Department of Computer Science Technical Reports

Department of Computer Science

1982

Solving Elliptic Problems Using ELLPACK Part 1: ELLPACK User's Guide; Part 2: The Problem Solving Modules

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

Ronald F. Boisvert

Report Number:
81-414

SOLVING ELLIPTIC PROBLEMS USING ELLPACK

Part 1: ELLPACK USER'S GUIDE
Part 2: THE PROBLEM SOLVING MODULES

John R. Rice
Computer Science Department
Purdue University

Ronald F. Boisvert
Scientific Computing Division
National Bureau of Standards

September 24, 1982

CSD-TR 414

ABSTRACT

This report describes how to use the ELLPACK system and language for solving elliptic problems. ELLPACK provides many facilities for solving two dimensional, linear elliptic partial differential equations on rectangular domains, several facilities for non-rectangular domains and for three dimensional rectangular domains. The system allows a user to attack non-linear problems by constructing various iterations of linear methods.

The current revision is a draft for part of the final documentation of the ELLPACK system. Corrections and suggestions for improvements are welcomed.

CONTENTS

REFERENCES

PREFACE

The ELLPACK system is the outgrowth of a cooperative project to study methods and software for elliptic problems. This project was coordinated by John R. Rice of Purdue University; the principal members of the project were

Randy Bank	University of Texas
Garrett Birkhoff	Harvard University
Ronald Boisvert	National Bureau of Standards
Stanley Eisenstat	Yale University
William Gordon	Drexel University
Elias Houstis	University of South Carolina
David Kincaid	University of Texas
Robert Lynch	Purdue University
Donald Rose	Bell Telephone Laboratories
Martin Schultz	Yale University
Andrew Sherman	Exxon Research
David Young	University of Texas

Substantial contributions of software were made by many others: Carl de Boor, John Brophy, Wayne Dyksen, Roger Grimes, Hartmut Foerster, LINPACK, William Mitchell, W. Proskurowski, John Respess Granville Sewell, Van Snyder, Paul Swarztrauber, Roland Sweet, Linda Thiel, William Ward and Alan Weiser. This project has received support from the National Science Foundation, the Department of Energy and the Office of Naval Research as well as from the participants' institutions.

ELLPACK was originally developed as a research tool to evaluate and compare mathematical software for solving elliptic problems. The idea was to create a system where individuals can contribute software modules which either completely or partially solve an elliptic problem. Those modules that partially solve the problems (e.g. discretize it) are combined with other modules to complete the solution. With all the software operating in the same environment one can make a performance evaluation of the modules. Several studies of this type have been made and Part 3 of this book presents simple examples of performance evaluation.

Considerable effort was put into making ELLPACK easy to use and to augment. The ELLPACK system presented in this book is useful not only for research into the performance of numerical methods and software, but also for education and actual problem solving. Standard elliptic problems of moderate difficulty can be stated and solved in a direct, simple manner. Many more complex problems, including nonlinear, time dependent and simultaneous equations, can be solved using more advanced ELLPACK facilities.

A. The ELLPACK Project

B. Mathematical Preliminaries (not included)

C. Numerical Methods Preliminaries (not included)

CHAPTER ONE: GENERAL DESCRIPTION AND A SIMPLE PROGRAM

1.A GENERAL DESCRIPTION

ELLPACK is a computer programming system for elliptic boundary value problems. The problems addressed include linear variable-coefficient elliptic equations of the form

$$au_{xx} + cu_{yy} + du_x + eu_y + fu = g$$

or, in self-adjoint form,

$$(pu_x)_x + (qu_y)_y + fu = g$$

defined on general two-dimensional domains, and their three-dimensional counterparts defined on rectangular boxes. For two-dimensional problems, boundary conditions may take the form

$$\alpha u_x + \beta u_y + \delta u = \varphi ,$$

where α, β, δ and φ are functions of x and y. The three-dimensional case is similar. Periodic boundary conditions are also admitted when the domains are rectangular. In addition, ELLPACK is organized so that it is possible to set up iterations to solve nonlinear problems (i.e. a, c, d, e, f, g, p, q, α, β, δ, φ are also functions of u, u_x etc.).

ELLPACK users specify the problem they wish to solve in an **ELLPACK program** written in a simple user-oriented **ELLPACK language.** The **ELLPACK system** processes this program by first translating it to a FORTRAN source program called the **ELLPACK control program**; this program is then compiled and linked to a precompiled **ELLPACK module library.** Finally, the program is executed, producing a solution to the problem. The process is illustrated in Figure 1.1.

The ELLPACK language is an extension of Fortran, and ordinary Fortran code can be mixed with ELLPACK statements. Elliptic equations, domains, and

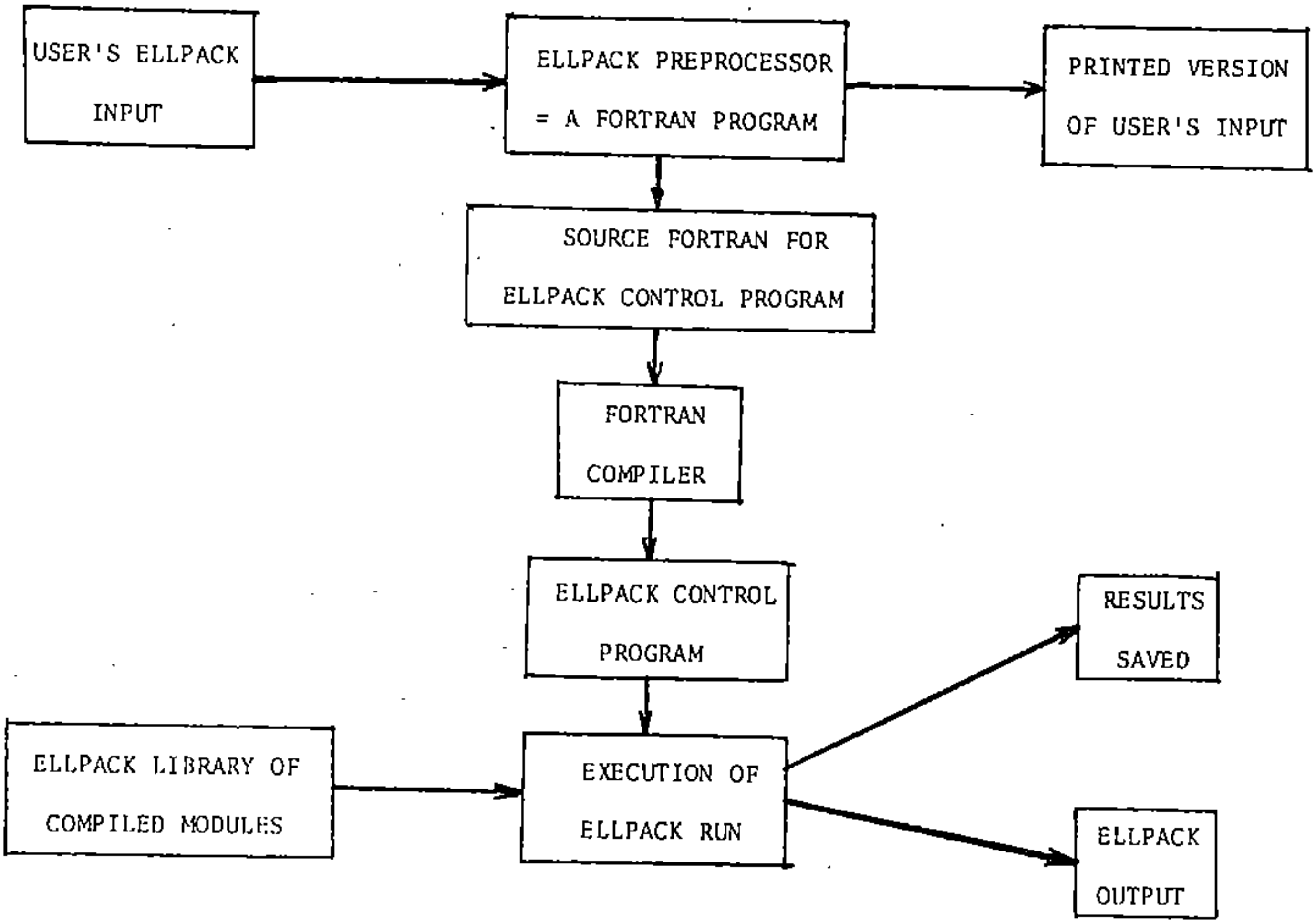
USER'S ELLPACK INPUT
ELLPACK PREPROCESSOR = A FORTRAN PROGRAM
PRINTED VERSION OF USER'S INPUT
SOURCE FORTRAN FOR ELLPACK CONTROL PROGRAM
FORTRAN COMPILER
ELLPACK CONTROL PROGRAM
RESULTS SAVED
ELLPACK LIBRARY OF COMPILED MODULES
EXECUTION OF ELLPACK RUN
ELLPACK OUTPUT

Figure 1.1. Schematic diagram of the processing of an ELLPACK run.

boundary conditions can be declared in this language, and powerful statements are available to help users get from the specified problem to useable output. These statements invoke modules in the ELLPACK module library. There are five basic types of modules:

Discretization. Replace the partial differential equation and boundary conditions by an approximate, finite system of linear algebraic equations.

Indexing. The equations and unknowns of the discrete system are reordered to facilitate solving the system.

Solution. The system of equations is solved.

Triple. Discretization and solution are performed as a single step.

Output. The approximate solution is tabulated or plotted.

One specifies the numerical method to be used by invoking, in turn, a discretization module, an indexing module, and a solution module (or a single triple module). In the ELLPACK language this is done by simply giving their names.

A large number of modules are available in ELLPACK for each stage of the computation. For example, discretization by various types of finite difference and finite element methods are possible, as well as solution of algebraic equations by both iterative and direct methods. Detailed descriptions of the available modules are given in Part 2. This easy access to a large repertoire of numerical methods makes ELLPACK useful in comparing solutions obtained by vastly different methods, as well as a "pilot plant" for large scale application problems.

During execution, ELLPACK modules communicate through fixed predefined collections of variables called **interfaces.** This process is illustrated in Figure 1.2 and described in detail in Part 4, where a complete specification of

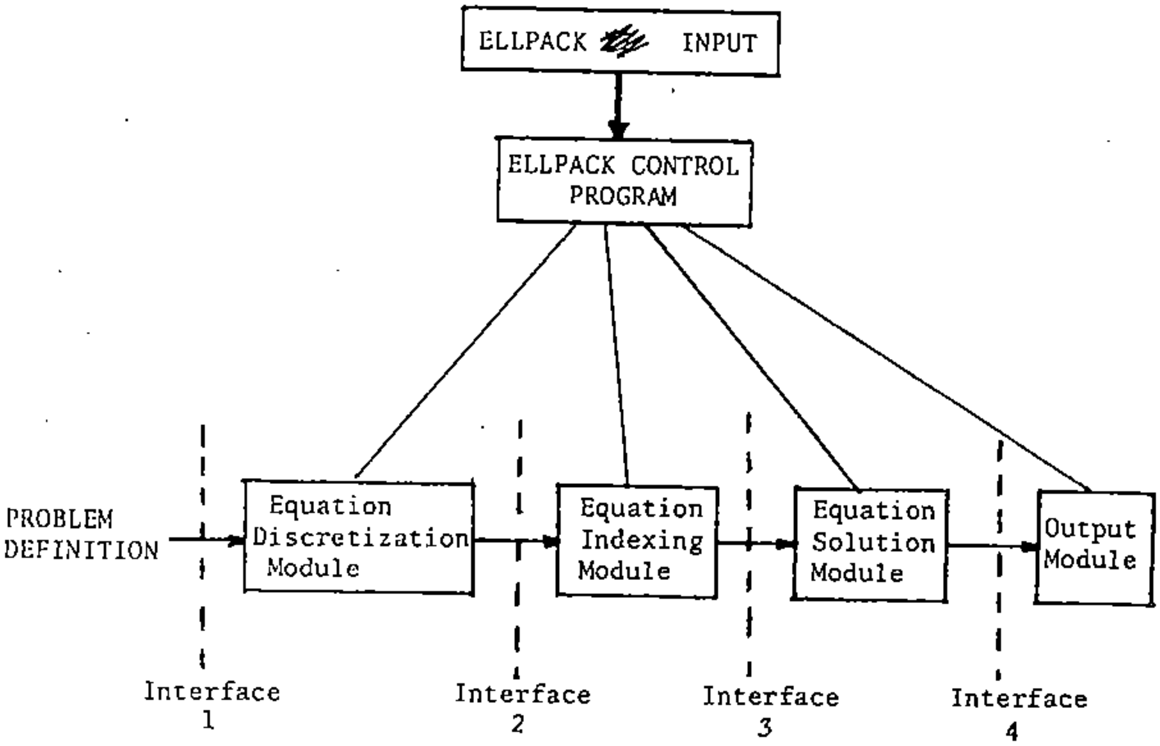

Figure 1.2. Basic organization of an ELLPACK computation. The user specifies the modules to be used and more than one combination may be used on a single ELLPACK run.

how to add new modules to the ELLPACK system is given.

