(j,j)*(X'X)**(-1)

Cov[B(i,j),B(i',j')]=SIGMA(j,j')[(i,i') - elt.of (X'X)**(-1)]

V(T(i,j))=[(j,j)-elt.of U'SIGMA U][(i,i)-elt.of C(X'X)**(-1)C']

V[T(i,)] = U'SIGMA U [(i,i)-elt. of C(X'X)**(-1)C']

V[T(,j)] = [(j,j) - elt. of U'SIGMA U] C(X'X)**(-1)C'

Cov[T(i,j),T(i',j')]=[(j,j')elt.of U'SIGMA U]
[(i,i')elt.of C(X'X)**(-1)C']

Notes

1. For a Less Than Full Rank (LTFR) model, if: (a) one agrees to use a specific, deterministic generalized inverse, (X'X)**(-) for (X'X), and (b) one agrees to accept B = (X'X)**(-)X'Y as the unique least squares estimator under consideration, then V(B) is a meaningful expression and one can obtain variances and covariances of B and its elements from the expressions in the table by replacing (X'X)**(-1) by (X'X)**(-) .

2. Unbiased estimators of all quantities in the table may be obtained by replacing parameters by estimators.

3. The "usual" standard errors of estimates are obtained by taking square roots of the variance estimates described in Note 2. These standard errors are biased estimators of the corresponding parameters, the square roots of the variances shown in the table.

Table A.5 Tests of Hypotheses in the General Linear Univariate Model

Hypotheses	Test Statistics	Degrees of Freedom
1. Ho: BETA = 0 Ha: BETA ^= 0	F = B'X'XB/(rs**(2)) Note: r=Rk(X)	r, N−r
2. Ho: BETA(i,)=0 Ha: BETA(i,)^=0,or Hb: BETA(i,)>0,or Hc: BETA(i,)<0	t = B(i,){ s**(2)[(i,i)**(−)elt.of (X'X)**(−)]**(−1/2) }	N−r
3. THETA = CB − THETA0 ax1 Ho: THETA = 0 Ha: THETA ^= 0	F=T'{[C(X'X)**(−)C']**(−1)}T/(as**(2))	N−r
4. Special Case: THETA a scalar (a=1) Ho: THETA = 0 Ha: THETA ^= 0, or Hb: THETA > 0, or Hc: THETA < 0	t=T[s**(2)C(X'X)**(−)C']**(−1/2)	N−r

Note: In 3 and 4 it is assumed a = Rank(C) and THETA is estimable.

Table A.6 Tests of Hypotheses in the General Linear Multivariate Model: I. Fundamental Notation for Tests of Ho: THETA = 0

Mathematical Notation	LINMOD Primer Notation	TESTGLH Code Notation	Description
p	p	_PARM_(1,2)	# of dependent var's (cols.) in Y
q	q		# of independent var's (cols of X)
N	N	_PARM_(1,1)	# of observations (# of rows of X, Y)
r = Rank(X)	r	_PARM_(1,3)	r=Rank(X), r<=q, r<N
df = N - r	df	_DF_	Error Degrees of freedm
THETA (axb)	THETA		CBU-THETA0 is a secondary parameter
a	a	_RQ1_	# of rows of THETA
b	b	_RP1_	# of columns of THETA
Ho: THETA=0	Ho: THETA=0		General Linear Multivariate Hypothesis
H (bxb)	H	_MSH_*(N-r)	THETA'[C(X'X)**(-1)C']THETA is the SSCP matrix for hypothesis
E (bxb)	E	_MSE_*(N-r)	E=U'SU*df, the SSCP matrix for error
EVAL(HE**(-1))	EVAL(HE**(-1))	_HEIVAL_	Eigenvalues of HE**(-1)
EVEC(HE**(-1))	EVEC(HE**(-1))	_CANVEC_	Eigenvectors of HE**(-1)
EVAL(H(H+E)**-1)	EVAL(H(H+E)**-1)	_CANRSQ_	Eigenvalues of H(H+E)**(-1)
s = Min(a,b)	s	_RS1_	s=Rank(H)=Rank(HE**(-1))= max # non-zero eigenvalues
m = (a-b-1)/2	m	_RM2_	standard parameter of multivariate test stat's
n = (df-b-1)/2	n	_RN1_	standard parameter of multivariate test stat's

