1978

ELLPACK 77 Contributor's Guide

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
78-267

ELLPACK 77 CONTRIBUTOR'S GUIDE

John R. Rice
Purdue University

June 10, 1978
CSD-TR 267

CONTENTS

ABSTRACT

This report gives the information needed for contributors to the ELLPACK cooperative effort to develop and evaluate methods and software for elliptic partial differential equations. The overall structure is described and then detailed information is given about the modules and their interfaces. The information is intended to be sufficient to allow the reader to prepare modules for contribution to ELLPACK 77, the version of ELLPACK with rectangular geometry. The user interface is described in some detail as certain features affect the way modules should be written. The portability notes are only reminders of the principal points one has to keep in mind while writing code that is intended to be run on a variety of machines. This is an updated and specialized version of ELLPACK Contributor's Guide, CSD-208, revised September 16, 1977. Other relevant documents are the ELLPACK User's Guide, CSD-TR 226 and the ELLPACK Distribution Guide, CSD-TR 254.

1. ELLPACK STRUCTURE. The basic organization of ELLPACK is illustrated by the diagram in Figure 1.

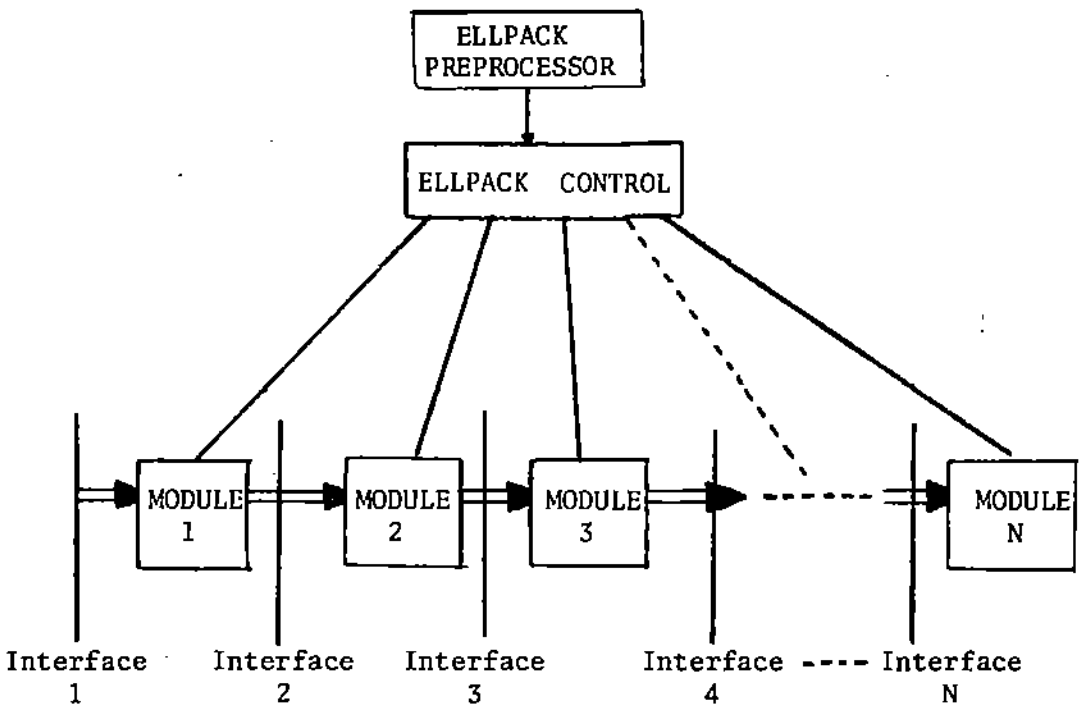

Figure 1. Basic organization of ELLPACK. The interfaces, rather than the modules, are the primary structural features of ELLPACK.

The preprocessor reads the problem and computational instructions and then translates them into the internal information for the run. The CONTROL actually invokes the requested ELLPACK modules. As suggested by Figure 1, the simplest case is straight successive execution of several modules which result in an approximate solution of the problem. Note, however, that the interfaces are the fixed points of the structure. A module may start at one interface and stop at any later one. The information at any interface is not destroyed or modified by the execution of any "later" modules. Thus once one sequence of modules has executed to establish the interface information,

then other modules may be executed starting at intermediate interfaces. For example, this allows for one ELLPACK run to form a system of equations once and then use several equation solvers on this system.

We describe three aspects of ELLPACK: the information structure, storage allocation and execution sequencing. There are four parts to the information structure of an ELLPACK run.

1) Problem Definition: This is a combination of simple variables (integer, real and logical), arrays (integer, real, etc., some of variable length) and coefficient functions (embedded in Fortran subroutines) which completely define the problem to be solved by ELLPACK, including the specification of the rectangular grid.

2) Control Information: This is a set of variables which specify various control features. For example, there is a DEBUG switch, an output level indicator and a switch for timing modules. Parameters may also be passed to modules.

3) Workspace: Space is declared which any module may use on a temporary basis. This scratch storage information is not available to subsequent modules.

4) Interface Definitions: This is a combination of simple variables and arrays (with problem dependent sizes) which are established by ELLPACK control. This information is maintained at all times during a run but its content can be modified by the output of various modules.

The allocation of storage is done by ELLPACK control in three ways, as follows

1) Standard Fixed Length Information: Many variables must be available to all ELLPACK modules and these are placed in labeled block COMMON's which every module may access. This includes various switches, the number of x,y,z grid lines, etc.

2) Variable Length Information: Arrays whose lengths are problem dependent are declared by ELLPACK control and passed to modules as arguments. All these arrays are also placed, one by one, on COMMON blocks for access by user defined subprograms. Some of these arrays are part of the problem definition and some part of the interfaces. ELLPACK control computes the sizes of these arrays for each module and thus the module contributor must specify formulas for these sizes in terms of problem definition parameters. If several modules involve the same interface then the maximum required storage is declared for each array.

3) Workspace Information: An array is placed in blank COMMON for each module to use as it pleases. Each module contributor must specify the size of the workspace in terms of the problem definition parameters and is responsible for its proper use. The amount of workspace allocated is the maximum of all the modules' requirements.

The execution of an ELLPACK run occurs in the following sequence

1) ELLPACK preprocessor:
   (a) Reads and processes the user input (ELLPACK program).
   (b) Prints the problem statement in a standard format.
   (c) Creates ELLPACK control, a Fortran main program to use modules as specified.
   (d) Creates Fortran programs for functions involved in the problem definition.
2) Fortran compiler: compiles the programs the ELLPACK preprocessor generates.
3) System loader: loads the compiled programs plus those modules needed for the run. The modules are already compiled and in the ELLPACK library.
4) Execution of the resulting job.

The programs are created by ELLPACK from user input by inserting statements and values into "template" programs which already have their structure and form determined.

The execution of a job is essentially done by executing a sequence of subprograms. This sequence is processed one item at a time and the appropriate module is executed. It is planned for ELLPACK control to check for illegal module sequencing, but the user has the final responsibility to specify a valid computational sequence.

The nature of ELLPACK is probably best understood at this point by examining the sample ELLPACK programs given in the ELLPACK Users Guide. A schematic processing of an ELLPACK run is given in Figure 2.

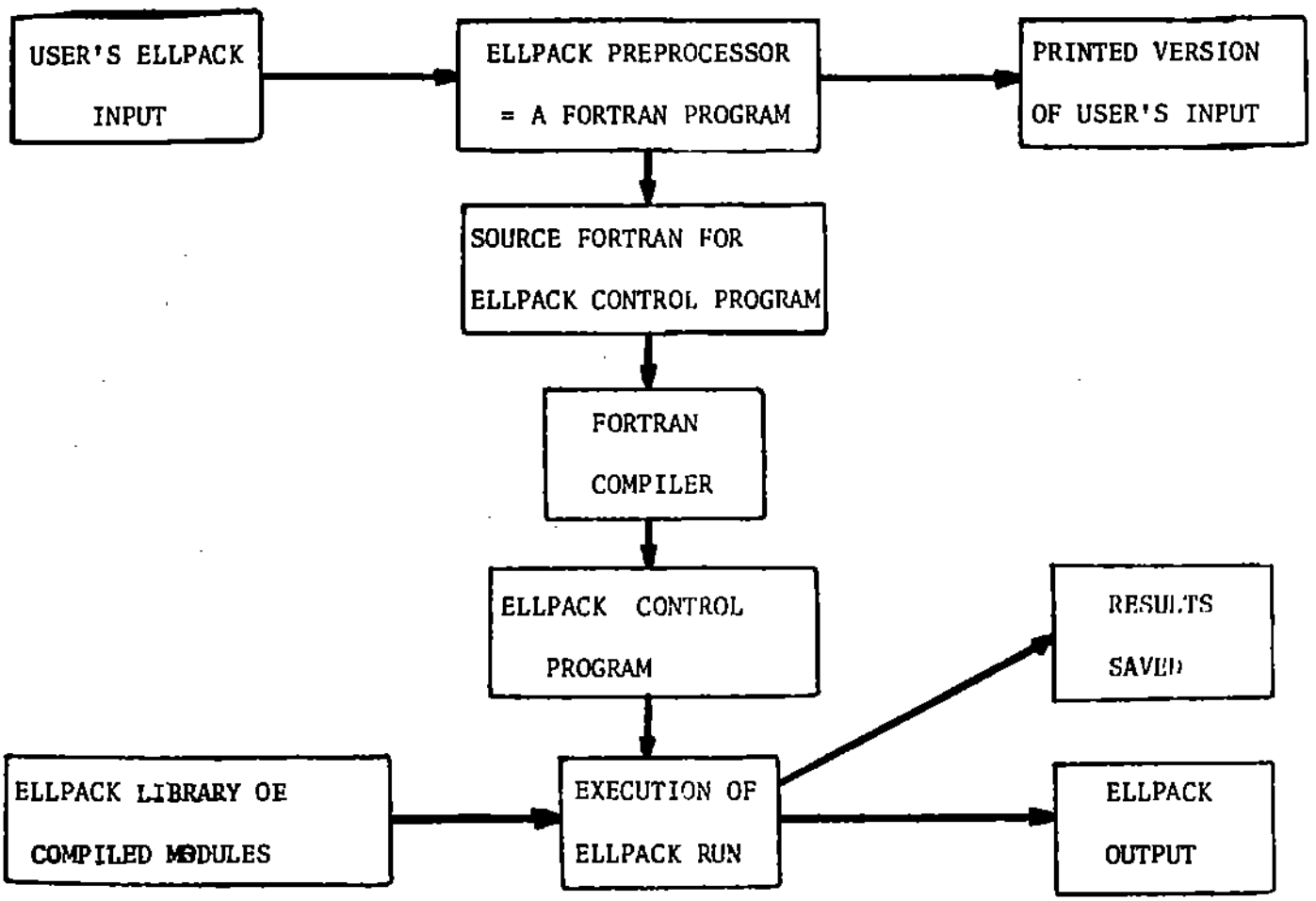

Figure 2. Schematic diagram of the processing of an ELLPACK run. This report describes how to add modules to ELLPACK 77.