The final essential ingredient in the solution of an elliptic problem with ELLPACK is the specification of a rectangular grid to cover the domain. When this grid is made finer the approximations used by discretization modules are more accurate (within the constraints of machine arithmetic), but computer time and memory requirements also increase (i.e. there are more algebraic equations generated). When a non rectangular domain is specified in ELLPACK, the **domain processor** is invoked. It sets up tables which relate the rectangular grid to the domain in a way useful to discretization and triple modules.

1.B A SIMPLE ELLPACK PROGRAM

We show a very simple elliptic problem and an ELLPACK program which generates an approximate solution. A table of this solution is printed and a contour plot is produced;

Elliptic Problem

The partial differential equation $Lu = f$ is

$$u_{xx} + u_{yy} + 3u_x - 4u = exp(x+y)sin(\pi x),$$

the domain R is the rectangle $0<x<1$, $-1<y<2$ and the boundary conditions $Mu = g$ are

$$\begin{array}{ll} u = 0, & x=0,\ -1<y<2 \\ u = x, & 0<x<1,\ y=2 \\ u = \frac{y}{2}, & x=1,\ -1<y<2 \\ u = \sin(\pi x) - \frac{x}{2}, & 0<x<1,\ y=-1 \end{array}$$

The ordinary finite difference approximation (5-POINT STAR) is used to discretize the problem at points of a square grid with spacing 1/5. The resulting linear system is solved with ordinary Gauss elimination for band matrices.

```
*         **************************************************
*         *                                                *
*         *  EXAMPLE ELLPACK PROGRAM 1.B1                  *
*         *                                                *
*         **************************************************
*
OPTIONS.        TIME  $  MEMORY
*
EQUATION.
                UXX  +  UYY  +  3.0*UX  -  4.0*U  =  EXP(X+Y)*SIN(PI*X)
*
BOUNDARY.       U = 0.0                   ON  X = 0.0
                U = SIN(PI*X) - X/2.0     ON  Y =-1.0
                U = Y/2.0                 ON  X = 1.0
                U = X                     ON  Y = 2.0
*
GRID.           6 X POINTS  $  6 Y POINTS
*
DISCRETIZATION. 5 POINT STAR
INDEXING.       NATURAL
SOLUTION.       LINPACK BAND
*
OUTPUT.         TABLE (U)  $  PLOT (U)
*
END.

    APPROXIMATE MEMORY REQUIREMENTS

    WORKSPACE            1875      GRID LINES           13
    LINEAR EQNS           576      UNKNOWNS             36
    INTERPOLATION         141      DOMAIN INFO           0
    AMATRX,BVECTR         504      TOTAL MEMORY       3145


        SYMBOL TABLE INPUT TIME  2.52 SECONDS
        PROGRAM PROCESSING TIME   .90 SECONDS
           TEMPLATE OUTPUT TIME  1.90 SECONDS
                     TOTAL TIME  5.32 SECONDS
```

Output of ELLPACK run:

```
-----------------------
 DISCRETIZATION  MODULE
-----------------------

   5-POINT  STAR

   DOMAIN                              RECTANGLE
   X INTERVAL              .000E+00,   .100E+01
   Y INTERVAL             -.100E+01,   .200E+01
   DISCRETIZATION                        UNIFORM
   GRID                                  6 X  6
   HX                                   .200E+00
   HY                                   .600E+00
   B.C.S ON PIECES 1,2,3,4               1,1,1,1
   OUTPUT LEVEL                                1
   NUMBER OF EQUATIONS                        16
   MAX NO. OF UNKNOWNS PER EQ.                 5
   EXECUTION SUCCESSFUL


----------------
 INDEXING  MODULE
----------------
```

```
  N A T U R A L

  NUMBER OF EQUATIONS                        16
  EQUATIONS/UNKNOWNS NUMBERED
      IN ORDER GENERATED
  EXECUTION SUCCESSFUL


----------------
SOLUTION  MODULE
----------------

  L I N P A C K    B A N D

  NUMBER OF ROWS                             13
  NUMBER OF COLUMNS                          16
  NUMBER OF LOWER CO-DIAGONALS                4
  NUMBER OF UPPER CO-DIAGONALS                4
  LINPACK BAND GIVES 2 TIMINGS
      SETUP TIME AND SOLUTION TIME
  EXECUTION SUCCESSFUL


-------------------
ELLPACK  77  OUTPUT
-------------------


        +++++++++++++++++++++++++++++++++++++++++++++
        +                                           +
        +    TABLE OF U      ON   6 X   6  GRID     +
        +                                           +
        +++++++++++++++++++++++++++++++++++++++++++++


   X-ABSCISSAE ARE
   ---------------
    .000000E+00     .200000E+00     .400000E+00     .600000E+00
    .800000E+00     .100000E+01

      Y =  .200000E+01
      ----------------
    .000000E+00     .200000E+00     .400000E+00     .600000E+00
    .800000E+00     .100000E+01

      Y =  .140000E+01
      ----------------
    .000000E+00    -.689090E-01    -.478790E-01     .978828E-01
    .368059E+00     .700000E+00

      Y =  .800000E+00
      ----------------
    .000000E+00    -.691488E-01    -.663472E-01     .238169E-01
    .194251E+00     .400000E+00

      Y =  .200000E+00
      ----------------
    .000000E+00    -.621819E-01    -.808831E-01    -.535813E-01
    .147178E-01     .100000E+00

      Y = -.400000E+00
      ----------------
    .000000E+00    -.202565E-02    -.252810E-01    -.732527E-01
   -.135945E+00    -.200000E+00

      Y = -.100000E+01
      ----------------
    .000000E+00     .487785E+00     .751057E+00     .651056E+00
    .187785E+00    -.500000E+00
```

```
---------------------
 ELLPACK  77  OUTPUT
---------------------


      +++++++++++++++++++++++++
      +                       +
      +    EXECUTION  TIMES   +
      +                       +
      +++++++++++++++++++++++++


    MODULE NAME                   SECONDS
    ----------------------------------------

    5-POINT STAR                      .12
    NATURAL                           .02
    LINPACK BAND SETUP                .03
    LINPACK BAND                      .03
    TABLE                             .38
    PLOT                             6.37
    TOTAL TIME                       7.08
```

This program consists of several **segments** whose names (EQUATION, BOUNDARY, and so on) begin in column 1 of a line, the rest is written in free format (excluding column 1). The dollar sign is a separator to allow more than one item on one line in a segment. Parts of the program include Fortran expressions (+3., EXP(X+Y)*SIN(PI*X), etc.) which must follow the rules of Fortran. Lines beginning with * are comments.

This example is the simplest case of an ELLPACK program: one defines the elliptic problem in the EQUATION and BOUNDARY segments, OPTIONS are chosen, a rectangular grid is defined in the GRID segment, the solution method is specified in the DISCRETIZATION, INDEXING and SOLUTION segments and the desired output is specified in the OUTPUT segment. Every ELLPACK program ends with END.

The ELLPACK preprocessor lists the program with an identifying heading. It also prints the memory estimates as requested in the OPTIONS segment along with its execution time. Each ELLPACK module prints a simple summary message. The output segment contains two requests, one is a table of the solution on the grid and the other is a contour plot produced by some graphics device.

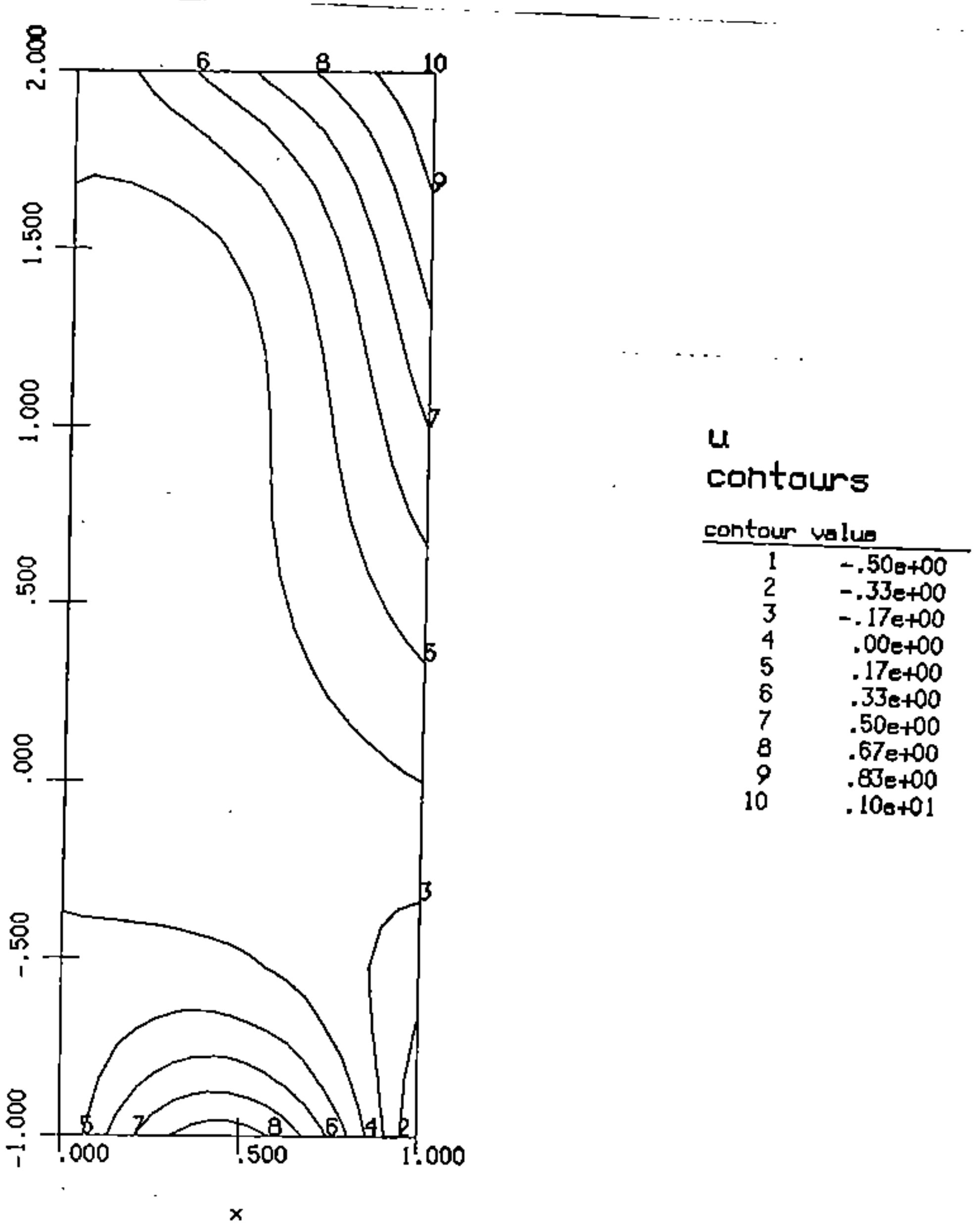

Figure 1.3. The contour plot produced by PLOT(U) in the example ELLPACK 1B1 program. This plot is made with an electrostatic printer.

The graphics connected to ELLPACK will vary from installation to installation.

1.C ORGANIZATION OF THE BOOK

The basic features of the ELLPACK language are presented in the next chapter, then three more examples are presented in detail in Chapter 3 (one solves the same problem with two different choices of methods, another illustrates non-rectangular geometry and the third shows how Fortran can be interspersed with ELLPACK statements.) Chapters 4 and 5 describe and illustrate more advanced features of ELLPACK. There are a number of examples of advanced applications in these two chapters. Part 2 (Chapters 6 through 9) contains summary descriptions of the over 40 modules available and an overview of the ITPACK and YALEPACK software included in ELLPACK. Part 3 (Chapters 10-12) presents a basic performance evaluation of many of the ELLPACK modules. The objective is to give the reader some feel for the properties of various methods (software modules) and not to present a complete scientific evaluation.

Part 4 (Chapters 13-18) is a **Contributor's Guide**; it provides the information to prepare a new module for the ELLPACK system. The ELLPACK system is designed so that new modules can be easily added (and corresponding additions made to the language). There is useful information for those who wish to attempt advanced ELLPACK applications, otherwise this and the following Part 5 are not relevant to the use of ELLPACK. Part 5 (Chapters 19 and 20) is an **Installation Guide**; it provides detailed information on how to install ELLPACK and to make modifications to it. The basic ELLPACK system can be installed without much difficulty; one needs to know how to manipulate files and to create a library from a set of Fortran programs. Tailoring the ELLPACK system is more complicated and while it does not require specific expertise of a system programmer, one is more likely to have seen the kinds of things that have to be done.

The Appendices contain reference material, there are brief summaries of the **PG system** and the **TOOLPACK template processor** which are used to create the ELLPACK system. There is the **PDE population** a set of over 60 linear elliptic partial differential equations on two dimensional rectangular domains. These can be used as a problem population for a systematic performance evaluation.