Table A.7 Tests of Hypotheses in the General Linear Multivariate Model: II. Special Cases which Reduce to Univariate Tests

Hypotheses	Test Statistic	Degrees of Freedom
1. Ho: BETA(,j)=0 Ha: BETA(,j)^=0	$F = \dfrac{[B(,j)]'X'X[B(,j)]}{r*s(j,j)}$	r, N−r
2. Ho: BETA(i,j)=0,vs. Ha: BETA(i,j)^=0,or Hb: BETA(i,j)>0,or Hc: BETA(i,j)<0	$t = B(i,j)\{ s(j,j) * [(i,i)\text{-elt. of } (X'X)^{**}(-)] \}^{**}(-1/2)$	N−r
3. THETA = CBU − THETA0 (axb)		
3.1 b=1 (0 has one column), a>1 Ho: THETA = 0 Ha: THETA ^= 0	$F=T'\{[C(X'X)^{**}(-)C]^{**}(-1)\}T/(aU'SU)$ where T = CBU − THETA0	a, N−r
3.2 b=1, a=1 (THETA is a scalar) Ho: THETA = 0, vs. Ha: THETA ^= 0, or Hb: THETA > 0, or Hc: THETA < 0	$t = T[U'SU\ C(X'X)^{**}(-)C']^{**}(-1/2)$ (see 3.1 for definition of T)	N−r
4. a=1, b > 1 Ho: THETA = 0 Ha: THETA ^= 0	$F=\dfrac{df+1-b}{b} * \dfrac{1 - Tr[(H+E)^{**}(-1)]}{Tr[H(H+E)^{**}(-1)]}$	b, 2(df+1−b)

Note: Case 4 is actually not a "special case which reduces to a univariate test"; however, in Case 4 all the "usual" multivariate test statistics are equivalent and reduce essentially to Tr(HE**(−1)). Moreover, the indicated F statistic has, under Ho, a central F distribution with the indicated degrees of freedom.

Table A.8 Tests of Hypotheses in the General Linear Multivariate Model: III. General Linear Multivariate Hypothesis Ho: THETA = 0 vs. Ha. THETA ^= 0

A. Fundamental Statistics:

A.1. Roy's Largest Root

RLR(s) = Max[EVAL(H(H+E)**(-1))]
= Max[EVAL(HE**(-1))] / [1 + Max(EVAL(HE**(-1)))]

See Heck's charts for many critical values.

A.2. Wilks' Likelihood Ratio

W = |H|/|H+E| with parameters s, m, n
See Note 1.

A.3. Hotelling-Lawley Trace

U(s) = Tr[HE**(-1)] with parameters s, m, n
= sum of the s non-zero eigenvalues of HE**(-1)

A.4. Pillai's Trace

V(s) = Tr[H(H+E)**(-1)]

B. Exact Distributions of W for Special Cases

B.1. a = 2, b > 1 (THETA has 2 rows)

$$F = \frac{1 - W^{**}(1/2)}{W^{**}(1/2)} * \frac{df + 1 - b}{b}$$

with 2b, 2(df+1-b) degrees of freedom

B.2. b = 2, a >= 1 (THETA has 2 columns)

$$F = \frac{1 - W^{**}(1/2)}{W^{**}(1/2)} * \frac{df - 1}{a}$$

with 2a and 2(df-1) degrees of freedom

B.3. The cases a = 1, b >= 1, and b = 1, a >= 1 reduce to univariate tests; see Table A.7.