2. ELLPACK 77. The principal characteristic of ELLPACK 77 is that a rectangular domain (in 2 or 3 dimensions) is assumed . The modules and interfaces for ELLPACK 77 are shown in Figure 3.

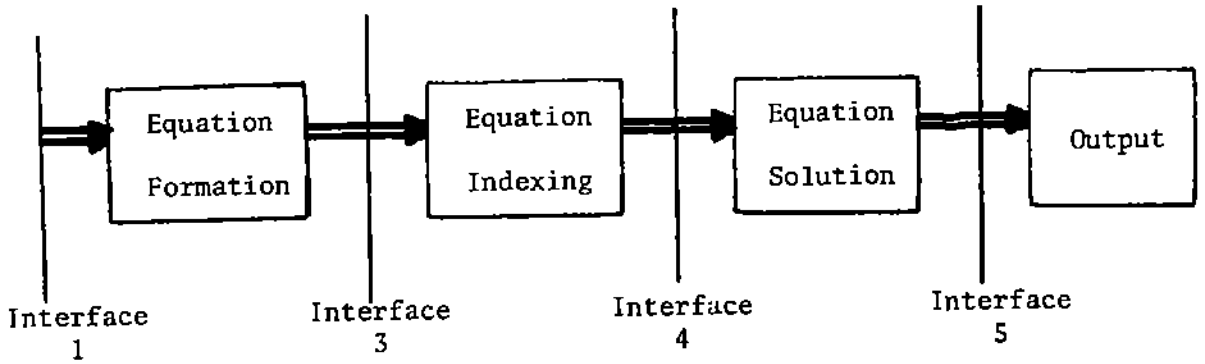

Figure 3. The basic modules and interfaces for ELLPACK 77. Interface 2 will appear in ELLPACK 78.

The items to be specified are

General and control information

Interface information for each interface

Workspace

There is a standard framework for the regions illustrated in Figures 4 and 5.

2.1 General and Control Information. This information consists of three types: Fortran functions available, variables placed in labeled COMMON blocks and arrays to be passed as arguments. This information is described primarily in terms of the Fortran code that appears at various places. The function supplied for the definition of the partial differential equation coefficients is

```
      SUBROUTINE PDE(X,Y,CVALUS)
C         TWO DIMENSIONS
C         VALUES OF EQUATION COEFFICIENTS AT (X,Y) IN THE ORDER
C         UXX, UXY, UYY, UX, UY, U, RIGHT SIDE
      REAL              CVALUS(7)
C
C
      SUBROUTINE PDE(X,Y,Z,CVALUS)
C         THREE DIMENSIONS
C         VALUES OF EQUATION COEFFICIENTS AT (X,Y,Z) IN THE ORDER
C         UXX,UXY,UYY,UXZ,UYZ,UZZ,UX,UY,UZ,U, RIGHT SIDE
      REAL              CVALUS(11)
C
      REAL                FUNCTION PDERHS(X,Y)
C           VALUE OF THE RIGHT HAND SIDE OF THE PDE
C           IN THREE DIMENSIONS THE ARGUMENTS ARE X,Y,Z
```

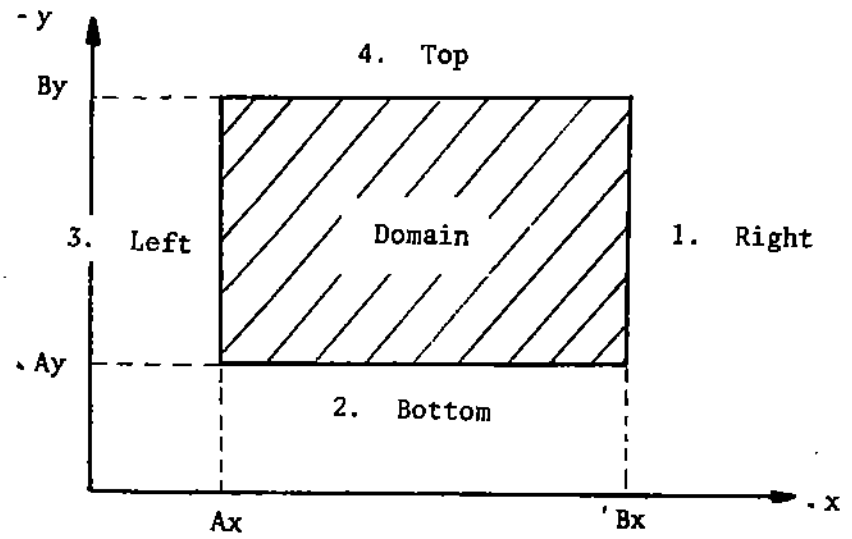

Figure 4. The standard two dimensional rectangular domain. The sides are indexed by the numbers 1 to 4 as shown.

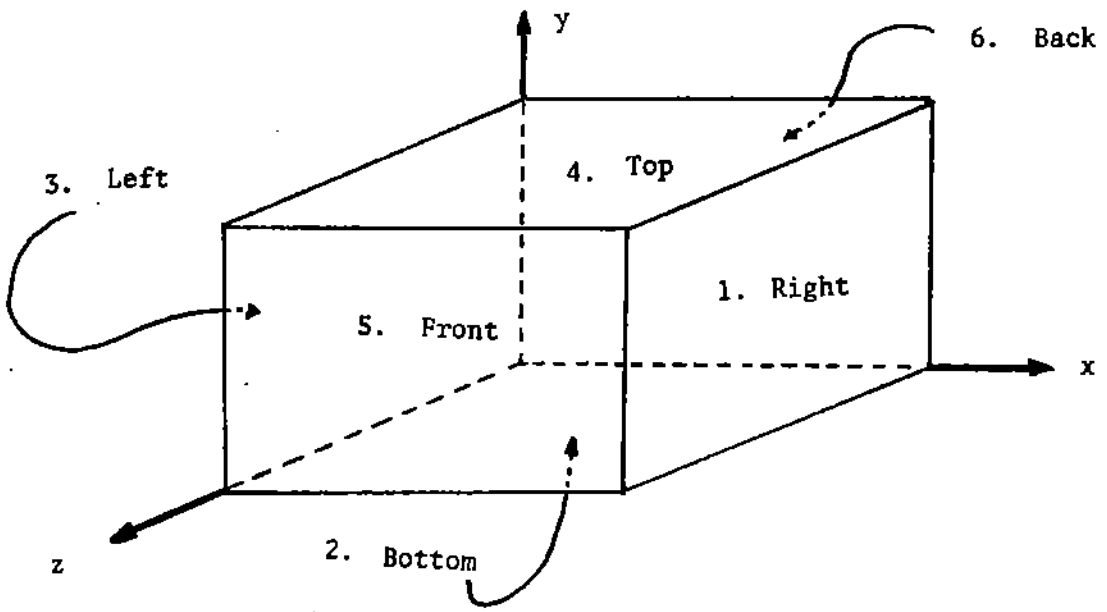

Figure 5. The standard three dimensional domain. The sides are indexed by the numbers 1 to 6 as shown and the intervals (Ax,Bx), (Ay,By) and (Az,Bz) are defined analogous to Figure 4.

For the self adjoint form of the equations the two functions p and q in $(p(x,y)u_x)_x + (q(x,yu_y)_y$ are identified with UXX and UYY in the subroutine PDE. The subroutine PDE is constructed even for the special cases of constant coefficients. However, those constants are also placed in a labeled COMMON statement and many modules will take values from there rather than from PDE. These statements are, for two and three dimensions, respectively:

```
      COMMON / CPDE / CUXX,CUXY,CUYY,CUX,CUY,CU
      REAL            CUXX,CUXY,CUYY,CUX,CUY,CU
      COMMON / CPDE / CUXX,CUXY,CUYY,CUX,CUY,CU,
     A                CUXZ,CUYZ,CUZZ,CUZ
      REAL            CUXX,CUXY,CUYY,CUX,CUY,CU,CUXZ,CUYZ,CUZZ,CUZ
```

The linear boundary conditions are defined by

```
      REAL            FUNCTION BCOND(I,X,Y,BVALUS)
C         TWO DIMENSIONS
C         VALUES OF BOUNDARY CONDITION COEFFICIENTS ON SIDE I AT (X,Y)
C         IN THE ORDER
C             U, UX, UY, RIGHT SIDE( ALSO = VALUE OF BCOND )
      REAL            BVALUS(4)
C
C
      REAL            FUNCTION BCOND(I,X,Y,Z,BVALUS)
C         THREE DIMENSIONS
C         VALUES OF BOUNDARY CONDITION COEFFICIENTS ON SIDE I AT (X,Y,Z)
C         IN THE ORDER
C             U, UX, UY, UZ, RIGHT SIDE
      REAL            BVALUS(5)
```

Providing both the side index I and the point coordinates x and y is slightly redundant. It provides for more natural programming and, more important, it is compatible with the more general domains of ELLPACK 78.