CHAPTER 2. THE ELLPACK LANGUAGE

2.A GENERAL ORGANIZATION OF AN ELLPACK PROGRAM.

The ELLPACK program should be interpreted as the main program of a Fortran job. The basic blocks of statements in an ELLPACK program are **segments.** The segments that define the elliptic problem and options are like declarations: they must come first and they are not executed. The other segments (except END) are executed and the flow of the computation is controlled by placing them in the proper sequence. Ordinary Fortran statements may be interspersed among the executable segments, and there is also a facility to specify Fortran subprograms (but not ELLPACK subprograms).

A brief summary of the segments is given in groups.

Group 1 Segments define the elliptic problem. They must appear before any from Group 2 and, except for GRID, appear exactly once.

EQUATION. Specifies the partial differential equation.

BOUNDARY. Specifies the domain and boundary conditions.

GRID. Specifies a set of vertical and horizontal grid lines. GRID can appear more than once to change the grid size provided that MAXGRID is set in an OPTION segment before the first GRID segment.

HOLE. Defines a hole in the domain and associated boundary conditions. This segment can appear more than once if several holes are present. It must follow the BOUNDARY segment.

ARC. Defines an interface or slit in the domain on which additional conditions are prescribed. Its use is governed by the same rules as the HOLE segment.

Blanks are not allowed in these or any other segment names. Segment names may be abbreviated by two or more of their leading characters.

Group 2 Segments specify the executable ELLPACK modules and may appear more than once. A specific ordering is usually required; e.g., DISCRETIZATION, INDEXING, SOLUTION, OUTPUT or TRIPLE, OUTPUT.

DISCRETIZATION. Specifies a module to define a discrete approximation to the elliptic problem; this generates a system of linear algebraic equations. (This is the first phase of an ELLPACK solution algorithm.)

INDEXING. Specifies a module to reorder the linear equations and the unknowns. (This is the second phase of an ELLPACK solution algorithm.)

SOLUTION. Specifies a module which solves the linear equations. (This is the final phase of an ELLPACK solution algorithm.)

TRIPLE. Specifies a combination method which includes discretization, indexing and solutions all in one module.

PROCEDURE. Specifies various other optional actions in solving or analyzing the problem.

OUTPUT. Selects desired ELLPACK-generated output (printed and graphical).

Group 3 Segments may appear anywhere in the program and as many times as desired.

* Specifies a comment.

OPTIONS. Specifies which of various options are desired.

FORTRAN. Specifies that the statements which follow are user supplied executable Fortran statements.

(blank line) Allowed at any point

Group 4 Segments specify various information and can appear at most once.

DECLARATIONS. Provides Fortran declarations for the user provided executable Fortran statements. Must appear at the beginning of the program.

SUBPROGRAMS. Specifies that a set of Fortran subprograms (FUNCTION or SUBROUTINE) follows. This segment must go just before the END segment.

GLOBAL. Gives declarations (primarily COMMON blocks) that are placed within Fortran programs generated by ELLPACK to define the elliptic problem. Must appear at the beginning of the program.

END. Specifies the end of the ELLPACK program.

Two or more letters of the beginning of a segment name form an acceptable abbreviation. All segment names and their abbreviations must end with a period. Each DISCRETIZATION, INDEXING, SOLUTION, EQUATION, TRIPLE and PROCEDURE segment must be on a single line. The line may, however, be continued by putting a period in column 1. No segment can be longer than 1000 characters. The OPTIONS, OUTPUT, BOUNDARY, GRID, HOLE and ARC segments may use several lines and the separator $ may be used to place several parts of these segments on one line. If these segments are broken in the middle of a word or expression, then the continuation convention (period in column 1) must be used. The segments FORTRAN, SUBPROGRAMS, DECLARATIONS and GLOBAL start with the segment name on a separate line followed by lines of Fortran code.

The independent variables are denoted by X, Y, and Z (X and Y for two-dimensional problems). The dependent variable is denoted by U, its first derivatives u_x, u_y and u_z by UX, UY, and UZ, and the second derivatives UXX, UYY, UXY, and so on. These names are reserved in ELLPACK and Fortran variables with these names cannot be used safely. Once the PDE is solved the functions U(X,Y), UX(X,Y), etc. (U(X,Y,Z), UX(X,Y,Z), etc. in three dimensions) become defined and may be used as ordinary Fortran functions. The complete set of **reserved names in ELLPACK is:**

X, Y, Z U, UX, UXX, UY, UYY, UYX, UXY, UZ, UZZ, UZX, UXZ, UZY, UYZ, TRUE, ERROR, RESIDU, ON, FOR, TO, LINE, PI plus any 6 character name starting with C, I, L, Q or R followed by a digit. The words ON, FOR, TO and LINE must actually only be avoided in Fortran functions of the BOUNDARY segment. The variable PI is set to the mathematical constant π and can be used anywhere in the ELLPACK program. The six character names are internal Fortran variables for ELLPACK; their use would create a name conflict. The meanings of the initial characters are

C	Common blocks
I	Integers
L	Logical
Q	Subprograms
R	Real (or double precision)

2.B SEGMENTS WHICH DEFINE THE PROBLEM AND GRID (GROUP 1)

We describe the rules (syntax) for defining the PDE problem and associated rectangular grid. The notation <word> is used to specify an item that is to be provided or defined later. Thus

<coef> u = <right side>

can represent $(x^2+1)\,u = x\ cos(x)$ with coef $=x^2+1$ and right side $= x\ cos(x)$.

EQUATION. segment

The EQUATION segment specifies the partial differential equation to be solved. In the definition of the equation, the dependent variable and its derivatives are denoted by U, UX, UXX, etc. The equation is specified in the form

<operator> = <right side>

where <operator> is a list of terms of the form

<coefficient> * <derivative>

The terms <coefficient> and <right side> denote any valid Fortran **real arithmetic expressions** as well as the separators + or -, and <derivative> denotes one of U, UX, UY, UZ, and so on. If the coefficient of a derivative is zero, then the associated term need not appear.

Some examples of the EQUATION segment are given below.

```
*      LAPLACE'S EQUATION
*
EQUATION.  UXX + UYY = 0.

*      AN EQUATION WITH CONSTANT COEFFICIENTS
*
EQUATION.  -4.*UXX + .377*UXX - 3.*PI*UYY + 3.E+4*UX = SIN(X+COS(X*Y))

* THE COEFFICIENTS OF UYY AND U ARE GIVEN AS FORTRAN FUNCTIONS.
* THESE ARE SUPPLIED BY THE USER IN THE SUBPROGRAMS SEGMENT.

EQUATION.  (X**2 + Y**2 + 16.)*UXX + VALUYY(X,Y)*UYY
.          -2.234E- 3*ATAN2(Y,X) UX + 1.4*UY - VALU(X,Y)*U = 0.
```

There is a special ELLPACK form for **self-adjoint equations** which are written in the form

$$(p(x,y)u_x)_x + (q(x,y)u_y)_y + r(x,y)u = f(x,y)$$

It is

```
EQ.     (P(X,Y)*UX)X + (Q(X,Y)*UY)Y + R(X,Y)*U = G(X,Y)
```

The functions P, Q, R and G may be replaced by any Fortran expressions. There is an alternate way to indicate a self-adjoint equation by using the OPTIONS segment as follows:

```
*            SELF-ADJOINT, ALTERNATE FORM
OPTION.      SELF-ADJOINT = .TRUE.
EQUA.  P(X,Y)*UXX + Q(X,Y)*UYY + R(X,Y)*U = G(X,Y)
```

Note that several modules apply only to PDEs written in self-adjoint form.

BOUNDARY. segment

The BOUNDARY segment specifies the boundary of the domain *R* and the boundary conditions on them. We first describe general two-dimensional domains in ELLPACK; the special facilities for the simpler cases of rectangular two- and three-dimensional domains are described after that. The boundary is broken up into a series of pieces which must join together in sequence. A piece and condition are specified by

<condition> ON <piece>

where <condition> is one of the following:

PERIODIC or

<expression>*UX + <expression>*UY + <expression>*U = <expression>

where <expression> is a legal Fortran expression. The three terms on the left can be in any order, and any term may be omitted if its coefficient expression is zero. If the <condition> preceeding ON is omitted, then the preceeding <condition> is used as the default condition.

Periodic boundary conditions may only be applied in the case of rectangular domains. If PERIODIC is specified on one side, then it must also be specified on the opposite side. Any of the other usual types of boundary conditions can be specified using the second form. For instance, a Dirichlet condition is specified as

U = F(X,Y)

and a Neumann condition as

A(X,Y)*UX + B(X,Y)*UY = F(X,Y)

where (X,Y) is a point on the boundary and (A(X,Y), B(X,Y)) is the unit vector normal to the boundary (pointing outward). The latter reduces to ± UX=F(Y) or

± UY=F(X) for rectangular domains.

A non-rectangular two dimensional domain is specified as a sequence of parameterized sides. The general form of <piece> is

X = <expression>, Y = <expression> FOR <parameter> = <a> TO <b>

where

<parameter> is a real Fortran variable that parameterizes the side

<expression> is a Fortran expression in the parameter

<a>, <b> are Fortran expressions that evaluate to constants which determine the initial and final value of the parameter.

The pieces are assumed to be given in counter-clockwise order, (this may be overridden by putting CLOCKWISE = .TRUE. in an OPTION segment). Each piece starts on a new line unless the $ separator is used. The parameter must increase from <a> to <b>. **It is essential that the parameterization be of ordinary size and not vary erratically along the boundary.** The continuity of joining the pieces is checked and the joining must be done accurately. Two simple examples of non-rectangular boundary and boundary condition specification follow:

```
*                CIRCULAR  DISK WITH CENTER 1,1
BOUND. U = 0.0 ON X = 1.-COS(PI*THETA), Y = 1.-SIN(PI*THETA)
:                                         FOR THETA = 0. TO 2.
*
*        QUARTER ANNULUS
BOUNDARY.

 U=100.        ON X=SIN(PI*T), Y=COS(PI*T)         FOR T=0. TO 0.5
 U=100.*(2.-X) ON X=R,  Y=0.                       FOR R=1. TO 2.0
 U=0.0         ON X=2.*COS(PI*T),Y=2.*SIN(PI*T) FOR T=0. TO 0.5
 U=100.*(2.-Y) ON X=0.0, Y=2.-R                    FOR R=0. TO 1.0
```

The reserved words ON, FOR, LINE and TO cannot have blanks in them and **must have blanks on both sides of them.**

There is a special simple form for **straight line pieces** of the boundary. In this case <piece> appears as:

LINE <x-constant>, <y-constant> TO <x-constant>, <y-constant>

where <x-constant>, <y-constant) are the coordinates of the end point of the piece; they may be any Fortran expression that evaluates to a constant. Straight line sides may be connected by the following multiple side form:

```
<condition> ON LINE <x>, <y> TO <x>, <y>
<condition>                  TO <x>, <y>
    ...                          ...
<condition>                  TO <x>, <y>
```

The boundary condition <condition> may be omitted if it is the same as for the preceeding pieces (for both straight line pieces or parameterized pieces). Several groups of TO <x>, <y> can be placed on one line as long as the same boundary condition holds, as for example

U=1.0 LINE <x1>,<y1> TO <x2>,<y2> TO <x3>,<y3> TO · · · TO <xK>,<yK>

The complex example in Figure 2.1 of a non-rectangular domain specification follows:

```
*               FOUR SIDED, NON-RECTANGULAR DOMAIN
OPTION.         CLOCKWISE = .TRUE.
BOUNDARY.

     U = 0.0 ON LINE 4.,4. TO 1., 4.
                           TO 1., 0.5
     U = (X-4.)*(Y-.5)     TO 4.,-0.5
           ON X = 4.+.1*P*(P-4.5)**2, Y=-.5+P FOR P=0. TO 4.5
```

This example shows how omitting the boundary condition specifies it to be the previous one and how the LINE specifications continues from piece to piece.

A second complicated example follows:

```
*          SIX SIDED REGION WITH 3 STRAIGHT SIDES

U=0.0           ON X=-T, Y=(T-1.)**2       FOR T = 1. TO 2.
                ON X=(P-1)**2-2, Y=P       FOR P = 1. TO 2.
U=(2.*X-Y)**2   ON LINE -1.,2. TO .5,2. TO 1.,1. TO 0.,0.
UX-3.0*U=.5     ON X=-SQRT(PHI), Y=SIN(PI*PHI)/5. FOR PHI = 0.0 TO 1.0
```

There is a special abbreviated form for **rectangular domains** (in two or

Figure 2.1. A nonrectangular domain with its parameterization and boundary conditions given.

three dimensions) where <piece> is of the form

<variable> = <constant>

Here <variable> is one of X,Y,Z and <constant> is a Fortran expression that evaluates to a constant. Two examples of defining a two dimensional domain and its boundary conditions follow:

```
*         ABBREVIATED BOUNDARY FORM FOR A RECTANGLE
BOUND.
     U = 1.0                       ON X = 0.0
     U + X*UX = (X+Y)*EXP(Y)       ON X = 1.0
     UY = 0.0                      ON Y = 0.0
     U = EXP(X)                    ON Y = 1.0
*         BOUNDARY CONDITION CARRIED FORWARD FROM PIECE TO PIECE
BO.  U = 0.0                       ON Y = 0.0
                                   ON Y = PI/2.
                                   ON X = 0.0
     U = EXP(1.)*SIN(2.*PI*Y)      ON X = EXP(1.0)
```

The preceding example can be written in a more compact form using the $ separator as follows

```
BOUNDARY.
  U = 0.0 ON Y = 0.0 $ ON Y = PI/2. $ ON X= 0.0
  U = EXP(1.)*SIN(2.*PI*Y) ON X = EXP(1.0)
```

The extension of this notation to three dimensional rectangles is straight forward; six rather than four sides and conditions are required and boundary conditions can include U, UX, UY and UZ terms.