Table A.8 (continued)

C. Approximation for the Distribution of W for the Cases a > 2, b > 2

Rao's F Approximation

Let: $Rm = df - (b - a + 1)/2$

$Rs = [(a^{**}(2)b^{**}(2) - 4) / (a^{**}(2) + b^{**}(2) - 5)]^{**}(1/2)$

$R1 = (ab - 2) / 4$

$$F = \frac{1 - W^{**}(1/Rs)}{W^{**}(1/Rs)} * \frac{Rm\ Rs - 2\ R1}{ab}$$

Under Ho, F is approximately distributed as a central F variable with ab and (Rm Rs - 2 R1) degrees of freedom.

D. Approximations to the Distributions of Trace Statistics

D.1. Hotelling-Lawley Trace

$F = (df2)*U(s) / (df1)*s,$

where $df1 = s(2m + s + 1) = ab$

$df2 = 2(sn + 1)$

Under Ho, F is approximately distributed as a central F with df1 and df2 degrees of freedom.

D.2. Pillai's Trace

$F = (df2)*V(s) / [(df1)(s - V(s))]$

where $df1 = s(2m + s + 1) = ab$

$df2 = s(2n + s + 1) = s(df + s - b)$

Under Ho, F is approximately distributed as a central F with df1 and df2 degrees of freedom.

APPENDIX B: INSTALLATION GUIDE

B.1 Overview

The code was developed in SAS 6.08 for OS/2. Experience with other IML programs on mainframe MVS and UNIX workstations leads us to believe that the code will run with any version of SAS 6.08 or later (and will not run in 6.04). All code is supplied as ASCII text. This facilitates distribution, and also helps insure compatibility across platforms. Left and right bracket symbols were replaced with (| and |) to insure compatibility with some IBM mainframe environments.

The user may wish to take advantage of the IML facility to store interpreted code. This shortens the start-up time for running LINMOD (See B.4).

B.2 Installation Steps

1. Create at least one backup copy of all LINMOD software and documentation files.

2. Copy the source code to a directory accessible to SAS jobs. The directory structure of the distribution diskette is recommended. In MVS, a partitioned file may be a very convenient approach. Note that the statement %LET LMDIRECT = --- ; must reflect the location where LINMOD source code has been stored.

B.3 Optional Steps

3.1 Optionally modify DEFOPT.IML to change the settings of default options. The file INITL.IML contains the matrix of option names. This implies which location in _OPT_ should be reset from 0 to 1 or 1 to 0. Alternately, the user may wish to add the three lines of code
OPT_ON= { } ;
OPT_OFF= { } ;
RUN SETOPT;
at the end of the file LINMOD.IML (***after*** RUN _INITL( );).

3.2 In turn, the manual may be modified to reflect the changes.

4. Optionally the line of code _EPSL_=1E-12; should be modified to reflect the number of digits of accuracy of the computer and operating system in use. This value corresponds to the smallest value distinguishable by the machine from zero. Note that this reflects the number of digits of accuracy, not the most extreme exponent. It is likely that this value is conservative by one or two digits for current PC software. The current value was chosen in the MVS environment of PROC MATRIX, with a maximum nominal 14.2 digits of accuracy (without rounding).

B.4 Storing IML Code

Storing the IML code is quite easy to do, and saves a substantial start up time on each invocation of LINMOD. Once the source version has been installed as above, the program STORE1.SAS, in the EXAMPLES directory can be used to store the program. STORE2.SAS uses the stored code to reproduce EXAMPLE1.SAS results.

The directory STORED should be created, and the two files \STORED\LINMOD.IML and \STORED\MACROLIB.IML should be copied there. These files are different from \SOURCE\LINMOD.IML and \SOURCE\MACROLIB.IML to allow changing only one statement to use stored code. Replace
%LET LMDIRECT = ..\SOURCE\ ; with
%LET LMDIRECT = ..\STORED\ ;
In order to understand the process, list LINMOD.IML and MACROLIB.IML. These directions and the two files may need to be modified to reflect local directory structures, as well as computer operating system differences.