The information about the problem definition is placed in labeled COMMON according to variable type (this is done to assist portability) as shown below

```
      COMMON/ PROBL / DIM2,DIM3,POISON,LAPLAC,CONSTC,SELFAD,
     A        HOMOEQ,CROSST,HOMOBC,DIRICH,NEUMAN,MIXED,UNIFRM
      LOGICAL DIM2,DIM3,POISON,LAPLAC,CONSTC,SELFAD,CROSST,
     A        DIRICH,NEUMAN,MIXED,UNIFRM,HOMOEQ,HOMOBC
      COMMON/ PROBR / AX,BX,AY,BY,AZ,BZ,HX,HY,HZ
      REAL            HX,HY,HZ,AX,BX,AY,BY,AZ,BZ
      COMMON/ PROBI / NGRIDX,NGRIDY,NGRIDZ,BCTYPE
      INTEGER NGRIDX,NGRIDY,NGRIDZ,BCTYPE(6)
C
C         DIM2   = TRUE FOR A TWO-DIMENSIONAL PROBLEM
C         DIM3   = TRUE FOR A THREE-DIMENSIONAL PROBLEM
C         POISON = TRUE FOR THE POISSON PROBLEM
C         LAPLAC = TRUE FOR THE LAPLACE EQUATION
C         CONSTC = TRUE FOR A CONSTANT COEFFICIENTS EQUATION
C         SELFAD = TRUE FOR AN EQUATION WRITTEN IN SELF-ADJOINT FORM
C         CROSST = TRUE FOR AN EQUATION WITH A UXY TERM IN IT
C         HOMOEQ = TRUE FOR A HOMOGENEOUS EQUATION
C         DIRICH = TRUE FOR DIRICHLET BOUNDARY CONDITIONS
C         NEUMAN = TRUE FOR NEUMANN BOUNDARY CONDITIONS
C         MIXED  = TRUE FOR GENERAL MIXED BOUNDARY CONDITIONS
C         HOMOBC = TRUE FOR BOUNDARY CONDITIONS
C         UNIFRM = TRUE FOR UNIFORM GRID SPACING
C         NGRIDX,NGRIDY,NGRIDZ = NUMBER OF X,Y,Z GRID LINES INCLUDING EDGES
C         BCTYPE = INDICATORS OF BOUNDARY CONDITION TYPE FOR I=1,4 OR 1,6
C                     = 1 FOR DIRICHLET
C                     = 2 FOR NEUMANN
C                     = 3 FOR MIXED
C         HX,HY,HZ = X,Y,Z GRID SPACING FOR UNIFORM GRIDS
C         AX,BX,AY,BY,AZ,BZ  = END POINTS OF INTERVALS DEFINING DOMAIN
```

Note that the array BCTYPE is of fixed length for ELLPACK 77 but will become variable for ELLPACK 78. It indicates the type of the boundary condition on the various sides of the domain. There are two or three arrays from the problem definition which define the rectangular grid. These module arguments are:

```
C
C         ARRAYS THAT DEFINE THE RECTANGULAR GRID
C     GRIDX(I), I = 1,NGRIDX  GRIDY(I), I = 1,NGRIDY  GRIDZ(I), I=1,NGRIDZ
C         THE INPUT MODULE DECLARES THE ACTUAL DIMENSIONS, THE
C         ARRAYS SHOULD BE USED IN THE MODULES AS FOLLOWS
C
      REAL              GRIDX(NGRDXD),GRIDY(NGRDYD),GRIDZ(NGRDZD)
C
C         THE VARIABLES NGRDXD( = NGRIDX DECLARATION ),NGRDYD,NGRDZD
C         HAVE NAMES DIFFERENT FROM NGRIDX,ETC. BECAUSE THEY MUST BE
C         PASSED AS ARGUMENTS AND THEIR NAMES CANNOT CONFLICT WITH
C         THOSE IN COMMON.
C
```

This completes the definition of the problem. There are other variables of a general nature defined below which are relevant to the modules

```
      COMMON/ CONTRL/ DEBUG,TIMER,SYMMET
      COMMON/ INTEGS/ NUMBEQ,NUMCOE,NROW,NCOL,NBAND,IROT,LEVEL,
     A                INITL,INDIS,INSOL,MINPUT,MOUTPT,MEMORY,ININD,
     B                NGRDXD,NGRDYD,NGRDZD,MXNEQ,MXNCOE,NROWD,NCOLD
      COMMON/ REALS / ERRMAX,SOLMAX,TRUMAX,RESMAX,TIMES
C
      LOGICAL DEBUG,TIMER,SYMMET
      REAL            TIMES(28)
C
C         DEBUG  = TRUE FOR DEBUGGING LEVEL .GE. 3
C         LEVEL  = PRINT LEVEL CONTROL
C                       = 0   NO OUTPUT EXCEPT FATAL ERROR MESSAGES
C                       = 1   MININAL SUMMARY OUTPUT
C                       = 2   MORE INFORMATION ABOUT THE COMPUTATION
C                       = 3   DEBUGGING OUTPUT
C                       = 4   VERY DETAILED DEBUGGING OUTPUT
C         TIMER  = TRUE TO MEASURE EXECUTION TIMES
C         TIMES  = ARRAY OF MEASURED TIMES, INITIAL TIME IS THE LAST ITEM
C         INITL  = SWITCH FOR FIRST INVOCATION OF A MODULE
C                  ALLOWS FOR A SET-UP CALCULATION THE FIRST TIME A PARTICULAR
C                  SUBPROGRAM IS USED.
C         MINPUT = STANDARD INPUT  UNIT NUMBER
C         MOUTPT = STANDARD OUTPUT UNIT NUMBER
C
C     THE FOLLOWING ARE UNLIKELY TO CONCERN A CONTRIBUTOR
          IROT   = ARRAY TO KEEP DOMAIN SIDES IN RIGHT ORDER
C         INDIS  = NUMBER OF MOST RECENTLY EXECUTED DISCRETIZATION.
C         ININD  = NUMBER OF MOST RECENTLY EXECUTED INDEXING.
C         INSOL  = NUMBER OF MOST RECENTLY EXECUTED SOLUTION.
C
C    THE REMAINING VARIABLES ARE DEFINED LATER
```

2.2 Interface 1: Initial Situation. At this interface all of the general and control information of Section 2.1 is defined. Since no module executes in front of this interface, this information cannot be changed during a run.

In addition, the workspace allocation has been made as follows.

```
C
C     WORKSPACE ALLOCATION IS MADE IN ELLPACK CONTROL IN BLANK COMMON.
C         A MODULE MAY USE THIS HE PLEASES,  EXAMPLE.
C                   MAIN PROGRAM
      COMMON WORKSP(2000)
C                   MODULE
      COMMON ITABLE(5,5),XLIST(220),PIVOTS(55),REST(2)
C
C         NOTE THAT THE ARRAY REST ACTUALLY HAS 2000-(25+220+55) = 1700
C         WORDS AVAILABLE WHICH THE MODULE CONTRIBUTOR CAN USE AS HE
C         SEES FIT.  THIS MUST BE DONE VERY CAREFULLY AND COMPLICATED
C         MEMORY MANAGEMENT HERE WILL PROBABLY CAUSE ERRORS SOMEDAY
C         TAKE CARE WHEN MIXING DOUBLE PRECISION ( REAL BECOMES
C         DOUBLE PRECISION ON IBM) AND INTEGERS.
```

2.3 Interface 3: Equations Generated by Discretization. (Interface 2 appears in ELLPACK 78). The equations are generated and, if feasible, "attached" to the domain. A particular sparse matrix representation is used and an array is created to associate variables with particular grid points or squares. Thus the N^{th} equation is represented by:

COEF(N,1), COEF(N,2), COEF(N,3), ··· , COEF(N,MXNCOE-1), RSIDE

ID(N,1), ID(N,2), ID(N,3), , ID(N,MXNCOE-1), IDELEM

where

COEF(N,K) = K^{th} coefficient of N^{th} linear equation

RSIDE = right side of N^{th} linear equation

ID(N,K) = identity (column number) of the variable multiplied by COEF(N,K)

IDELEM = Pointer for N^{th} equation to its geometrical location (as described below)

MXNCOE = 1+ maximum number of coefficients in any equation.

The values in COEF and ID are left justified with zero fill except that RSIDE and IDELEM are always in the last spot. If ID(J) = 0, then that and all subsequent ID's are zero and the corresponding COEF values do not appear in the equation.

These representations are placed in two arrays

COEF (N,M)	M = 1 TO MXNCOE	Note: COEF(N,MXNCOE)=RSIDE
	N = 1 TO MXNEQ	
IDCOEF(N,M)	M = 1 TO MXNCOE	Note: IDCOEF(N,MXNCOE)=IDELEM
	N = 1 TO MXNEQ	

where MXNEQ is the maximum number of equations (that is, these are the values in the declaration statements for COEF and IDCOEF).

In many discretizations there is a natural association of the variables and equations with the grid and this association should be preserved (in a uniform manner) if present. Thus in finite differences

one has an equation and an unknown associated with each grid point. In finite elements one has several unknowns associated with a grid point and several equations associated with each grid square. We number the grids point (IX,IY,IZ) by

IGRID = IX + (NGRIDX (IY-1 + NGRID (IZ-1))

and define a mapping from the unknown indices in IDCOEF to grid numbers. Specifically, the indices in IDCDEF range from 1 to NUMBEQ (where NUMBEQ $\leq$ MXNEQ is the actual number of equations) while the grid numbers range from 1 to NGRIDX * NGRIDY * NGRIDZ and IGRID(K), K = 1 to MXNEQ, maps indices of unknowns to grid numbers by the formula above i.e. unknown K is associated with the grid point on square IGRID(K). If this association does not make sense (such as for Taylor series or harmonic polynomial expansions) then IGRID(K) = K.

The same scheme is used to define IDELEM (in IDCOEF(MXNCOE)) if the equation is naturally associated with a grid point or square.

Certain conventions are to be followed for certain common situations:

1. Association of Equations and Unknowns. If, as in finite differences, there is a natural or possible association of equations with unknowns, then the coefficient of this unknown is in COEF(N,1) and IDCOEF(N,1) = N. If this convention is not followed, this should be noted in the "Module Information" provided (see Appendix 3)
2. Symmetric Systems of equations. If it is known that the linear system is symmetric, then the upper triangular part (including diagonal) is placed in COEF first and the lower triangular part (excluding the diagonal) is copied into COEF after that. In this case the ELLPACK variable SYMMET must be set to .TRUE.

In addition to the arrays specifying the equations, there are some other variables returned via the labeled common as seen below:

```
C       VARIABLES RETURNED BY THE EQUATION FORMATION MODULE
C
C       NUMBEQ = ACTUAL NUMBER OF EQUATIONS GENERATED
C       NUMCOE = ACTUAL MAXIMUM NUMBER OF COEFFICIENTS IN AN EQUATION
C                NOT INCLUDING THE RIGHT SIDE
C       SYMMET = .TRUE. IF EQUATIONS ARE GENERATED IN THE SYMMETRIC FORMAT
```

An equation formation module could appear as shown below as far as the definition of Interface 3 is concerned, the calling sequence is not fixed however.

```
      SUBROUTINE FORMEQ(GRIDX,NGRDXD,GRIDY,NGRDYD,COEF,MXNCOE,MXNEQ,
     A                  IDCOEF)
      REAL              GRIDX(NGRDXD),GRIDY(NGRDYD),COEF(MXNCOE,MXNEQ)
      INTEGER IDCOEF(MXNCOE,MXNEQ)
```

2.4 Interface 4: Equation Indexing. The equation formation module generates the equations in whatever order the elements are processed. The equation indexing module defines another order which is appropriate for a specific linear equation solving module. Though the essence of this module is to define a new ordering, in some cases it might be natural to place the equations into another array at the same time. This should be done only if it is obviously inefficient to do otherwise. More than one indexing module might be appropriate to obtain the desired ordering.