GRID. segment

The GRID. segment defines a rectangular grid placed over the domain. The general form of the segment is a set of terms:

<n> <variable>   POINTS   <point list>

where

<n> = number of points; must be constant unless MAXGRID option is used
<variable> = variable involved (one of X, Y or Z),
<point list> = list of grid coordinates in increasing order.

These terms must be on separate lines or separated by a $. For two dimensional domains there must be one set of points specified for X and another for Y. In three dimensions there must also be a specification for Z. If the following grid is specified

$$n_1 \text{ X POINTS } x_1, x_2, \ldots, x_{n_1}$$

$$n_2 \text{ Y POINTS } y_1, y_2, \ldots, y_{n_2}$$

then the rectangular grid is made up of the lines

$$x = x_1,\ x = x_2, \ldots, x = x_{n_1}$$

$$y = y_1,\ y = y_2, \ldots, y = y_{n_2}.$$

See Figure 2.2 for an example 4 by 5 grid.

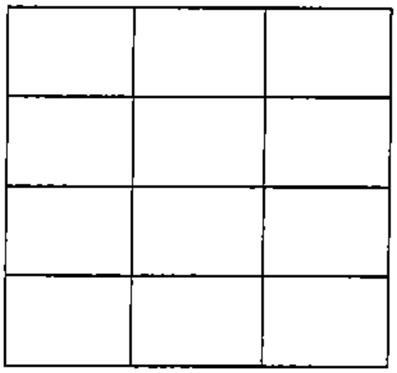

Figure 2.2 The rectangular grid defined by 4 X POINTS $ 5 Y POINTS

For **uniformly spaced grids** <point list> may take the form

<a> TO <b>

where

<a> = initial value of the grid variable
<b> = final value of the grid variable

In this case the points used to discretize the variable are

$$p_i = (i-1) * \frac{(b-a)}{(n-1)} + a, \; i = 1,2, \cdots ,n$$

For **rectangular domains** <a> TO <b> is not used and the initial and final values of the variable correspond to the rectangle.

Some possible combinations are illustrated by the following examples:

```
*
*                  GENERAL NON-UNIFORM CASE

GRID.     7 X POINTS  -1.0, -.8, -.5, 0.0, .5, .8, 1.0
          5 Y POINTS   0.0,  .2,  .5,  .8, 1.0

*                  UNIFORM GRID - FOR NON-RECTANGULAR DOMAINS

GRID.     7 X POINTS -1.0 TO 1.0  $  4 YPOINTS  0.0 TO 1.0

*
*                  UNIFORM GRID - ON A RECTANGLE
*                  FORM VALID ONLY FOR RECTANGULAR DOMAINS

GRID.     7 X-POINTS  $  4 Y-POINTS

*
*                  MIXED CASE FOR 3-D, - ALWAYS RECTANGULAR

GRID.     7 X-POINTS  $  4 YPOINTS  -2.0 TO -1.0
          6 Z POINTS -1.0, -.7, -.25, .25, .7, 1.0
```

2.C SEGMENTS WHICH SPECIFY THE METHODS TO BE USED

We next describe the four segments which specific methods (that is, particular ELLPACK library modules) to be used in solving the problem. These are DISCRETIZATION, INDEXING, SOLUTION and TRIPLE. Most ELLPACK programs have three segments present corresponding to the three steps in approximately solving the problems (See Figure 1.2). However, modules in a TRIPLE segment incorporate all three of these steps.

A summary description of each of these modules is given in Chapter 9. References are given there for further information about the numerical methods used. Note that **each module has restrictions** on its use (such as a self-adjoint equation or a rectangular domain). One must read the descriptions before using the modules. Many modules accept parameters which are placed in parentheses following the module name. These parameters may be specified in any order and default values are provided; setting a parameter value for one use of a module does not affect the default value for later appearances. The **module parameters** are specified as <parameter> = <value>, the <parameter> is an actual variable in the Fortran program generated so one must not use the same name for something else. This is true even if the default parameter value is used. Two simple examples of the use of parameters follow

```
        SOR(OMEGA = 1.8, ITMAX=100, IADAPT=1)
        SOR(ZETA = 1.E-4, OMEGA = 1.88)
```

Module names (unlike segment names and ELLPACK reserved words) may have dashes and blanks in them. All dashes and blanks are removed before the word is to be reorganized. The following are legal

5 POINT STAR, 5-POINT STAR, 5 POINT-STAR, 5-POI NT STAR

8 X POINTS, 8 X-POINTS, 8 XPOINTS, 8-POINTS, 8 XPO INTS

LINPACK BAND LINPACKBAND, LIN-PACK-BAND

Examples of method specifications follow.

```
*
*                  ORDINARY FINITE DIFFERENCES AND GAUSS ELIMINATION
*
DISCRETIZATION.    5-POINT STAR
INDEXING.          AS IS
SOLUTION.          BAND GE
*
*                  A FINITE ELEMENT METHOD WITH ITERATION
*
DISCRETIZATION.    SPLINE GALERKIN(DEGREE=3,SMOOTH=2)
INDEXING.          AS IS
SOLUTION.          SOR
*
*                  A SINGLE MODULE FOR THE PROBLEM
*
TRIPLE.            FFT 9-POINT(ORDER=4)
*
*                  SOLVE THE SAME PROBLEM BY GAUSS ELIMINATION
*                  SPARSE MATRIX METHOD AND TWO ITERATIVE METHODS
*                  THE OUTPUT BETWEEN SOLUTIONS IS OMITTED
*
DIS.               5-POINT STAR
```

```
INDEX.          AS IS
SOLUTION.       SPARSE (NSP=16000)
SOL.            JACOBI CG(ITMAX=200, ZETA=1.E-4)
INDEX.          RED-BLACK
SO.             REDUCED SYSTEM CG(ITMAX=200,ZETA=1.E-4,IADAPT=1)
```

The last code segment illustrates the action of **ELLPACK interfaces.** Once the discretization is made by 5-POINT STAR the resulting information is held fixed at this interface until another discretization is made. Similarly, after the AS IS the indexing interface is held fixed while SPARSE and JACOBI CG solution modules are used. Then the RED-BLACK indexing module replaces the indexing interface information and REDUCED SYSTEM CG can be used. The 5-POINT STAR interface is not affected by using the second indexing module.

It is **important to note that not all combinations of modules are legal.** ELLPACK users should understand the basic premises of each module so they can determine whether a combination of modules is legal. Some illegal combinations are fairly obvious such as using the symmetric linear equation solver LINPACK SPD BAND with a discretization module that does not yield a symmetric linear system. Similarly, an INDEXING module which tries to minimize matrix bandwidth is not likely to help with a module for solving the equations iteratively. The module descriptions in Chapter 9 indicate some combinations which are legal. There are many other combinations possible and one should be cautious when the first using a new combination. A table at the beginning of Chapter 9 also gives some guidance as to which combinations are legal.

DISCRETIZATION. segment

This segment names a module to be used to form a linear system of equations. The content of this segment is a single module name and the list of available modules is expandable. The basic set in ELLPACK consists of

5-POINT STAR Ordinary second order divided central differences (Restricted to two dimensional domains)

7-POINT STAR Ordinary second order divided central differences (Restricted to three dimensional rectangular domains with Dirichlet boundary conditions)

SPLINE GALERKIN Galerkin method with piecewise polynomials of general degree and smoothness (Restricted to self-adjoint problems on two dimensional rectangular domains)

HERMITE COLLOCATION Collocation method with bi-cubic Hermite piecewise polynomials (Restricted to rectangular domains in two dimensions)

HODIE ACF Higher order finite differences for $a(x,y)u_{xx} + c(x,y)u_{yy} + f(x,y)u$ (Restricted to rectangular domains in two dimensions)

COLLOCATION Collocation method with Hermite bicubics on nonrectangular domains. (Restricted to two dimensions)

The complete set of modules supplied with the ELLPACK system contains about 10 more discretization modules, a list of these modules is given at the start of Chapter 9. Note that INTERIOR COLLOCATION is more efficient than HERMITE COLLOCATION, but not quite as general in the boundary condition it handles.

INDEXING. segment

These modules take the linear system produced by the DISCRETIZATION and reorganize it by renumbering the equations and/or unknowns. For example, one may wish to have the nested dissection ordering of the equations before using Gauss-elimination to solve them. The basic set in ELLPACK consists of:

NESTED DISSECTION Computes the nested dissection ordering of the equations.

AS IS The ordering is that of the generation of the equations and unknowns by the discretization modules. This is the default case.

RED-BLACK The variables and unknowns are numbered as on a checker board, all "red" points before the "black" points. Used only with REDUCED SYSTEM iteration.

MINIMAL DEGREE	Computes the minimal degree ordering of the equations

There are several other INDEXING modules supplied with the complete ELLPACK system described in Chapter 9.

SOLUTION. segment

These modules solve the linear system of equations. This step may also involve reformatting the equations. For example, modules for solving banded systems of linear equations require the equations to be in a certain band matrix format before the Gauss elimination is done, so they do the reformatting as well as the solution of the equations. The basic set in ELLPACK are

BAND GE	Gauss elimination with scaled partial pivoting for a general band matrix
LINPACK SPD BAND	Cholesky elimination for a symmetric positive definite band matrix
JACOBI CG	Jacobi iteration with conjugate gradient acceleration
REDUCED SYSTEM CG	Reduced system iteration with conjugate gradient acceleration (Assumes the RED-BLACK indexing)
SOR	SOR iteration
SPARSE	General sparse matrix Gauss elimination

There are several other SOLUTION modules described in Chapter 9 and supplied with the complete ELLPACK system.

TRIPLE. segment

These modules combine the functions of discretization, indexing and solution of the resulting linear system. The one included in the basic set of ELLPACK modules is FFT 9-POINT which solves the Helmholtz equation on two dimensional rectangles with second, fourth or sixth (for Poisson problems only) order finite differences using Fast Fourier Transform techniques. There are several other

TRIPLE modules described in Chapter 9 and supplied with the complete ELLPACK system.

2.D. FORTRAN AND PROGRAM CONTROL

Fortran has two distinct uses in ELLPACK. The first and simplest is to define various functions that appear in the problem. Simple expressions like SIN(X+2.5*Y) can just be inserted wherever needed, but more complex functions may need several Fortran statements. These functions can be defined as ordinary Fortran FUNCTION subprograms and appended to the ELLPACK program in the SUBPROGRAM segment just before the END segment. They can then be used to define coefficients in the EQUATION or BOUNDARY segments just like built-in Fortran functions.

The second use of Fortran is to allow special calculations to be done. They might be something simple like printing a heading and a few key parameters or computing the maximum of UX(X,Y)**2 + UY(X,Y)**2. They might be complex auxiliary computations that require the full range of Fortran facilities. They might be computations that interact with the ELLPACK modules to solve nonlinear or other special problems. The more complicated uses are presented and illustrated in Chapters 4 and 5.

There are three segments for Fortran use.

FORTRAN. segment

The FORTRAN segment indicates lines of executable Fortran code to be inserted into the control program generated by ELLPACK. The ELLPACK system uses statement labels starting at 20000, so such labels must be avoided in the users program. This segment is illustrated as follows:

```
*                PRINT A HEADING
FORTRAN.
C       Q1DATE IS AN ELLPACK UTILITY TO PROVIDE THE DATE
        CALL Q1DATE(IMO,IDAY,IYR)
        WRITE(6,20) IMO, IDAY, IYR
   20   FORMAT(///20X, 'WING LIFT CALCULATION', 5X, I2, 2('-',I2)/
     A         20X, 'USING FINE GRID AND CUBIC SPLINES'///)

*
*       COMPUTE SOME PROPERTIES OF THE SOLUTION
*       AFTER THE PROBLEM IS SOLVED. U, UX AND UY ARE FUNCTIONS
*       DEFINED FROM THIS APPROXIMATE SOLUTION.
FORTRAN.
        DMAX = 0.0
        USUM = 0.0
        DO 10 I = 1,10
             YG = (I-1)*.2
             DO 10 J = 1,10
                  XG = (I-1)*.1
                  DMAX = AMAX1(DMAX, UX(XG,YG)**2 + UY(XG,YG)**2)
   10             USUM = DSUM + ABS(SQRT(U(XG,YG))
        USUM = USUM*.02
        PRINT 20, DMAX,USUM
   20   FORMAT(//'DMAX =', F10.4, 10X, 'SIZE SQRT(U)' = F10.4)
```

DECLARATION. segment

The DECLARATION segment indicates lines of Fortran declaration statements to be placed at the beginning of the ELLPACK control program. For example:

```
DECLARATIONS.
     INTEGER DIGIT, COUNTS(10)
     REAL MAXU4, MINU4, LOADS(20)
```

There is a real **work space array** R1WORK always available for use. This array is used for temporary storage by modules; it may also be used for scratch storage in Fortran segments. Note that the contents of R1WORK are probably altered by any ELLPACK module. Its size (given by the Fortran variable I1MWRK) is usually fairly large and can be made larger using the OPTIONS segment.