The arrays in this interface are NDXEQ,NDXUNK, INVNDX, and, possibly, AMATRX and BVECTR. They are defined as follows

2.4 Interface 4: Equation Indexing. The equation formation module generates the equations in whatever order the elements are processed. The equation indexing module defines another order which is appropriate for a specific linear equation solving module. Though the essence of this module is to define a new ordering, in some cases it might be natural to place the equations into another array at the same time. This should be done only if it is obviously inefficient to do otherwise. More than one indexing module might be appropriate to obtain the desired ordering.

The arrays in this interface are NDXEQ, NDXUNK, INVNDX, and, possibly, AMATRX and BVECTR. They are defined as follows

```
C         THE THREE ARRAYS AT INTERFACE 3 = EQUATION INDEXING
      INTEGER NDXEQ(MXNEQ),NDXUNK(MXNEQ),INVNDX(MXNEQ)
C         THE TWO ARRAYS FOR REFORMATTING THE LINEAR SYSTEM
      REAL              AMATRX(NROWD,NCOLD),BVECTR(NROWD)
C
C         MXNEQ,NROWD AND NCOLD ARE VALUES IN DIMENSION DECLARATIONS
C         ACTUAL SIZE OF AMATRX IS SPECIFIED BY NROW, NCOL IN COMMON
C
```

The array NDXUNK(J) maps the unknown with index J (in column J) into the new unknown index. The array INVNDX inverts this map, that is

$$J = INVNDX(NDXUNK(J))$$

The array NDXEQ maps the index N of COEF(N,J) into the new equation index. Often the functions NDXEQ and NDXUNK are the same.

2.5 Interface 5: Equation Solution: The primary output of the equation solution modules is the array UNKNWN of values for the unknowns. This array is declared by ELLPACK control and two typical forms of an equation solving module as follows:

```
C                    TYPICAL SOLUTION BY A DIRECT METHOD
      SUBROUTINE EQSOL(AMATRX,BVECTR,NROWD,NCOLD,UNKNWN,MXNEQ)
      REAL             AMATRX(NROWD,NCOLD),UNKNWN(MXNEQ),BVECTR(NROWD)
C
C
C                    TYPICAL SOLUTION BY AN ITERATIVE METHOD
      SUBROUTINE SOR(COEF,MXNCOE,MXNEQ,IDCOEF,NDXEQ,INVNDX,UNKNWN)
      REAL             COEF(MXNCOE,MXNEQ),UNKNWN(MXNEQ),
      INTEGER IDCOEF(MXNCOI,MXNEQ),NDXEQ(MXNEQ),INVNDX(MXNEQ)
```

In most instances the first step of the solution module is to change the representation of the linear system. Thus a band solver will form a band matrix from the COEF and IDCOEF arrays; an iterative method might rewrite the equations so that minimal use of pointers is required during the iterations. The variables AMATRX and BVECTR are provided as the plan to put the new representation. In some cases the work of this representation change is significant and the contributor may want to put it in a separate subroutine and time its performance.

2.6 OUTPUT Module Requirements. One facility of the output module is to produce tables or plots of the approximate solution and, if the true solution is available, the error. This requires that a connection must be made between the unknowns obtained by the equation solution module and the approximations made in the equation formation module. Thus every equation formation module must have an associated function to evaluate the approximate solution. The output module uses a function provided by the equation formation module contributor, for example:

```
REAL             FUNCTION ANSOR(X,Y,UNKNWN,MXNEQ,NDXUNK)
REAL             UNKNWN(MXNEQ)
INTEGER NDXUNK(MXNEQ)
```

This function is embedded in the ELLPACK function SOLUT and a contributor of a discretization module should examine that function to see various techniques needed to evaluate the solution at a general point. If more than one equation formation module is invoked on a run, then the control module takes care that

the correct solution evaluation is used. The contributor may use any ELLPACK 77 arrays in his solution evaluation. ELLPACK provides a standard routine for bivariate and trivariate quadratic interpolation, so tabulation of the solution values on the grid is sufficient for evaluation at a general point.

The output module may use the true solution function

```
FUNCTION  TRUE(X,Y)
FUNCTION  TRUE(X,Y,Z)
```

if it is provided at the input as a Fortran function.

3. STEPS FOR MODULE INSERTION.

There are four steps to adding a module to the ELLPACK system once its arguments and common blocks are properly associated with the ELLPACK variables. These steps are:

1. Additions to Preprocessor tables
2. Additions to Preprocessor code
3. Recompile modified Preprocessor subprograms
4. Compile new modules and add to ELLPACK library

In order to allow experimentation without affecting the preprocessor, three dummy modules are included with the following ELLPACK and FORTRAN names:

TEST DISCRETE	TESTDI
TEST INDEX	TESTIN
TEST SOLUTION	TESTSO

These steps are explained in more detail below.

3.1 Preprocessor Interface. The procedures are essentially the same for discretizations, indexing and solution modules. The comments in the preprocessor subprogram DISCRT give a general guide; INDEX and SOLUT are handled similarly and we discuss the DISCRT case in detail.

Step 1. Choose a new descriptive name for the user to use. Put that in an empty place (index NUNAME) in the NAMEDS array, character by character. Only the first 20 characters are used.

Step 2. Put the number of non-blank characters in the descriptive name in NAMELG(NUNAME) with the DATA statement provided.

Step 3. Choose a Fortran subroutine name and put it in the corresponding empty places (index NUNAME) in the NAMESB array. You may use an existing Format statement (601, 603, 604, 607, 609, 610) if the argument list is what you want. Otherwise you must create a new Format 600 + NUNAME to call your subroutine with your desired argument list.

Step 4. At the continuation statement 100 + 10*NUNAME insert the Fortran statements to define the storage allocation variables

KBAND = band width of equations as generated

KXNCID = column size of IDCOEF array to be declared

KXNCOE = column size of COEF array to be declared

KXNEQ = row sizes of COEF and IDCOEF to be declared

KWORKS = amount of workspace needed

If some storage allocations cannot be made until all ELLPACK segments are processed, then the array ENDEC of switches may be used to do that calculation in CLOSER. This switch may be used for other things as illustrated by the code at the beginning of CLOSER.

Note that the CALL you write will be timed, you can time separate phases of your module by breaking it into pieces and putting in intervening timing calls as in FORMAT 620 (with obvious modifications).

The dummy modules in ELLPACK 77 are:

```
TEST DISCRETIZATION
    CALL TESTDI(GRIDX,NGRDXD,GRIDY,NGRDYD,GRIDZ,NGRDZD,
   A            COEF,MXNCOE,MXNEQ,IDCOEF,IGRID,UNKNWN,
   B            AMATRX,BVECTR,NROWD,NCOLD)
```

Associated declarations:

```
    REAL GRIDX(NGRDXD),GRIDY(NGRDYD),GRIDZ(NGRDZD),
   A     COEF(MXNCOE,MXNEQ), AMATRX(NROWD,NCOLD),BVECTR(NROWD)
   B     UNKNWN(MXNEQ)
    INTEGER IDCOEF(MXNCOE,MXNEQ),IGRID(MXNEQ)
```

TEST INDEX

```
      CALL TESTIN(COEF,MXNCOE,MXNEQ,IDCOFF,AMATRX,NROWD,
     A            NCOLD,BVECTR,NDXEQ,NDXUNK,INVNDX,IGRID)
```

Associated declarations:

```
      REAL COEF(MXNCOE,MXNEQ),AMATRX(NROWD,NCOLD),BVECTR(NROWD)
      INTEGER IDCOEF(MXNCOE,MXNEQ),NDXEQ(MXNEQ),NDXUNK(MXNEQ),
              INVNDX(MXNEQ),IGRID(MXNEQ)
```

TEST SOLUTION

```
      CALL TESTSO(COEF,MXNCOE,MXNEQ,IDCOEF,AMATRX,NROWD,
                  NCOLD,BVECTR,NDXEQ,NDXUNK,INVDX,IGRID,
                  UNKNWN)
```

Associated declarations:

```
      REAL COEF(MXNCOE,MXNEQ),AMATRX(NROWD,NCOLD),BVECTR(NROWD),
     A     UNKNWN(MXNEQ)
      INTEGER IDCOEF(MXNCOE,MXNEQ),NDXEQ(MXNEQ),NDXUNK(MXNEQ),
           INVNDX(MXNEQ),IGRID(MXNEQ)
```

There is a built-in facility to read input parameters from the ELLPACK 77 user statements, e.g.

```
      DISCRETIZATION.   HOLR 9-POINT(IORDER=4)
```

The Fortran statement "IORDER=4" is automatically written just before the subroutine call associated with HOLR 4-POINT and IORDER is an argument to this subroutine. An unlimited number of such statements can be included, separated by commas.

3.2 Module Interface with Control Program. Appendix One contains a sample ELLPACK 77 control program and the file SEQUENCE contains the calls on the modules. Each module is automatically preceded by

```
   50X CONTINUE
       IF(TIMER) READ CLOCK
       assign indicator value of current module index
             (e.g. INDIS = 4 or INSOL = 2)
       assignment statements that pass parameters from user to
```

ELLPACK control

and each module is followed by

```
IF( .NO7. TIMER)                        GO TO 600
    READ CLOCK
    TIMES(KTIMES) = change in clock readings
    KTIMES = KTIMES + 1
    GO TO 600
```

In between these statement groups the contributor can have any Fortran code written that he likes by putting them in appropriate WRITE statements in the preprocessor. See, for example, the 22 lines (9 statements) written for an output module at 504 in Appendix One.

Two notes: The variable INITL may be set to 1 to signal that certain initialization steps are needed. Thus in some output, certain tables are filled on the very first call to subprograms of a module, then INITL is set to 0 and this code is skipped on later calls. If the contributor wants to time K+1 pieces of his module, then his code in the preprocessor must have the statement

```
TEXTRA = TEXTRA + K
```

in the assignments of preprocessor variables associated with the module. This allows the TIMES array to be dimensioned sufficiently large.

Finally, we repeat that the module arguments must coincide with the ELLPACK 77 variables and the module's declarations must contain appropriate labeled common blocks to communicate values of ELLPACK 77 variables.

4. USER INTERFACE AND OTHER CONSIDERATIONS.

There are several things influenced by modules which are not directly controlled by them. It is helpful to keep these external requirements in mind while preparing modules. Five such considerations are discussed here.

4.1 Intermediate Output. We may visualize three primary modes of operation of a program (including ELLPACK). One is where the user just wants the result quickly. Another is where a knowledgeable user wants to know in detail what happended (probably things have gone wrong). The third is where a user wants to confirm his hopes that things are alright (or check his suspicions that things are in error) and thus he wants a modest description of how the calculation went. In other words, the user may want little, some or a lot of intermediate output. Each module should print a line saying it has executed.

Intermediate output levels are provided through the control variables LEVEL and DEBUG. Each module contributor must provide the mechanisms to react to these control settings. One mark of a high quality, usable program is that meaningful intermediate output is available; to provide this is often surprisingly difficult.

4.2 Final Output. There are several items of final output which must be provided from information generated by various modules. The most critical of these is the requirement to tabulate or plot the computed solution, residual, true solution or error. The module that solves the equations may have no idea of the interpretation of the unknowns and the tabulation and plotting routines must be independent of the particular way the differential equation is solved. Thus the contributors of modules that form the linear equations (and thus define the unknowns) must provide a function subprogram

which allows one to evaluate the approximate solution at any arbitrary point. If the solution is obtained as a set of tabulated values then this subprogram should use some interpolation scheme compatible with the accuracy expected in the approximate solution. ELLPACK provides quadratic interpolation.

4.3 Performance Monitoring. One useful kind of final output relates to the performance of a module, e.g. execution time used, memory actually used, sizes of particular matrices, etc. Each contributor should identify any information of this type which he feels would be valuable and which can be made available without much difficulty. Timing of complete modules will be made by ELLPACK itself.

It is possible to time parts of a module by breaking it up into two or more subroutines. Thus the typical situations is, schematically,

```
READ CLOCK
CALL MODULE
READ CLOCK, COMPUTE ELAPSED TIME AND SAVE IT
```

If a contributor desires, he may break his module up and insert it as:

```
READ CLOCK
CALL PHASE 1
READ CLOCK, COMPUTE ELAPSED TIME AND SAVE IT
CALL PHASE 2
READ CLOCK, COMPUT  ELAPSED TIME AND SAVE IT
```

Since a user expects the final timing output to have one time per module, it is strongly suggested that a contributor who does this print out an explanatory line for all but the lowest level of output. Note that the preprocessor must be informed of extra clock readings by inserting a statement

```
TEXTRA = TEXTRA+K
```

with the module text whenever a contributor reads the clock K extra times.

4.4 Pre-execution Estimates. There are estimates which must be made before execution begins. These are the dimension values to be used in the arrays passed to any particular module. Each contributor must provide a formula for computing these values based on the problem definition information. These formulas are used by the ELLPACK preprocessor for the declaration statements in the ELLPACK control program. It would be helpful if a contributor could provide formulas for approximate estimates of execution time and total memory used for a module. These formulas can include only problem definition information plus one or two parameters for calibration to particular machines. It is expected that there will be a mode of ELLPACK use where only the input processor is executed to check the problem definition and to obtain estimates of the execution time and memory usage to be expected. This has not been implemented as of Summer, 1978.

4.5 Legal Module Use. The contributor of each module is required to prepare a short statement which defines, as precisely as possible, the limits on the use of the module in combination with other ELLPACK modules. The information should be in a form that can be readily translated to direct decisions about the legality of the use of the module in various circumstances. These decisions will be made by the input processor and ELLPACK control so as to minimize waste from attempting illegal module combinations. This has not been implemented as of Summer, 1978.

4.6 PORTABILITY NOTES.

The success of ELLPACK depends on being able to use programs from various places together. No one has the time to rewrite programs just to get them to run on machines different from the ones upon which they were developed. There are currently two approaches to portability: automatic

translation of Fortran code from one dialect to another and use of a Fortran subset that works on all common machines. For a description of the first approach and a survey of other work, see T.J. Aird, E.L. Battiste and W.C. Gregory, Portability of mathematical software coded in Fortran, ACM Trans. Mathematical Software 3 (1977), 113-127. The second approach is best illustrated by use of the PFORT verifier, see B.G. Ryder, The PFORT verifier, Software Practice and Experience, 4 (1974) 359-377. There are several useful Fortran compilers which check for ANSI standard Fortran violations. ELLPACK contributors may use a portable subset of Fortran augmented by very modest amounts of hand translation where essential; the preprocessor and some modules will rely on the Fortran Converter of Aird et al.

Several helpful rules for portability should be followed.

A. Provide well documented, commented, modular and structured programs.

B. Clearly identify and localize things that are

   i. Machine dependent (tolerance parameters, unusual instructions required, double precision requirements)

   ii. System dependent (timing routines, local output and plotting routines, scratch tapes)

   iii. Dimension dependency. Try not to assume that ELLPACK will always use less than 10 boundary pieces or less than 100,000 words of central memory, etc.

   iv. Machine or Assembly language. (Efficient packing of information, graphics facilities, inner loops of equation solvers)

C. Use ANSI standard Fortran if at all possible, avoid features which are implemented in various ways (ENTRY points, nonstandard RETURN)

D. Remember that for scientific computing IBM computers are usually run in double precision. Various things can go wrong here such as: ABS must be DABS, 1.3E+3 must be 1.3D+3, 1.23 might need to be 1.23D0, the format E12.3 might not work, etc. In particular, all REAL variables and functions should be explicitly declared and room (16 spaces) left to change REAL to DOUBLE PRECISION without moving anything else. Mixing different types (REAL and INTEGER) of variables in COMMON causes trouble.

E. Remember that IBM uses only 4 characters per word, thus formats for character strings should be A4 or shorter. Some compilers balk at numerical constants longer than their word length allows.

Most of the rules cause little trouble if used in the initial program development. The small costs in time and storage for following these rules are minor compared to the savings in human time which is the scarcest resource in ELLPACK.

4.7 ELLPACK 78. The ELLPACK 77 system is partially a test bed for the ELLPACK 78 system which contains facilities for general geometry. Some aspects of ELLPACK 78 are described in the ELLPACK Contributor's Guide, CSD-TR 208, revised Sept. 16, 1977. ELLPACK 78 will become operational in the summer of 1978 and when a reasonably stable system is available to the ELLPACK group, it will be documented with a user's and contributor's guide. The current version of ELLPACK 77 already contains certain facilities for ELLPACK 78 which may appear mysterious to someone who studies the pre-processor or control programs in any detail. Hopefully, these items (which are principally unused variables and arrays) will not cause any confusion.

5. ELLPACK IMPLEMENTATION.

This section contains a brief description of how ELLPACK may be implemented. Considerable detail is given in the ELLPACK Distribution Guide for those who actually want to run ELLPACK. We plan to make the ELLPACK system reasonably portable. However, to make it truely portable in the sense of running without modification on any system with standard Fortran imposes a large penalty on ELLPACK. Thus we make the following assumptions about the environments where ELLPACK is to be used.

A. A standard Fortran compiler is available.

B. The operating system allows Fortran preprocessors. One major manufacturer has a flaw in its operating system design which requires trickery to implement Fortran preprocessors.

C. The operating system allows a reasonable number of files (probably on disks) to be created by a job and simply manipulated. Library files of already compiled programs and other information are possible.