SUBPROGRAMS. segment

The SUBPROGRAMS segment indicates Fortran complete FUNCTIONS or SUBROUTINES. For example, the user can define A(X,Y) or A(X,Y,Z) to be the coefficient of UXX in the PDE. This segment must be **at the end of the ELLPACK program**, just before the END segment.

Lines with a $ in column 1 are handled specially in SUBPROGRAM segments; the $ is stripped off and the line is copied, shifted one character to the left. The $ is for those Fortran systems (mercifully rare) that require control cards for each Fortran subprogram.

2.E. OUTPUT AND OPTIONS SEGMENTS

OPTIONS. segment

This segment sets various switches of the ELLPACK system. The OPTIONS segment must be near the start of the program, similar to a declaration. Some options may be changed during execution by setting internal ELLPACK Fortran variables. If this can be done, the variables are listed with the description of the option.

INTERPOLATION=k — Select the method of interpolation to define U(X,Y), etc. off the grid (for finite difference methods only).

- k=QUADRATICS — Local quadratic polynomials (default)
- k=SPLINES — Use B-splines of degree appropriate for the order of the discretization module. For nonrectangular domains...***. See [deBoor, 1978] for a description of B-splines; the interpolation routines were adapted from deBoor's PPPACK software

LEVEL = k — Set output levels (0-5) in ELLPACK run.

- LEVEL=0 — Requests no output from modules except fatal error messages
- LEVEL=1 — Request minimal output (default)
- LEVEL=2 — Requests reasonable summary of what happened
- LEVEL=3,4,5 — More and more intermediate output, primarily useful for debugging
- I1LEVL — Fortran variable for LEVEL

MEMORY — Give estimates of the memory used in the ELLPACK run with some breakdown

NO EXECUTION	Do not run ELLPACK program
PAGE=k	Select type of pagination for module output
PAGE=0	No page advances
PAGE=1	New page before DIS, TRIPLE, TABLE or SUMMARY (default)
PAGE=2	New page before every module and OUTPUT segment
I1PAGE	Fortran variable for PAGE
SELF-ADJOINT=k	Set the switch for self adjoint form of the PDE. k may be .TRUE. or .FALSE.
L1SELF	Fortran variable for SELF-ADJOINT
TIME	Give the execution times of each module
L1TIME	Fortran variable for TIME(can only turn TIME off)
MAX WORKSPACE=k	Limit the automatic workspace estimate and declare the workspace array R1WORK to have dimension at most I1MWRK=k.
MIN WORKSPACE=k	Set workspace array R1WORK to have dimension at least I1MWRK=k.

Options are not dynamic and, if given more than once, the last appearance is used.

The INTERPOLATION option specifies how the functions U(X,Y), UX(X,Y), etc. are defined for some, mainly finite difference, modules. If a discretization produces approximate values only on a grid of points, then an interpolation algorithm is used to provide values off the grid. Only one interpolation algorithm can be used in an ELLPACK run. The choice INTERPOLATION = SPLINES usually involves a substantial computation, but is more appropriate for use with higher order accurate finite difference discretizations.

There are several other options that are discussed in Chapter 4. Some OPTIONS segment examples follow.

```
*
*           REQUEST BASIC STATISTICS ON PERFORMANCE
*
OPTIONS.  TIME $ MEMORY
*
*           ENLARGE WORKSPACE, REQUEST MORE OUTPUT
*
```

```
OPT.      MIN-WORKSPACE=7500 $ LEVEL=2
*
*            ENLARGE WORKSPACE, SUPPRESS PAGING
*
OPT.      MIN-WORKSPACE=12000 $ PAGE=0
```

OUTPUT. segment

This segment specifies various kinds of output from the computation. The requests are of the forms:

<type> or <type>(<function>) or <type>(<function),<grid>)

where <type> is a keyword, <function> is a function name and <grid> defines a grid. The default <grid> is the one defined in the GRID segment. A uniform grid within the standard grid is defined by NX,NY or NX,NY,NZ where NX, NY and NZ are integers, the number of grid lines for each of the X,Y,Z variables. The list of types is:

MAX(f)
MAX(f,grid)
Print maximum value, simple least squares and average absolute value (L_1 norm of f), all based on the grid. These values are, respectively,

$\max |f(x_i,y_j)|$

$\left[\frac{1}{NX*NY}\sum_{i,j} f^2(x_i,y_j)\right]^{1/2}$

$\frac{1}{NX*NY}\sum |f(x_i,y_j)|$

where the grid is (x_i,y_j), $i=1$ to NX $j=1$ to NY.

RMS(f)
RMS(f,grid)
Same as MAX

NORM(f)
NORM(f,grid)
Same as MAX

PLOT(f)
PLOT(f,grid)
Contour plot of a function(f) of two variables. Here the grid size determines the smoothness of the contour lines, the default grid is 20 x 20.

PLOT DOMAIN
Display the domain with grid lines

TABLE(f)
TABLE(f,grid) Print table of function f at grid points

SUMMARY(f)
SUMMARY(f,grid) Equivalent to MAX(f) $ TABLE(f)

The function f may be one of the following standard ELLPACK functions or any user named function of two variables (or 3 variables in three dimensions):

U, UX, UY, UZ, UXX, UYY, UZZ, UXY, UXZ, YYZ — The solution function and its derivatives (defined after the solution is computed or after U has been initialized in a TRIPLE segment.)

TRUE — The known solution of the problem. Defined by the user in the SUBPROGRAM segment as
REAL FUNCTION TRUE(X,Y)
or
REAL FUNCTION TRUE(X,Y,Z).

ERROR — The error in the computed solution. The function TRUE must be provided, otherwise TRUE=0 is used.

RESIDU — If $Lu = f$ represents the partial differential equation, then the residual is $LU = f$ where U is the computed solution. If the equation is given in self-adjoint form
$(p(x,y)u_x)_x + (q(x,y)u_y)_y + r(x,y)u = f(x,y)$
then the user must supply the Fortran functions
REAL FUNCTION CDXU(X,Y)
and
REAL FUNCTION CDYU(X,Y)
which return the values of $\frac{\partial p}{\partial x}$ and $\frac{\partial q}{\partial y}$ respectively. Three dimensions requires similar functions CDXU, CDYU, CDZU with arguments X,Y,Z.

The following illustrate the OUTPUT segment's use.

```
*              CHECK HOW GOOD A SOLUTION IS (TRUE SOLUTION KNOWN)
*
OUTPUT.      MAX(ERROR) $ PLOT(ERROR) $ MAX(RESIDU)

*
*               QUICK LOOK AT RESULTS
*
OUTPUT.      SUMMARY(U) $ PLOT(U)

*
*            OUTPUT FUNCTIONS RELATED TO SOLUTION AND THE FORTRAN FUNCTION
*                 REAL FUNK(X,Y)
*                 FUNK = UX(X,Y)**2 + UY(X,Y)**2
*                 RETURN
*                 END
*
OUT.         TABLE (U) $ TABLE(UXX) $ TABLE(FUNK)
```

```
•
•             TABLE ON A GRID DIFFERENT THAN IN DISCRETIZATION
•
OUT.        TABLE(U,12,12)  $  TABLE(ERROR,6,6)
```

2.F DEBUGGING ELLPACK PROGRAMS

* * * DRAFT DEFERRED * * *

CHAPTER 3. ELLPACK EXAMPLES

This chapter gives example ELLPACK programs with output to illustrate the use of the facilities of Chapter 2. The first example is a revision of the initial example of Chapter 1; the domain has been made non-rectangular, and a normal derivative boundary condition used on one piece. The second example is a completely general equation with mixed boundary conditions on a rectangular domain. The third example shows how ELLPACK and Fortran interact. A problem is discretized and then solved several times with an iterative method; each time the convergence test is changed and the purpose is to examine the effect on accuracy achieved and execution time.

3.A PROBLEM OF CHAPTER 1 REVISED WITH NON-RECTANGULAR DOMAIN

The first example shows a simple case of non-rectangular domain; a quadrilateral. The quadrilateral could be specified completely by the LINE facility, but actual parameterized pieces are given to illustrate their use. Note how a rectangular grid is placed over the domain. Ordinary finite differences are used along with band Gauss elimination; there are only a few discretization modules in ELLPACK that are applicable to non-rectangular domains. Once the problem is discretized, several indexing or solution modules may be applied.

```
*          *****************************************************
*          *                                                   *
*          *  EXAMPLE ELLPACK PROGRAM 3.A1                     *
*          *                                                   *
*          *  REMARKS                                          *
*          *      THIS IS THE SAME EQUATION AS THE EXAMPLE IN  *
*          *      CHAPTER 1.  THE BOUNDARY CONDITIONS ARE CHANGED *
*          *      AND THE DOMAIN IS NO LONGER RECTANGULAR.     *
*          *                                                   *
*          *****************************************************
*
EQUATION.  UXX  +  UYY  +  3*UX  -  4*U  =  EXP(X+Y)*SIN(PI*X)

BOUNDARY.  U = 0.           ON  X = 0., Y = T   FOR  T = -1. TO 2.
           U = X**2         ON  X = R,  Y = 2.  FOR  R =  0. TO 1.
           U = 1. + Y/2.    ON  LINE 1.,2.  TO 1., 0.
           U = X*Y/2.       ON  X = 1.-S, Y = -S   FOR S = 0. TO 1.

GRID.      6  X POINTS    0. TO 1.
           6  Y POINTS   -1. TO 2.
```

```
DISCRET.    5 POINT STAR
INDEXING.   AS IS
SOLUTION.   BAND GE

OUTPUT.     TABLE(U)  $  PLOT(U)

END.


         SYMBOL TABLE INPUT TIME  2.47 SECONDS
        PROGRAM PROCESSING TIME    .83 SECONDS
           TEMPLATE OUTPUT TIME  2.08 SECONDS
                     TOTAL TIME  5.38 SECONDS
```

Output of ELLPACK run:

```
------------------
 DOMAIN  PROCESSOR
------------------

    DOMAIN PROCESSOR BEGINNING EXECUTION
    FOUND  19 BOUNDARY POINTS WHERE THE
     4 PIECES INTERSECT THE   6 X   6 GRID


------------------------
 DISCRETIZATION  MODULE
------------------------

    5 - P O I N T     S T A R

    DOMAIN                          NON-RECTANGULAR
    UNIFORM GRID                             6 X  6
    HX                                    .200E+00
    HY                                    .600E+00
    OUTPUT LEVEL                                 1
    BOUNDARY CONDITIONS
    PIECE  1                                TYPE 1
    PIECE  2                                TYPE 1
    PIECE  3                                TYPE 1
    PIECE  4                                TYPE 1
    NUMBER OF EQUATIONS                         14
    MAX NO. OF UNKNOWNS PER EQ.                  5
    EXECUTION SUCCESSFUL


-----------------
 INDEXING  MODULE
-----------------

    N A T U R A L

    NUMBER OF EQUATIONS                         14
    EQUATIONS/UNKNOWNS NUMBERED
        IN ORDER GENERATED
    EXECUTION SUCCESSFUL


-----------------
 SOLUTION  MODULE
-----------------

    E L L P A C K     B A N D
    NUMBER OF ROWS                              13
    NUMBER OF COLUMNS                           14
    NUMBER OF LOWER CO-DIAGONALS                 4
    NUMBER OF UPPER CO-DIAGONALS                 4
    ELLPACK BAND GIVES 2 TIMINGS
        SETUP TIME AND SOLUTION TIME
    EXECUTION SUCCESSFUL


 --------------------
  ELLPACK  78  OUTPUT
 --------------------


       ++++++++++++++++++++++++++++++++++++++++++++++++++
       +                                                +
       +    TABLE OF U       ON   6 X   6  GRID         +
       +                                                +
       ++++++++++++++++++++++++++++++++++++++++++++++++++


    X-ABSCISSAE ARE
    ---------------
     .000000E+00     .200000E+00     .400000E+00     .600000E+00
     .800000E+00     .100000E+01

       Y =  .200000E+01
       ----------------
     .000000E+00     .400000E-01     .160000E+00     .360000E+00
     .640000E+00     .100000E+01

       Y =  .140000E+01
       ----------------
     .000000E+00    -.966931E-01    -.846697E-01     .664861E-01
     .351265E+00     .700000E+00

       Y =  .800000E+00
       ----------------
     .000000E+00    -.743894E-01    -.727578E-01     .189445E-01
     .191984E+00     .400000E+00

       Y =  .200000E+00
       ----------------
     .000000E+00    -.717549E-01    -.915115E-01    -.597291E-01
     .135870E-01     .100000E+00

       Y = -.400000E+00
       ----------------
     .000000E+00    -.776765E-01    -.114505E+00    -.120000E+00
     .000000E+00     .000000E+00

       Y = -.100000E+01
       ----------------
     .000000E+00     .000000E+00     .000000E+00     .000000E+00
     .000000E+00     .000000E+00
```