D. Certain common utility routines are available. These include file copying and concatenation, timing routines, and hard copy graphical output. The lack of some of these utilities can be overcome by simply deleting the corresponding features from ELLPACK.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++                                                  ++++++++++
++++++++++      FILE HEADER   FOR ELLPACK 77                ++++++++++
++++++++++                                                  ++++++++++
 PROGRAM ELL77(INPUT,OUTPUT,PLOT,SAVE,TAPE4=SAVE,
A             TAPE5=INPUT,TAPE6=OUTPUT)
 LOGICAL DIM2,DIM3,POISON,LAPLAC,CONSTC,SELFAD,CROSST,
A        DIRICH,HOMOEQ,NEUMAN,MIXED,HOMOBC,UNIFRM,
B        DEBUG,TIMER,RECTAN,SYMMET
 REAL AX,BX,AY,BY,AZ,BZ,HX,HY,HZ,CUXX,CUXY,CUYY,CUX,
A     CUY,CU,CUXZ,CUYZ,CUZZ,CUZ,GRIDX,GRIDY,GRIDZ,
B     AMATRX,BVECTR,UNKNWN,COEF,TABLEM,BPARAM,XBOUND,
C     YBOUND
 INTEGER IROT(6)
 EXTERNAL SOLUT,ERROR,RESID,TRUEEP
 COMMON WORKSP
 COMMON/ PROBL / DIM2,DIM3,POISON,LAPLAC,CONSTC,SELFAD,
A        HOMOEQ,CROSST,HOMOBC,DIRICH,NEUMAN,MIXED,UNIFRM
 COMMON/ PROBR / AX,BX,AY,BY,AZ,BZ,HX,HY,HZ
 COMMON/ PROBI / NGRIDX,NGRIDY,NGRIDZ,BCTYPE
 COMMON/ CPDE  / CUXX,CUXY,CUYY,CUX,CUY,CU,
A                CUXZ,CUYZ,CUZZ,CUZ
 COMMON/ CONTRL/ DEBUG,TIMER,SYMMET
 COMMON/ INTEGS/ NUMBEQ,NUMCOE,NROW,NCOL,NBAND,IROT,
A                LEVEL,IPACK1,IPACK2,INITL,INDIS,INSOL,
B                MINPUT,MOUTPT,MEMORY,ININD,NGRDXD,
C                NGRDYD,NGRDZD,MXNEQ,MXNCOE,NROWD,NCOLD
 COMMON / REALS / ERRMAX,SOLMAX,TRUMAX,RESMAX,TIMES
 COMMON / BNDRY / IPIECE,NBOUND,NBNDPT
 COMMON / COEFZZ / COEF
 COMMON / IDCOZZ / IDCOEF
 COMMON / IGRIZZ / IGRID
 COMMON / NDXEZZ / NDXEQ
 COMMON / NDXUZZ / NDXUNK
 COMMON / INVNZZ / INVNDX
 COMMON / AMATZZ / AMATRX
 COMMON / BVECZZ / BVECTR
 COMMON / UNKNZZ / UNKNWN
 COMMON / GRIDXZ / GRIDX
 COMMON / GRIDYZ / GRIDY
 COMMON / GRIDZZ / GRIDZ
 COMMON / TABLZZ / TABLEM
 COMMON / BRANZZ / BRANGE
 COMMON / GTYPZZ / GTYPE
 COMMON / XBOUZZ / XBOUND
 COMMON / YBOUZZ / YBOUND
 COMMON / PIECZZ / PIECE
 COMMON / BPTYZZ / BPTYPE
 COMMON / BNEIZZ / BNEIGH
 COMMON / BPARZZ / BPARAM
 COMMON / BGRIZZ / BGRID
 DIMENSION GRIDX(5     )
 DIMENSION GRIDY(7     )
 DIMENSION GRIDZ(1     )
 DIMENSION MODSQ(11)
 DIMENSION WORKSP(  4004)
 DIMENSION COEF( 140,17)
 DIMENSION IDCOEF( 140,17)
 DIMENSION IGRID( 140)
 DIMENSION NDXEQ( 140),NDXUNK( 140),INVNDX( 140)
 DIMENSION AMATRX( 140,  72),UNKNWN( 140),BVECTR( 140)
 INTEGER BCTYPE(   6)
 INTEGER   GTYPE(  1,  1)
 INTEGER   BRANGE(2,  1)
 DIMENSION XBOUND(  1),YBOUND(  1),BPARAM(  1)
 INTEGER   PIECE (  1),BPTYPE(  1),BNEIGH(  1)
 INTEGER   BGRID (  1)
 DIMENSION TABX(    5),TABY(    5),TABZ(    1)
 DIMENSION TABLEM(    5,    7,    1)
 INTEGER NAMES(4,2)
 REAL            TIMES(  12)
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++                                                ++++++++++
++++++++++      FILE DATA     FOR ELLPACK 77              ++++++++++
++++++++++                                                ++++++++++
      DATA MODSQ / 6, 1, 3, 5, 7, 2, 4, 5, 7, 8,
     B             9/
      DATA NAMES(1,1),NAMES(1,2),NAMES(2,1),NAMES(2,2),NAMES(3,1),
     A     NAMES(3,2),NAMES(4,1),NAMES(4,2)
     B     / 4HRESI,4HDUAL,4HSOLU,4HTION,4HTRUE,4H    ,4HERRO,4HR     /
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++                                                ++++++++++
++++++++++      FILE ASSIGN   FOR ELLPACK 77              ++++++++++
++++++++++                                                ++++++++++
      DIM2   = .TRUE.
      DIM3   = .FALSE.
      POISON = .FALSE.
      LAPLAC = .FALSE.
      CONSTC = .FALSE.
      SELFAD = .FALSE.
      HOMOEQ = .FALSE.
      HOMOBC = .FALSE.
      CROSST = .FALSE.
      RECTAN = .FALSE.
      UNIFRM = .FALSE.
      DIRICH = .TRUE.
      NEUMAN = .FALSE.
      MIXED  = .FALSE.
      DEBUG  = .FALSE.
      TIMER  = .FALSE.
      MINPUT = 5
      MOUTPT = 6
      DO 5 I = 1,6
    5     IROT(I) = 0
      IROT(1) = 2
      IROT(3) = -2
      LEVEL = 1
      MXNUNK = 1
      DO 6 I = 1,6
    6     BCTYPE(I) = 0
      AZ = 0.0
      BZ = 0.0
      ERRMAX = 0.0
      RESMAX = 0.0
      HZ = 0.0
      MEMORY = 0
      TFIRST = 0.0
      TLAST = 0.0
      KTIMES = 1
      AX = 0.0
      BX = EXP(1.)
      IF( AX .LT. BX ) GO TO 210
      ASAVE = AX
      AX = BX
      BX = ASAVE
      IROT(1) = 0
      IROT(3) = 0
  210 CONTINUE
      AY = 1.0
      BY = 0.0
      IF( AY .LT. BY ) GO TO 220
      ASAVE = AY
      AY = BY
      BY = ASAVE
      IROT(2) = +2
      IROT(4) = -2
  220 CONTINUE
      IR = 1 + IROT(1)
      BCTYPE(IR) = 1
      IR = 2 + IROT(2)
      BCTYPE(IR) = 2
      IR = 3 + IROT(3)
      BCTYPE(IR) = 1
      IR = 4 + IROT(4)
      BCTYPE(IR) = 3
      MIXED = .TRUE.
```

```
      NGRDXD = 5
      NGRIDX = 5
      HX = (BX-AX)/  4.
      DO 11 J = 1,  4
   11    GRIDX(J) = AX + FLOAT(J-1)*HX
      GRIDX(  5) = BX
      NGRIDY = 7
      NGRDYD = 7
      HY = (BY-AY)/  6.
      DO 12 J = 1,  6
   12    GRIDY(J) = AY + FLOAT(J-1)*HY
      GRIDY(  7) = BY
      UNIFRM = .TRUE.
      NGRDZD = 1
      NGRIDZ = 1
      TIMER = .TRUE.
      NPDIM =    1
      NBDIM  =    1
      MXNCOE =  17
      MXNEQ  = 140
      NROWD  = 140
      NCOLD  =  72
      NBAND  =  35
      MEMORY =   22257
      IPACK1 = (MXNUNK+1)*(NGRIDX+1)
      IPACK2 = IPACK1*(NGRIDY+1)
      DO 7 I = 1,    12
    7    TIMES(I) = 0.
      CALL SECOND(TFIRST)
      TIMES(  12) = TFIRST
      CALL PLOTS
     ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
     ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
     ++++++++++                                              ++++++++++
     ++++++++++      FILE SEQUENCE FOR ELLPACK 77            ++++++++++
     ++++++++++                                              ++++++++++
      KSEG = 0
                                                          GO TO 600
  500 CONTINUE
      IF( TIMER ) CALL SECOND(TFIRST)
      INITL = 1
      INDIS =  1
      CALL STAR5 (GRIDX,NGRDXD,GRIDY,NGRDYD,COEF,MXNCOE,MXNEQ,IDCOEF,
     A            IGRID)
      IF( .NOT. TIMER )                       GO TO 600
      CALL SECOND(TLAST)
      TIMES(KTIMES) = TLAST-TFIRST
      KTIMES = KTIMES+1
      GO TO 600
  501 CONTINUE
      IF( TIMER ) CALL SECOND(TFIRST)
      INITL = 1
      INDIS =  6
      CALL P3C1CO(GRIDX,NGRDXD,GRIDY,NGRDYD,COEF,MXNCOE,MXNEQ,IDCOEF,
     A            IGRID)
      IF( .NOT. TIMER )                       GO TO 600
      CALL SECOND(TLAST)
      TIMES(KTIMES) = TLAST-TFIRST
      KTIMES = KTIMES+1
      GO TO 600
  502 CONTINUE
      IF( TIMER ) CALL SECOND(TFIRST)
      ININD =  3
      CALL NATORD(COEF,MXNCOE,MXNEQ,IDCOEF,AMATRX,NROWD,NCOLD,
     A            BVECTR,NDXEQ,NDXUNK,INUNDX)
      IF( .NOT. TIMER )                       GO TO 600
      CALL SECOND(TLAST)
      TIMES(KTIMES) = TLAST-TFIRST
      KTIMES = KTIMES+1
      GO TO 600
  503 CONTINUE
      IF( TIMER ) CALL SECOND(TFIRST)
      ININD =  1
      CALL COLBND(COEF,MXNCOE,MXNEQ,IDCOEF,AMATRX,NROWD,NCOLD,
     A            BVECTR,NDXEQ,NDXUNK,INUNDX)
      IF( .NOT. TIMER )                       GO TO 600
      CALL SECOND(TLAST)
      TIMES(KTIMES) = TLAST-TFIRST
      KTIMES = KTIMES+1
      GO TO 600
```

```
  504 CONTINUE
      IF( TIMER ) CALL SECOND(TFIRST)
      INSOL =   2
      CALL BNDSOL(AMATRX,NROWD,NCOLD,UNKNWN,BVECTR,MXNEQ)
      IF( .NOT. TIMER )                                  GO TO 600
      CALL SECOND(TLAST)
      TIMES(KTIMES) = TLAST-TFIRST
      KTIMES = KTIMES+1
      GO TO 600
  505 CONTINUE
      IF( TIMER ) CALL SECOND(TFIRST)
      INITL = 1
      CALL CONTUR(TRUEEP,NAMES(3,1),NAMES(3,2),
     A            NGRDXD,NGRDYD,NGRDZD,GRIDX,GRIDY,GRIDZ,
     B            UNKNWN,MXNEQ,NDXUNK,TABLEM,COEF,IDCOEF,MXNCOE)
      CALL REGPLT(GRIDX,GRIDY,NGRIDX,NGRIDY,AX,BX,AY,BY)
      IF( .NOT. TIMER )                                  GO TO 600
      CALL SECOND(TLAST)
      TIMES(KTIMES) = TLAST-TFIRST
      KTIMES = KTIMES+1
      GO TO 600
  506 CONTINUE
      IF( TIMER ) CALL SECOND(TFIRST)
      INITL = 1
      ERRMAX = FNCMAX(ERROR ,NAMES(4,1),NAMES(4,2),
     $            NGRIDX,NGRIDY,NGRIDZ,GRIDX,GRIDY,GRIDZ,
     A            NGRDXD,NGRDYD,NGRDZD,GRIDX,GRIDY,GRIDZ,
     B            UNKNWN,MXNEQ,NDXUNK,TABLEM,COEF,IDCOEF,MXNCOE)
      RESMAX = FNCMAX(RESID ,NAMES(1,1),NAMES(1,2),
     $            NGRIDX,NGRIDY,NGRIDZ,GRIDX,GRIDY,GRIDZ,
     A            NGRDXD,NGRDYD,NGRDZD,GRIDX,GRIDY,GRIDZ,
     B            UNKNWN,MXNEQ,NDXUNK,TABLEM,COEF,IDCOEF,MXNCOE)
      IF( .NOT. TIMER )                                  GO TO 600
      CALL SECOND(TLAST)
      TIMES(KTIMES) = TLAST-TFIRST
      KTIMES = KTIMES+1
      GO TO 600
  507 CONTINUE
      IF( TIMER ) CALL SECOND(TFIRST)
      INITL = 1
      NTABX =     5
      NTABY =     5
      NTABZ =     1
      STEP = (BX-AX)/  4.
      DO   701 I=1,  4
  701     TABX(I) = AX + FLOAT(I-1)*STEP
      TABX(  5) = BX
      STEP = (BY-AY)/  4.
      DO   702 I=1,  4
  702     TABY(I) = AY + FLOAT(I-1)*STEP
      TABY(  5) = BY
      CALL TABLER(SOLUT ,NAMES(2,1),NAMES(2,2),
     $            NTABX,NTABY,NTABZ,TABX,TABY,TABZ,
     A             NGRDXD,NGRDYD,NGRDZD,GRIDX,GRIDY,GRIDZ,
     B             UNKNWN,MXNEQ,NDXUNK,TABLEM,COEF,IDCOEF,MXNCOE)
      IF( .NOT. TIMER )                                  GO TO 600
      CALL SECOND(TLAST)
      TIMES(KTIMES) = TLAST-TFIRST
      KTIMES = KTIMES+1
      GO TO 600
  598 CONTINUE
  600 KSEG = KSEG+1
      IF( KSEG .GT. 10)     GO TO 605
      MSEG = MODSQ(KSEG)
      GO TO(500,501,502,503,504,505,506,507,
     B                      600,598,650), MSEG
  605 CONTINUE
  650 CONTINUE
      CALL SECOND(TFINAL)
      TIMES(KTIMES) = TFINAL - TIMES(  12)
      WRITE(MOUTPT,700) (I,TIMES(I),I=1,KTIMES)
  700 FORMAT(/////3X,34HEXECUTION TIME FOR MODULES  (SEC.)/
     A  3X,26(1H-)//(4(5X,I3,1X,6HTIME =,F7.2,3X)/))
      CALL PLOT(0.,0.,999)
      STOP
      END
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++                                              ++++++++++
++++++++++      FILE FORTRAN  FOR ELLPACK 77            ++++++++++
++++++++++                                              ++++++++++
      SUBROUTINE PDE(X,Y,CVALUS)
      REAL             CVALUS(7)
      CVALUS( 1) = 1.0
      CVALUS( 2) = 0.0
      CVALUS( 3) = +6.
      CVALUS( 4) = 0.0
      CVALUS( 5) = -4.
      CVALUS( 6) = +(DUB9(X)-3.)
      CVALUS( 7) = EXP(X+Y)*DUB9(X)*(2./(1.+X)-1.)
      RETURN
      END
      REAL             FUNCTION PDERHS(X,Y)
      PDERHS = EXP(X+Y)*DUB9(X)*(2./(1.+X)-1.)
      RETURN
      END
      REAL             FUNCTION BCOND(I,X,Y,BVALUS)
      REAL             BVALUS(4)
      COMMON / INTEGS / NUMBEQ,NUMCOE,NROW,NCOL,NBAND,IROT(6),
     A                  LEVEL,IPACK1,IPACK2,INITL,INDIS,INSOL,
     B                  MINPUT,MOUTPT,MEMORY,ININD,NGRDXZ,
     C                  NGRDYZ,NGRDZZ,MXNEQZ,MXNCOZ,
     D                  NROWDZ,NCOLDZ
      ISIDE = I + IROT(I)
      GOTO(10,20,30,40),ISIDE
   10 BVALUS(1) = 1.0
      BVALUS(4) =TRUE(0.0,Y)
      BVALUS(2) = 0.0
      BVALUS(3) = 0.0
                 GO TO 999
   20 BVALUS(3) = 1.0
      BVALUS(4) =EXP(1.+X)*SQRT(DUB9(X)/2.)
      BVALUS(1) = 0.0
      BVALUS(2) = 0.0
                 GO TO 999
   30 BVALUS(1) = 1.0
      BVALUS(4) =TRUE(2.71828182846,Y)
      BVALUS(2) = 0.0
      BVALUS(3) = 0.0
                 GO TO 999
   40 CONTINUE
      BVALUS(4) =2.*EXP(X)
      BVALUS(1) =1.+X
      BVALUS(3) =1.+X
      BVALUS(2) = 0.0
                 GO TO 999
  999 BCOND = BVALUS(4)
      RETURN
      END
      FUNCTION TRUE(X,Y)
      TRUE = EXP(X+Y)/(1.0+X)
      RETURN
      END
      FUNCTION DUB9(T)
      DUB9 = 2./(1.+T)**2
      RETURN
      END
C
C--------  THIS ENDS THE FORTRAN PART CREATED BY THE ELLPACK ----------
C          PREPROCESSOR.  THE FOLLOWING PROGRAMS ARE LOADED, ALREADY
C          COMPILED, FROM A LIBRARY.  WE INDICATE THEIR TYPICAL
C          FORTRAN VERSIONS
C
C-------- THE APPROXIMATE SOLUTION EVALUATION PROGRAM -------
      FUNCTION SOLUT(X,Y,UNKNWN,MXNEQ,NDXUNK)
C          -----
      END
      SUBROUTINE STAR5(GRIDX,NGRDXD,GRIDY,NGRDYD,COEF,MXNCOE,MXNEQ,
     A                 IDCOEF,IGRID)
** CALL DECK DECLARE = STANDARD ELLPACK SYSTEM DECLARATIONS
      COMMON TEMP1,TEMP2,SAVE(8),ITHNK(10)
      REAL             GRIDX(NGRDXD),GRIDY(NGRDYD),COEF(MXNCOE,MXNEQ)
      INTEGER IDCOEF(MXNCOE,MXNEQ),IGRID(MXNEQ)
C          ----
      END
```

```
      SUBROUTINE NATORD(COEF,MXNCOE,MXNEQ,IDCOEF,AMATRX,NROWD,
     A                  NCOLD,BVECTR,NDXEQ,NDXUNK,INVNDX)
** CALL DECK DECLARE = STANDARD ELLPACK SYSTEM DECLARATIONS
      COMMON LIST1(48),LIST2(2,48),TABLE(6,6)
      REAL             COEF(MXNCOE,MXNEQ),AMATRX(NROWD,NCOLD),BVECTR(NROWD)
      INTEGER IDCOEF(MXNCOE,MXNEQ),NDXUNK(MXNEQ),NDXEQ(MXNEQ),
     A        INVNDX(MXNEQ)
C          -----
      END
C
      SUBROUTINE BNDSOL(AMATRX,NROWD,NCOLD,BVECTR,UNKNWN,MXNEQ)
** CALL DECK DECLARE = STANDARD ELLPACK SYSTEM DECLARATIONS
      COMMON ATEMP(48,16),BTEMP(48,2),VEC1(48),IPIVOT(48),GLUG
      REAL             AMATRX(NROWD,NCOLD),UNKNWN(MXNEQ),BVECTR(NROWD)
C          -----
      END
C
      SUBROUTINE CONTUR(TRUEEP,NAME1,NAME2,NGRDXD,NGRDYD,GRIDX,GRIDY,
     A                  UNKNWN,MXNEQ,NDXUNK,TABLEM,COEF,IDCOEF,MXNCOE)
** CALL DECK DECLARE = STANDARD ELLPACK SYSTEM DECLARATIONS
      COMMON LPLOT(130),CONTUR(250),CRSECT(130)
      REAL             GRIDX(NGRDXD),GRIDY(NGRDYD),COEF(MXNCOE,MXNEQ),
     A                 UNKNWN(MXNEQ),TABLEM(NGRDXD,NGRDYD)
      INTEGER IDCOEF(MXNCOE,MXNEQ),NDXUNK(MXNEQ)
C          -----
      END
C
C---------MORE OUTPUT SUBROUTINES ADDED FROM LIBRARY-----------------
C
C-------- END OF PROGRAMS LOADED FROM ELLPACK BINARY LIBRARY ----------
C
C
C--------  THIS ENDS THE FORTRAN PART CREATED BY THE ELLPACK ----------
C          PREPROCESSOR.  THE FOLLOWING PROGRAMS ARE LOADED, ALREADY
C          COMPILED, FROM A LIBRARY.  WE INDICATE THEIR TYPICAL
C          FORTRAN VERSIONS
C
C-------- THE APPROXIMATE SOLUTION EVALUATION PROGRAM -------
      FUNCTION SOLUT(X,Y,UNKNWN,MXNEQ,NDXUNK)
C          -----
      END
      SUBROUTINE STARS(GRIDX,NGRDXD,GRIDY,NGRDYD,COEF,MXNCOE,MXNEQ,
     A                 IDCOEF,IGRID)
** CALL DECK DECLARE = STANDARD ELLPACK SYSTEM DECLARATIONS
      COMMON TEMP1,TEMP2,SAVE(8),ITHNK(10)
      REAL             GRIDX(NGRDXD),GRIDY(NGRDYD),COEF(MXNCOE,MXNEQ)
      INTEGER IDCOEF(MXNCOE,MXNEQ),IGRID(MXNEQ)
C          ----
      END
C
      SUBROUTINE NATORD(COEF,MXNCOE,MXNEQ,IDCOEF,AMATRX,NROWD,
     A                  NCOLD,BVECTR,NDXEQ,NDXUNK,INVNDX)
** CALL DECK DECLARE = STANDARD ELLPACK SYSTEM DECLARATIONS
      COMMON LIST1(48),LIST2(2,48),TABLE(6,6)
      REAL             COEF(MXNCOE,MXNEQ),AMATRX(NROWD,NCOLD),BVECTR(NROWD)
      INTEGER IDCOEF(MXNCOE,MXNEQ),NDXUNK(MXNEQ),NDXEQ(MXNEQ),
     A        INVNDX(MXNEQ)
C          -----
      END
C
      SUBROUTINE BNDSOL(AMATRX,NROWD,NCOLD,BVECTR,UNKNWN,MXNEQ)
** CALL DECK DECLARE = STANDARD ELLPACK SYSTEM DECLARATIONS
      COMMON ATEMP(48,16),BTEMP(48,2),VEC1(48),IPIVOT(48),GLUG
      REAL             AMATRX(NROWD,NCOLD),UNKNWN(MXNEQ),BVECTR(NROWD)
C          -----
      END
C
      SUBROUTINE CONTUR(TRUEEP,NAME1,NAME2,NGRDXD,NGRDYD,GRIDX,GRIDY,
     A                  UNKNWN,MXNEQ,NDXUNK,TABLEM,COEF,IDCOEF,MXNCOE)
** CALL DECK DECLARE = STANDARD ELLPACK SYSTEM DECLARATIONS
      COMMON LPLOT(130),CONTUR(250),CRSECT(130)
      REAL             GRIDX(NGRDXD),GRIDY(NGRDYD),COEF(MXNCOE,MXNEQ),
     A                 UNKNWN(MXNEQ),TABLEM(NGRDXD,NGRDYD)
      INTEGER IDCOEF(MXNCOE,MXNEQ),NDXUNK(MXNEQ)
C          -----
      END
C
C---------MORE OUTPUT SUBROUTINES ADDED FROM LIBRARY-----------------
C
C-------- END OF PROGRAMS LOADED FROM ELLPACK BINARY LIBRARY ----------
C
```

ELLPACK 77
STANDARD NAMES

***** FUNCTIONS AND SUBROUTINES ******

BCOND = FUNCTION WITH BOUNDARY CONDITION COEFFICIEIENTS
BCOORD = SUBROUTINE WITH THE BOUNDARY DEFINITION FUNCTIONS E78
PDE = SUBROUTINE WITH PARTIAL DIFFERENTIAL EQUATION COEFFICIENTS
RHSPDE = RIGHT HAND SIDE OF PARTIAL DIFFERENTIAL EQUATION
SOLUT = EVALUATION OF THE APPROXIMATE SOLUTION
TRUE = THE TRUE SOLUTION (IF KNOWN)

****** = THE ONES BELOW ARE FROM CONTRIBUTORS

ABASIS = CONTRIBUTOR-S MODULE
ALPHAS = CONTRIBUTOR-S MODULE
APXUNK = ITPACK MODULE
BANDWI = CONTRIBUTOR-S MODULE
BASE = CONTRIBUTOR-S MODULE
BBASIS = CONTRIBUTOR-S MODULE
BCUBCO = CONTRIBUTOR-S MODULE
BDHOMG = CONTRIBUTOR-S MODULE
BETALN = ITPACK MODULE
BETAS = CONTRIBUTOR-S MODULE
BETA = ITPACK MODULE
BICUBH = CONTRIBUTOR-S MODULE
BNDEQ5 = CONTRIBUTOR-S MODULE
BNDPC5 = CONTRIBUTOR-S MODULE
BNDSOL = CONTRIBUTOR-S MODULE
BOUNDR = CONTRIBUTOR-S MODULE
CDXU = CONTRIBUTOR-S MODULE
CDYU = CONTRIBUTOR-S MODULE
CGAPRH = CONTRIBUTOR-S MODULE
CHCJCG = ITPACK MODULE
CHEBY = ITPACK MODULE
CHGJCG = ITPACK MODULE
CHGJSI = ITPACK MODULE
CHRSCG = ITPACK MODULE
CHRSSI = ITPACK MODULE
CHSRCG = ITPACK MODULE
CHSRPA = ITPACK MODULE
CHSRSI = ITPACK MODULE
CJCG = ITPACK MODULE
COLAPR = CONTRIBUTOR-S MODULE
COLBND = CONTRIBUTOR-S MODULE
CONTUR = CONTRIBUTOR-S MODULE
CRED = CONTRIBUTOR-S MODULE
DBASE = CONTRIBUTOR-S MODULE
DDBASE = CONTRIBUTOR-S MODULE
DETERM = ITPACK MODULE
DIANDX = ITPACK MODULE
DIRBC9 = CONTRIBUTOR-S MODULE
DISCRT = CONTRIBUTOR-S MODULE
DRAW = CONTRIBUTOR-S MODULE
EIGVAL = ITPACK MODULE
ELIMD5 = CONTRIBUTOR-S MODULE
END1 = ITPACK MODULE
END2 = ITPACK MODULE
EQSOL = CONTRIBUTOR-S MODULE
ERROR = CONTRIBUTOR-S MODULE
EVENRD = CONTRIBUTOR-S MODULE
FETCHX = CONTRIBUTOR-S MODULE
FFT9 = CONTRIBUTOR-S MODULE
FILL0 = CONTRIBUTOR-S MODULE
FNCMAX = CONTRIBUTOR-S MODULE
FOUR = CONTRIBUTOR-S MODULE
GCONTR = CONTRIBUTOR-S MODULE
HOLR27 = CONTRIBUTOR-S MODULE
HOLR9A = CONTRIBUTOR-S MODULE
HOLR9 = CONTRIBUTOR-S MODULE
H9DFEQ = CONTRIBUTOR-S MODULE
IGET = CONTRIBUTOR-S MODULE
INTEQ5 = CONTRIBUTOR-S MODULE
INTUNK = ITPACK MODULE