3.B GENERAL EQUATION WITH MIXED BOUNDARY CONDITIONS RECTANGULAR DOMAIN

The second example shows a comparison of solving a problem with mixed boundary conditions by two different methods. Ordinary finite differences (5

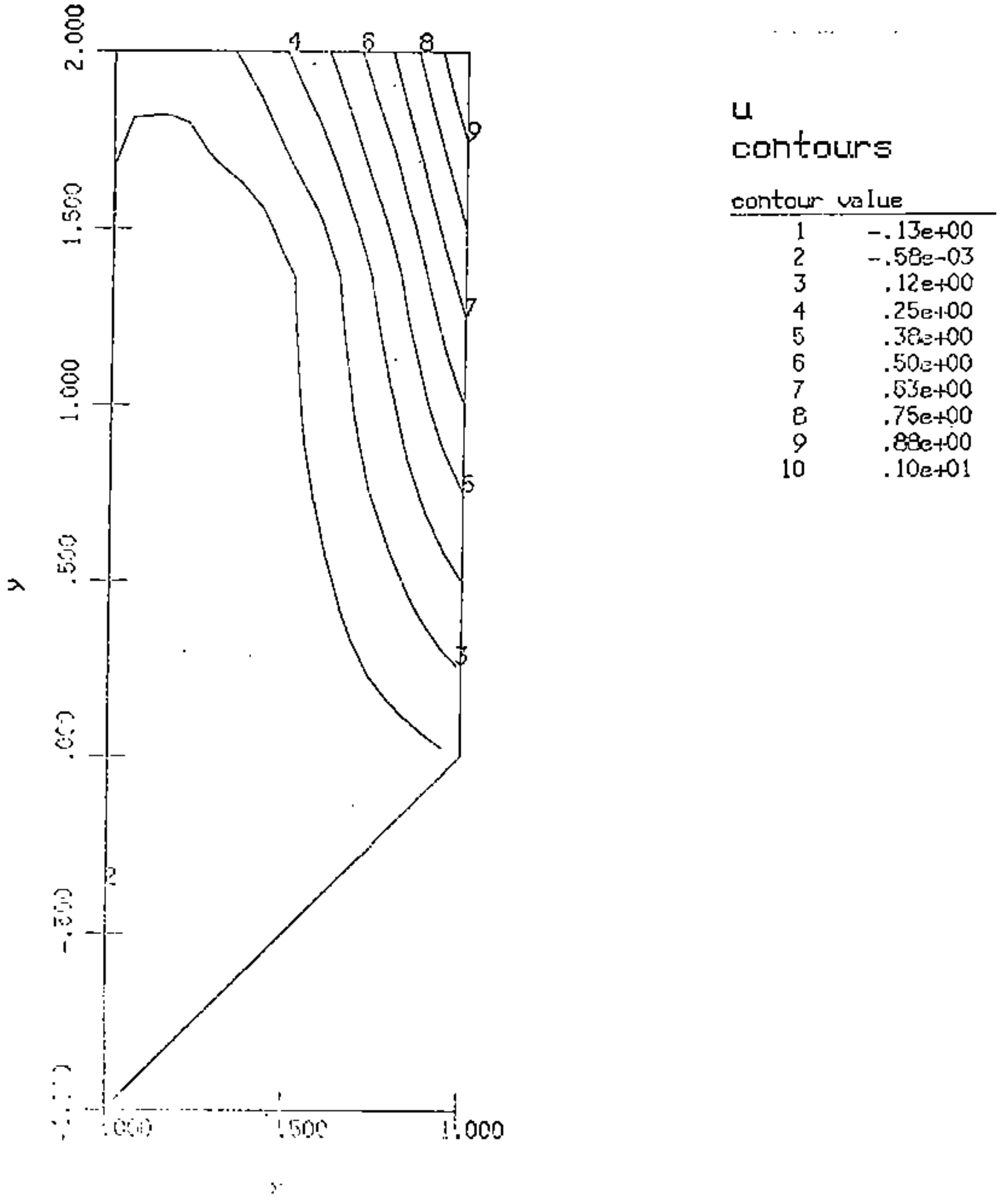

Figure 3.1. The contour plot produced by PLOT(U) in example ELLPACK program 3.A1.

POINT STAR) and Gauss elimination (BAND GE) gives an error of 1.5 percent while collocation (HERMITE COLLOCATION) and sparse Gauss elimination with pivoting (SPARSE PIVOTING) gives an error of 0.0055 percent. The times are time**FD and time**COL, both relatively small.

```
*         ****************************************************
*         *                                                  *
*         *  EXAMPLE ELLPACK PROGRAM 3.B1                    *
*         *                                                  *
*         *  REMARKS                                         *
*         *      THIS PROBLEM HAS MIXED BOUNDARY CONDITONS.  *
*         *      THE PROGRAM COMPARES TWO DISTINCT METHODS FOR *
*         *      SOLVING THE SAME PROBLEM.                   *
*         *                                                  *
*         *                                                  *
*         ****************************************************
*
OPTIONS.      LEVEL=1 $ TIME

EQUATION.
            UXX  +  (1.0+Y**2)*UYY  -  UX  -  (1.0+Y**2)*UY  =  F(X,Y)

BOUNDARY.
               - U +  UX = 0.           ON  X=1.
                 U       = TRUE(X,Y)    ON  Y=0.
                 U +  UX = 2.0*EXP(Y)   ON  X=0.
                 U       = TRUE(X,Y)    ON  Y=1.

GRID.         4 X POINTS  $  5 Y POINTS

OUT.          MAX(TRUE)

DIS.          5 POINT STAR
INDEX.        AS IS
SOL.          BAND GE

OUT.          TABLE(U) $ MAX(ERROR,7,9)

DIS.          HERMITE COLLOCATION
INDEX.        AS IS
SOL.          SPARSE PIVOTING  (MAXNZ=800)

OUT.          TABLE(U) $ MAX(ERROR,7,9)

SUBPROGRAMS.
      FUNCTION TRUE(X,Y)
C        THE STANDARD ELLPACK FUNCTION (IF KNOWN)
         TRUE = EXP(X+Y) + ((X*(X-1.0))**2)*ALOG(1.0+Y**2)
      RETURN
      END
      FUNCTION F(X,Y)
C        CONSTRUCT F SO TRUE IS AS GIVEN
         F = ALOG(1.0+Y**2) * (2.0 + X*(-14.0 + X*(18.0 - 4.0*X)))
     $      + 2.0*((X*(X - 1.0))**2)*(1.0 - Y - 2.0*Y**2/(1.0+Y**2))
      RETURN
      END

END.


       SYMBOL TABLE INPUT TIME  2.55 SECONDS
       PROGRAM PROCESSING TIME  1.33 SECONDS
          TEMPLATE OUTPUT TIME  2.22 SECONDS
                    TOTAL TIME  6.10 SECONDS
```

Output of ELLPACK run:

```
--------------------
 ELLPACK  77  OUTPUT
--------------------

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                        +
    +  MAX( ABS(TRUE  ) ) ON   4 X   5 GRID =  .7389056E+01   +
    +                                                        +
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


-----------------------
 DISCRETIZATION  MODULE
-----------------------

    5-POINT   STAR

    DOMAIN                                RECTANGLE
    X INTERVAL              .000E+00,   .100E+01
    Y INTERVAL              .000E+00,   .100E+01
    DISCRETIZATION                        UNIFORM
    GRID                                  4 X   5
    HX                                   .333E+00
    HY                                   .250E+00
    B.C.S ON PIECES 1,2,3,4               3,1,3,1
    OUTPUT LEVEL                                1
    NUMBER OF EQUATIONS                        12
    MAX NO. OF UNKNOWNS PER EQ.                 5
    EXECUTION SUCCESSFUL


-----------------
 INDEXING  MODULE
-----------------

    NATURAL

    NUMBER OF EQUATIONS                        12
    EQUATIONS/UNKNOWNS NUMBERED
        IN ORDER GENERATED
    EXECUTION SUCCESSFUL


-----------------
 SOLUTION  MODULE
-----------------

    LINPACK   BAND

    NUMBER OF ROWS                            13
    NUMBER OF COLUMNS                         12
    NUMBER OF LOWER CO-DIAGONALS               4
    NUMBER OF UPPER CO-DIAGONALS               4
    LINPACK BAND GIVES 2 TIMINGS
        SETUP TIME AND SOLUTION TIME
    EXECUTION SUCCESSFUL


--------------------
 ELLPACK  77  OUTPUT
--------------------
```

```
          +++++++++++++++++++++++++++++++++++++++++++
          +                                         +
          +    TABLE OF U      ON   4 X   5  GRID   +
          +                                         +
          +++++++++++++++++++++++++++++++++++++++++++


    X-ABSCISSAE ARE
    ---------------
     .000000E+00      .333333E+00      .666667E+00      .100000E+01

       Y =  .100000E+01
       ----------------
     .271828E+01      .382790E+01      .532872E+01      .738906E+01

       Y =  .750000E+00
       ----------------
     .211120E+01      .297928E+01      .416831E+01      .586884E+01

       Y =  .500000E+00
       ----------------
     .162945E+01      .231166E+01      .324632E+01      .457441E+01

       Y =  .250000E+00
       ----------------
     .125842E+01      .179158E+01      .251799E+01      .354874E+01

       Y =  .000000E+00
       ----------------
     .100000E+01      .139561E+01      .194773E+01      .271828E+01
```

```
--------------------
ELLPACK  77  OUTPUT
--------------------

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                      +
    +  MAX( ABS(ERROR ) ) ON   7 X   9 GRID =  .1142387E+00 +
    +                                                      +
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
------------------------
DISCRETIZATION  MODULE
------------------------

    C O L L O C A T I O N

    CASE                              NONHOMGENEOUS
    DOMAIN                                RECTANGLE
    X INTERVAL               .000E+00,    .100E+01
    Y INTERVAL               .000E+00,    .100E+01
    BOUND. COLLOC. PTS. PARAMS.           .000E+00
                                          .000E+00
    GRID                                  4 X   5
    HX                                    .333E+00
    HY                                    .250E+00
    OUTPUT LEVEL                                 1
    NUMBER OF EQUATIONS                         80
    MAX NO. OF UNKNOWNS PER EQ.                 16
    EXECUTION SUCCESSFUL
```

```
------------------
INDEXING  MODULE
------------------

    N A T U R A L
    NUMBER OF EQUATIONS                       80
    EQUATIONS/UNKNOWNS NUMBERED
        IN ORDER GENERATED
    EXECUTION SUCCESSFUL


-----------------
 SOLUTION  MODULE
-----------------

    S P A R S E     G E - P I V O T I N G

    NUMBER OF EQUATIONS                       80
    ESTIMATED MAX NUMBER OF NON-ZERO
        ELEMENTS IN UPPER TRI. FACTOR        800
    SIZE OF WORKING STORAGE                 3363
    SP. GE-PIV. GIVES 2 TIMINGS
        SETUP TIME AND SOLUTION TIME.
    NUMBER OF NON-ZERO MATRIX ELEMENTS       951
    NUMBER OF NON-ZERO ENTRIES IN
        UPPER TRIANGULAR FACTOR              783
    EXECUTION SUCCESSFUL


--------------------
 ELLPACK  77  OUTPUT
--------------------


          +++++++++++++++++++++++++++++++++++++++++++
          +                                         +
          +    TABLE OF U       ON   4 X   5  GRID  +
          +                                         +
          +++++++++++++++++++++++++++++++++++++++++++


    X-ABSCISSAE ARE
    ---------------
     .000000E+00     .333333E+00      .666667E+00      .100000E+01

       Y =  .100000E+01
       ----------------
     .271858E+01     .382819E+01      .532903E+01      .738940E+01

       Y =  .750000E+00
       ----------------
     .211736E+01     .297680E+01      .414562E+01      .575480E+01

       Y =  .500000E+00
       ----------------
     .164899E+01     .231217E+01      .322243E+01      .448180E+01

       Y =  .250000E+00
       ----------------
     .128415E+01     .179508E+01      .250400E+01      .349040E+01

       Y =  .000000E+00
       ----------------
     .100000E+01     .139563E+01      .194776E+01      .271828E+01


--------------------
 ELLPACK  77  OUTPUT
--------------------

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                          +
    +  MAX( ABS(ERROR ) ) ON   7 X   9 GRID =  .4091263E-03    +
    +                                                          +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

3.C EXAMPLE SHOWING HOW FORTRAN AND ELLPACK INTERACT

The third example is somewhat more complicated. The iteration method JACOBI CG has a parameter ZETA to terminate the iteration; the iteration on the linear system is done until the estimated error is less than ZETA. The object here is to test the effect of changing ZETA; values of 10^{-3}, 10^{-4} and 10^{-5} are used.

An important feature here is that parameters of the module JACOBI CG are changed at each iteration. Caution must be used as this does not always work; some parameters affect the program at preprocessing time rather than at execution time. (e.g. parameters which affect array sizes). Thus SPARSE(NSP=NWORK) will fail because a numerical value for NSP is required by the preprocessor and the value of the Fortran variable NWORK is not known until execution time. Another feature of this example is the use of self-adjoint form for the PDE.

This test shows that the stopping criterion has a substantial effect on the number of iterations. The results are summarized as follows:

ZETA	=	10^{-3}	10^{-4}	10^{-5}
Number of iterations	=	30	46	50

The maximum error in solving the elliptic problem is unaffected by these changes as it is due to the discretization error and not to the error in solving the linear system. Even with $ZETA = 10^{-3}$ the error in solving the linear system is less than the error .08 in discretizing the elliptic problem, thus no improvement is made by taking more iteractions.