```
ITCJCG  =  ITPACK MODULE
ITERM   =  ITPACK MODULE
ITJCG   =  ITPACK MODULE
ITJSI.  =  ITPACK MODULE
ITRSCG  =  ITPACK MODULE
ITRSSI  =  ITPACK MODULE
ITSRCG  =  ITPACK MODULE
ITSRPA  =  ITPACK MODULE
ITSRSI  =  ITPACK MODULE
JCG     =  ITPACK MODULE
JSI     =  ITPACK MODULE
KFOLD   =  CONTRIBUTOR-S MODULE
LCJCG   =  ITPACK MODULE
LIN5PT  =  ITPACK MODULE
LJCG    =  ITPACK MODULE
LJSI    =  ITPACK MODULE
LRBNDX  =  ITPACK MODULE
LRSCG   =  ITPACK MODULE
LRSSI   =  ITPACK MODULE
LSSORB  =  ITPACK MODULE
LSSORF  =  ITPACK MODULE
LSSRCG  =  ITPACK MODULE
LSSRKB  =  ITPACK MODULE
LSSRKF  =  ITPACK MODULE
LSSRPA  =  ITPACK MODULE
LSSRSI  =  ITPACK MODULE
LVBUPC  =  ITPACK MODULE
LVBU    =  ITPACK MODULE
MARK1   =  CONTRIBUTOR-S MODULE
NATNDX  =  ITPACK MODULE
NATORD  =  CONTRIBUTOR-S MODULE
NEARST  =  CONTRIBUTOR-S MODULE
NEG     =  CONTRIBUTOR-S MODULE
NSPIV1  =  CONTRIBUTOR-S MODULE
NSPIV   =  CONTRIBUTOR-S MODULE
ODDRD   =  CONTRIBUTOR-S MODULE
OMEG    =  ITPACK MODULE
OMGTST  =  ITPACK MODULE
PARAM   =  ITPACK MODULE
PARCG   =  ITPACK MODULE
PARRS   =  ITPACK MODULE
PARSRC  =  ITPACK MODULE
PCUBED  =  CONTRIBUTOR-S MODULE
PHI     =  ITPACK MODULE
P3C1CH  =  CONTRIBUTOR-S MODULE
P3C1CO  =  CONTRIBUTOR-S MODULE
P3C1GH  =  CONTRIBUTOR-S MODULE
QUADRD  =  CONTRIBUTOR-S MODULE
QUADRT  =  CONTRIBUTOR-S MODULE
RAYQT   =  ITPACK MODULE
RBNDX   =  ITPACK MODULE
READIN  =  ITPACK MODULE
REGPLT  =  CONTRIBUTOR-S MODULE
RESID   =  CONTRIBUTOR-S MODULE
RGHTSD  =  CONTRIBUTOR-S MODULE
RSCG    =  ITPACK MODULE
RSSI    =  ITPACK MODULE
SELF5   =  CONTRIBUTOR-S MODULE
SETF    =  CONTRIBUTOR-S MODULE
SNGSOL  =  CONTRIBUTOR-S MODULE
SOLUT   =  CONTRIBUTOR-S MODULE
SOLVER  =  ITPACK MODULE
SPARPR  =  CONTRIBUTOR-S MODULE
SPARSS  =  CONTRIBUTOR-S MODULE
SPINDX  =  ITPACK MODULE
SSORB   =  ITPACK MODULE
SSORCG  =  ITPACK MODULE
SSORF   =  ITPACK MODULE
SSORKB  =  ITPACK MODULE
SSORKF  =  ITPACK MODULE
SSORPA  =  ITPACK MODULE
SSORSI  =  ITPACK MODULE
STAR5   =  CONTRIBUTOR-S MODULE
STOREX  =  CONTRIBUTOR-S MODULE
SUM1    =  ITPACK MODULE
SUM2    =  ITPACK MODULE
SUM3    =  ITPACK MODULE
SYMBND  =  CONTRIBUTOR-S MODULE
SYM5PT  =  ITPACK MODULE
TABLER  =  CONTRIBUTOR-S MODULE
TFOLD   =  CONTRIBUTOR-S MODULE
TRUEEP  =  CONTRIBUTOR-S MODULE
TRUE    =  CONTRIBUTOR-S MODULE
TSTCHG  =  ITPACK MODULE
TSTSTP  =  ITPACK MODULE
UTDV    =  ITPACK MODULE
UTSTSV  =  ITPACK MODULE
UUNK5   =  CONTRIBUTOR-S MODULE
UVAL5   =  CONTRIBUTOR-S MODULE
VBUPC   =  ITPACK MODULE
VBU     =  ITPACK MODULE
VCOPY   =  ITPACK MODULE
VUMW    =  ITPACK MODULE
ZERO    =  CONTRIBUTOR-S MODULE
```

APPENDIX THREE

EXAMPLE OF MODULE INFORMATION
REQUIREMENT FOR USER GUIDE

MODULE NAME: P3C1-COLLOCATION

AUTHOR/DATE: Elias Houstis, 5/20/77

INITIAL/FINAL INTERFACE: EQUATION FORMATION-EQUATION INDEXING

MODULE FUNCTION: Discretizes a General Operator with Dirichlet, Neumann or Mixed Boundary Conditions.

RESTRICTIONS ON USE:

Two dimensions

Grid of size at least 3x3

METHOD DESCRIPTION: Collocation based on bicubic Hermite elements. Interpolates the nonhomogeneous boundary conditions. Elements satisfy exactly Dirichlet or Neumann homogeneous boundary conditions.

PARAMETERS: None

KEYWORDS THAT AFFECT MODULE: Homogeneous boundary conditions

STORAGE AND TIMING ESTIMATES:

Nonhomogeneous boundary conditions: 4*NGRIDX*NGRIDY equations are generated with 16 coefficients per equation

Homogeneous boundary conditions: 4*(NGRIDX-1)*(NGRIDY-1) equations are generated with 16 coefficients per equation

COMPATIBLE WITH ELLPACK VERSIONS: 0, Sept 77 , March 78, Sept 78