```
*       *******************************************************
*       *                                                     *
*       *  EXAMPLE ELLPACK PROGRAM 3.C1                       *
*       *                                                     *
*       *  REMARKS                                            *
*       *      SELF-ADJOINT PROBLEM SOLVED BY FINITE DIFFERENCES *
*       *      AND ITERATION. THE PROGRAM TESTS THE EFFECT OF  *
*       *      USING A BETTER GUESS TO START THE ITERATION AND *
*       *      OF CHANGING THE STOPPING CRITERION.             *
*       *                                                     *
*       *******************************************************
*

EQUATION.   ( W(X,Y)*UX )X  +  ( W(X,Y)*UY )Y  =  F(X,Y)

BOUNDARY.   UX = 0.0  ON  X = 0.5
            U  = 0.0  ON  X = 1.0
            UY = 0.0  ON  Y = 0.5
            U  = 0.0  ON  Y = 1.0

GRID.       17 X POINTS
            17 Y POINTS

OUT.        PLOT(TRUE)  $  MAX(TRUE)

DISCRET.    5 POINT STAR
INDEXING.   AS IS

FORTRAN.
C
C     PRINT NUMBER OF ITERATIONS FOR JACOBI CG WITH STOPPING CRITERION
C     ZETA=1/10.**N FOR N=3,4,5
C

      DO 100 NZETA = 3 , 5
C
         PRINT 10, 1./10.**NZETA
   10    FORMAT(//5X,'* * ZETA =',E10.3,' * *')

SOL.     JACOBI CG (ITMAX = 50, ZETA = 1./10.**NZETA)

OUT.     MAX(ERROR)


FORTRAN.
C
  100 CONTINUE

SUBPROGRAMS.
      FUNCTION W(X,Y)
         COMMON /CONCOM/ PI
         DATA PI/3.14159265358979/
         W = ((PI*COS(PI*X)*SIN(PI*Y))**2 +
     C         (PI*SIN(PI*X)*COS(PI*Y))**2)**0.15
         RETURN
      END
      FUNCTION TRUE(X,Y)
         COMMON /CONCOM/ PI
         TRUE = SIN(PI*X)*SIN(PI*Y)
         RETURN
      END
      FUNCTION F(X,Y)
         COMMON /CONCOM/ PI
C        CONSTRUCT F SO TRUE IS AS GIVEN
         PI2 = PI * PI
         SINPIX = SIN(PI*X)
         SINPIY = SIN(PI*Y)
         COSPIX = COS(PI*X)
         COSPIY = COS(PI*Y)
         TU = SINPIX*SINPIY
         TUX = PI*COSPIX*SINPIY
         TUXX = -PI2*TU
```

```
          TUY = PI*SINPIX*COSPIY
          TUYY = -PI2*TU
          F = W(X,Y)*(TUXX + TUYY) + CDXU(X,Y)*TUX + CDYU(X,Y)*TUY
          RETURN
      END

END.
```

```
        SYMBOL TABLE INPUT TIME  2.57 SECONDS
        PROGRAM PROCESSING TIME  1.57 SECONDS
           TEMPLATE OUTPUT TIME  2.17 SECONDS
                     TOTAL TIME  6.30 SECONDS
```

Output of ELLPACK run (abbreviated, **** indicates where lines are deleted):

```
 --------------------
 ELLPACK  77  OUTPUT
 --------------------

    +++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                       +
    +  MAX( ABS(TRUE  ) ) ON  17 X  17 GRID =  .1000000E+01  +
    +                                                       +
    +++++++++++++++++++++++++++++++++++++++++++++++++++++++++


 -------------------------
 DISCRETIZATION  MODULE
 -------------------------

    5 - P O I N T    S T A R

    DOMAIN                              RECTANGLE
    X INTERVAL               .500E+00,  .100E+01
    Y INTERVAL               .500E+00,  .100E+01
    DISCRETIZATION                        UNIFORM
    GRID                                 17 X  17
    HX                                   .313E-01
    HY                                   .313E-01
    B.C.S ON PIECES 1,2,3,4               1,2,2,1
    OUTPUT LEVEL                                1
    NUMBER OF EQUATIONS                       256
    MAX NO. OF UNKNOWNS PER EQ.                 5
    EXECUTION SUCCESSFUL


****


    * * ZETA =  .100E-02 * *


 -----------------
 SOLUTION  MODULE
 -----------------

    J A C O B I   C G

    JACOBI-CG HAS CONVERGED IN    30 ITERATIONS.
```

```
--------------------
 ELLPACK  77  OUTPUT
--------------------

    +++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                       +
    +  MAX( ABS(ERROR ) ) ON  17 X  17 GRID =  .8015466E-01  +
    +                                                       +
    +++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    * * ZETA =  .100E-03 * *


-----------------
 SOLUTION  MODULE
-----------------

    J A C O B I   C G

    JACOBI-CG HAS CONVERGED IN    46 ITERATIONS.


****

    * * ZETA =  .100E-04 * *


-----------------
 SOLUTION  MODULE
-----------------

    J A C O B I   C G

*** W A R N I N G ************

    IN ITPACK ROUTINE JCG.
    ZETA =  .100E-04. A VALUE THIS SMALL MAY HINDER CONVERGENCE.

    JACOBI-CG HAS CONVERGED IN    50 ITERATIONS.


****
```

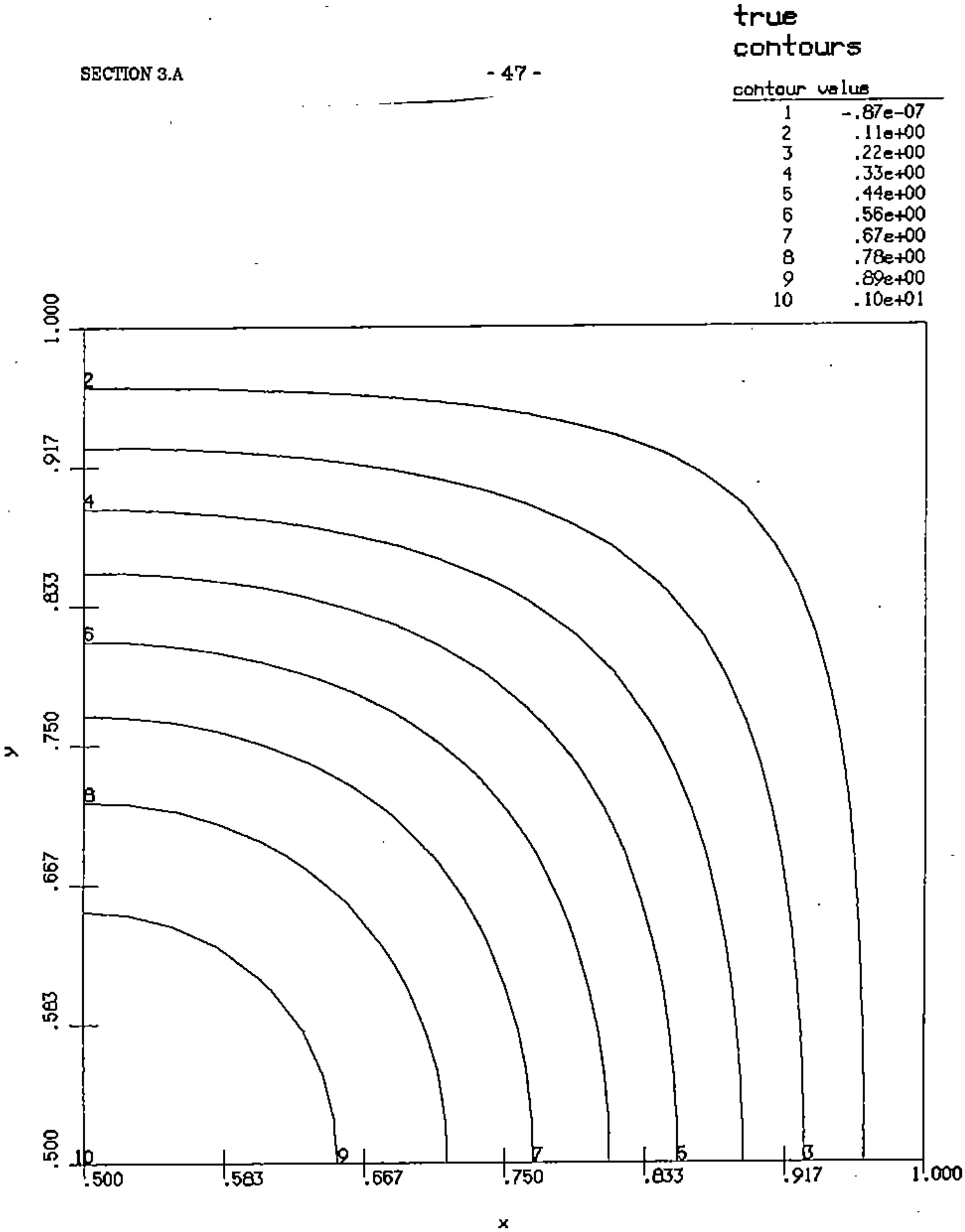

Figure 3.2. The contour plot produced by PLOT(TRUE) in example 3.C1

CHAPTER 4 ADVANCED ELLPACK FEATURES

This chapter presents features of ELLPACK which gives one more control over the problem solving process. The use of these features to solve more complex problems is illustrated in the final section of this chapter. Even more difficult examples are presented in Chapter 5. The additional ELLPACK language features are:

```
OPTIONS   :  TO SAVE OLD SOLUTIONS FOR ITERATION
             TO CONTROL STORAGE
             TO SET PROBLEM CHARACTERISTICS
HOLE, ARC:   TO HANDLE MORE COMPLEX DOMAINS
GLOBAL    :  TO PROVIDE PARAMETERS FOR THE PDE AND BOUNDARY CONDITIONS
PROCEDURE:   TO DISPLAY THE PATTERN OF NONZEROS IN THE MATRIX
             TO INITIALIZE UNKNOWNS FOR ITERATION METHODS
             TO COMPUTE EIGENVALUES OF THE DISCRETIZATION MATRIX
TRIPLE    :  TO INTERPOLATE BOUNDARY CONDITIONS
             TO INITIALIZE THE SOLUTION U
OUTPUT    :  TO TABLE INTERNAL ELLPACK VARIABLES FOR THE
                ELLIPTIC PROBLEM
                EQUATIONS, UNKNOWN AND INDEXES
                DOMAIN AND BOUNDARY
```

In addition, there is a section describing how one can access internal ELLPACK variables, including "preprocessor" or "template" variables. These features provide:

1. the capability to handle more general problems,
2. the capability to construct iterative methods (for nonlinear problems, etc.),
3. the ability to reduce computer resource use,
4. means to study the methods
5. more convenient programming in certain applications.

4.A ADDITIONAL SEGMENTS

There are four additional segments in ELLPACK described here.

HOLE. segment

This segment defines a hole to be removed from the domain of the problem. Its form is exactly like BOUNDARY except that the name HOLE is used. **HOLE segments must appear after the BOUNDARY segment,** and several HOLE segments may appear. The boundary of the hole must be given in the **opposite** direction of that of the domain boundary. Thus if CLOCKWISE = .TRUE. (specifying the domain boundary is defined clockwise) then the boundaries of the holes must be specified counter-clockwise.

The **grid must be fine enough** so that at least one interior grid point lies on any grid line between the boundary of a hole and the boundary of the domain. The short notation for rectangular domains cannot be used if there are holes in the rectangle. The reason is that short cuts are taken for rectangles in the preprocessor which leave it unprepared for a HOLE segment.

ARC. segment

This segment defines an arc or curved slit to be removed from the domain of the problem as well as side (boundary) conditions that apply on it. Its form is exactly like BOUNDARY and the same restrictions apply to ARC that apply to HOLE. Note that a single boundary condition is given on the arc. If "two sided" boundary conditions are needed, then long, narrow holes must be specified. See Section 5.A for examples which illustrate the technique. Arcs cannot divide the domain into two or more disjoint parts.

GLOBAL segment

This segment puts declarations in the ELLPACK control program as well as in all the Fortran subprograms generated by the ELLPACK preprocessor which define the PDE, the domain, and the boundary conditions. **It does not** affect the ELLPACK library subprograms (modules). Specifically, the internal ELLPACK affected are subprograms

Q1PRHS	The PDE right hand side function
Q1PCOE	The PDE coefficients subprogram
Q1BCOE	The boundary condition coefficients function
Q1BRHS	The boundary condition right hand side function
Q1BCOR	The boundary coordinates subroutine

This facility allows us to parameterize the elliptic problem and provide control of these parameters at the ELLPACK program level. To do this, one simply includes Fortran COMMON blocks in the GLOBAL segment. If the segment

```
GLOBAL. COMMON/SPECIL/A,K
```

is included then the generated right side function in the ELLPACK control program is

```
      REAL FUNCTION Q1PRHS(X,Y)
      COMMON/SPECIL/A
      Q1RPRHS = A *(1.+X)/(A + X * Y)
      RETURN
      END
```

Consider the following ELLPACK program fragment

```
EQ.      UXX + UYY - K*U = A(1.+X)/(A+X*Y)
GLOBAL.
         COMMON/PARAM/A,K
         REAL K
FORTRAN.
         DO 10 I = 1, 8
            A = 1. + (I-1) * 2.
```

```
          K = I

TRIPLE.   FFT 9-POINT(IORDER=4)
OUT.      PLOT(U) $ SUMMARY(U)

FORTRAN.
    10    CONTINUE
```

This problem is solved 8 times with 8 different values of the parameter A and K.

PROCEDURE. segment

The PROCEDURE segment provides facilities that are useful in solving or analyzing an elliptic problem but which are not one of the standard steps in solving the problem. The ELLPACK system is designed to allow one to add PROCEDURES for particular applications or specialized situations. The form of the PROCEDURE segment is the same as the DISCRETIZATION; a keyword with, possibly, parameters in parentheses.

There are four PROCEDURE facilities in the complete ELLPACK system:

EIGENVALUES: Compute eigenvalues of the discretization matrix.

HOMOGENIZE BOUNDARY CONDITIONS: Use the interpolant of INTERPOLATE BOUNDARY CONDITIONS to reduce the elliptic problem to one whose boundary conditions have zero right sides.

DISPLAY MATRIX PATTERN: Provides a printout of the pattern of non-zero elements in the matrix of the discretization.

INITIALIZE UNKNOWN FOR FINITE DIFFERENCES (U: <fname>). The values of the Fortran function

FUNCTION <fname> (X,Y)

are used to initial the unknowns R1UNKN of the linear system.

The default value for <fname> is U, otherwise <fname> must be provided in the SUBPROGRAM segment. If the default function $U(X,Y)$ is used, then this PROCEDURE **must be invoked before** the DISCRETIZATION. The standard use of this PROCEDURE is:

```
TRIPLE.      INTERPOLATE BOUNDARY CONDITIONS BY BLENDING
PROCEDURE.   INITIALIZE UNKNOWNS FOR FINITE DIFFERENCES
DIS.         5-POINT STAR
INDEX.       AS IS
SOLUTION.    SOR
```

The TRIPLE used here is described in the next section. This procedure depends on the discretization module's ordering of the grid points, it is applicable to the 5 POINT STAR, 7 POINT STAR and HODIE discretizations.

More detailed descriptions are given in Chapter 9 for each of these procedures.

4.B ADDITIONAL FEATURES OF BASIC SEGMENTS

OPTION. segment

There are several additional options useful for complex ELLPACK applications. There are:

(a) **Creation of functions for previous solutions of the problem.**

The option OLDU=k provides functions

```
U1, UX1, UY1, UXX1, UXY1, UYY1
...
UK, UXK, UYK, UXXK, UXYK, UYYK
```

which are the *k* previous solutions of the elliptic problem. These functions are used in iterations for nonlinear problems or for steps in time dependent problems. There are corresponding arrays R1UNK1, R1UNK2, ..., R1UNKk for the unknowns. Note that, except in a very few instances, **the discretization or grid cannot be changed while using a set of previous solutions.**

(b) Control of dimensions in ELLPACK

The ELLPACK system creates a Fortran program with dimensions declared for all variables. Sometimes ELLPACK creates arrays for a particular problem which are not used or which are declared larger than needed. Every dimension of an array has its own variable in the ELLPACK preprocessor and these can be set in the OPTIONS segment. Effective use of this facility requires one to become familiar with the ELLPACK control program.

For example, the dimension of the array of unknowns is I1MUNK and the statement

```
OPTION.   I1MUNK=388
```

sets this dimension to 388 independent of what the normal size of this array is. Table 4.1 gives a sample set of the more important array names, their dimension variables and a brief description of the array. See the tables in Chapter 18 for the complete set. Note that these variables all have third character *M*, the second character is 1 to indicate that this variable belongs to the ELLPACK control program.

Table 4.1. Variables for Control of more Important Array Dimensions in ELLPACK

Array Name	Preprocessor Dimension Control	Description
R1UNKN	I1MUNK	The unknowns of the linear system
R1COEF	I1MCOE	The discretization coefficients
I1IDCO	I1MIDC	The column identification for R1COEF
I1ENDX	I1MEND	The equation reordering permutation vector
I1UNDX	I1MUND	The unknown reordering permutation vector
I1MXEQ	R1COEF, I1IDCO, I1ENDX, I1UNDX, R1UNKN	Maximum number of equations
I1MXBP	R1XBND, R1YBND I1PECE, R1BPAR I1BPTY, I1BGRD I1BNGH	Maximum number of points on non-rectangular boundary

(c) **Control of problem characteristics.** The ELLPACK system automatically examines the equation and sets .TRUE. or .FALSE. values for the following Fortran variables:

L1LAPL	Laplace's equation
L1CSTC	Constant coefficients
L1POIS	Poisson problem
L1HMEQ	Homogeneous PDE

These automatic settings can be overridden by assigning new values to these variables in a Fortran segment. Thus, for example, setting

```
L1STC = .FALSE.
```

would have the PDE

```
EQUATION.  3*UXX + 2*UYY - 7*U = 0
```

classified as variable coefficients and thus a module which does something special for constant coefficients would not do that in this run. HODIE ACF is an example of such a module and one can evaluate the benefit of its special action for constant coefficients.

Warning: These options must be used with caution because

(a) there is no guarantee that a module acts upon these variables,

(b) a module can do its own analysis of the problem, and hence there might be no effect, and

(c) since values of these variables are known to the preprocessor, they sometimes determine the dimensions of arrays or even the selection of a subprograms loaded. In summary, these options should be tried on an experimental basis. Although there are many situations where they are convenient, there are some where they cause the ELLPACK run to fail (perhaps in a mysterious way).

(d) Initialization of solution function $u(x,y)$.

Iteration methods may be used in ELLPACK to solve nonlinear problems, time dependent problems or systems of algebraic equations. Each of these processes must be initialized; the facilities here are in the TRIPLE segment because initialization has the same effect as completely solving the elliptic problem. Here, however, we do not expect to obtain much accuracy. The three TRIPLE modules are:

INTERPOLATE BOUNDARY CONDITIONS BY BLENDING: Use Blending Function interpolations to define $U(X,Y)$ as a smooth function which exactly matches the elliptic problem's boundary conditions. Applicable only for rectangular domains and boundary conditions with constant coefficients.

INTERPOLATE BOUNDARY CONDITIONS BY BI-CUBICS: Use Hermite bi-cubics to define $U(X,Y)$ which interpolates the elliptic problem's boundary conditions at the boundary grid points plus two points in between each pair plus at the corners. Applicable only for rectangular domains and uncoupled boundary conditions (only UX or UY specified at any point).

INITIALIZE SOLUTION (U= <fname>): The values of the Fortran function

FUNCTION <fname> (X,Y)

are tabled at the grid and then extended by interpolations to define $U(X,Y)$, etc. everywhere.

In each case these TRIPLES create all the standard ELLPACK functions U, UX, UY, UXX, UXY and UYY. Note that INTERPOLATE BOUNDARY CONDITIONS BY BI-CUBICS produces a $U(X,Y)$ which is identically zero away from the boundary; its use is primarily for boundary layer problems where this interpolant provides an approximation to the differences between the "smooth" solution in the interior and the actual solution.

(e) **Tabulate internal variables in OUTPUT**

Certain tables of internal ELLPACK variables can be printed. These can be useful for complicated problems where one has to interact with the internal data of ELLPACK. The additional output statements are given below with a brief description of the resulting output.

TABLE PROBLEM	List of current status of ELLPACK variables which define the problem
TABLE INDEXES	Table of equations produced by discretization modules
TABLE INDEXES	Table of ELLPACK indexing arrays
TABLE UNKNOWN	Table of unknowns of the linear system
TABLE DOMAIN	Tables that define a non-rectangular domain's relation to the rectangular grid
TABLE BOUNDARY	Tables of the arrays produced by the domain

processor for non-rectangular domains.

To understand the information provided by these statements users must be familiar with the ELLPACK interfaces defined in Part 3, Chapter 14.

4.C ACCESS TO PREPROCESSOR VARIABLES

In some applications of ELLPACK one needs to refer to values which the preprocessor computes and which are inconvenient (or worse) to compute while writing the ELLPACK program. The simplest instance of this is the dimension of an array, say workspace R1WORK or the unknowns R1UNKN. If one wants to create a new array of the same or related size, one does not know what size to dimension it. There is a mechanism which allows access to certain variables of this type, called <u>template variables</u> from the ELLPACK program. In the above cases, the dimension of the R1WORK and R1UNKN arrays are I1WORK and $I1MUNK; these are exactly the same names as listed in Table 4.1 preceeded by a $. Thus, the ELLPACK code fragment

```
DECLARATIONS.
    REAL COPYU($I1MUNK), WORK2($I1WORK,2)
GLOGAL.
    COMMON/PASSER/UNKOLD($I1UNK)
```

is read by the preprocessor and correct numerical values substituted for template variables that appear. This substitution is made when the ELLPACK control program is generated so that it will compile.

Table 4.2 gives the more useful of these variables, a few others may be identified from the tables in Chapter 18.

Table 4.2. Available Template Variable

Name	Description
$I1KBAN	Band width of matrix generaed for band solver
$I1KWRK	Dimension of R1WORK, workspace

$I1MEND	Dimension of I1ENDX, indexing vector
$I1MIXD	Dimension of I1UNDX, indexing vector
$I1MNCO	Column dimension of R1COEF, coefficient matrix
$I1MXEQ	Dimension of I1UNKN, maximum number of unknowns
$I1MXPT	Row dimension of R1COEF, coefficient matrix
$I1NBND	Number of boundary pieces
$I1MXPT	Maximum number of boundary points
$I1NGRX	Dimension of R1NGRX, x-grid vector
$I1NGRY	Dimension of R1NGRY, y-grid vector
$I1NGRZ	Dimension of R1NGRZ, z-grid vector

If the variables are used in a context where they are not followed by a blank or special character (which is unlikely), the six characters may be enclosed in parentheses. That is $I1KBAN and $(I1KBAN) are treated the same so one can use the Fortran statements

```
      WRITE(I1OUTP,20) R1COEF
20   FORMAT(/ / 'THE $(1MXEQ)X$(I1MNCO) COEFFICIENT ARRAY' /
  A          ( $(I1MNCO)F10.5))
```

The Fortran 77 PARAMETER statement can be used with these variables to create dimensions for arrays of related sizes. For example, the statements (assuming ELLPACK is being used with a Fortran 77 compiler).

```
DECLARATIONS.
     PARAMETER (NSIDE = ($I1NGRX-2) * ($I1NGRY-2))
     PARAMETER (ENEDGE = 2*($I1NGRX + $I1NGRY-1))
     REAL INTER(NSIDE,3), RECTB(NEDGE), UUINTER(NSIDE)
     INTEGER IDEDGE(NEDGE), KTYPE(NSIDE)
```

gives numerical values to NSIDE and NEDGE by the time the ELLPACK control program is generated. Thus, one has five arrays whose dimensions are related to the number of interior and edge points of the rectangular grid.

4.D ADVANCED ELLPACK EXAMPLES

This section presents four example problems solved with ELLPACK. The first illustrates how to make a parameter study (vary physical parameters) for an application to the solidification of alloys. This is an actual application of

ELLPACK to a real world problem. The second example shows how to use ELLPACK PROCEDURES to analyze numerical methods. While this example is artificially simplified here, these procedures can be very useful in practice. The third example illustrates how to solve nonlinear problems using Picard iteration. It also illustrates how one can use internal ELLPACK variables if one knows about them and needs to. In this instance, we could avoid any use of internal variables as pointed out in the discussion. Finally, there is an example solving a problem on an elliptical domain with an elliptical hole in it.

EXAMPLE 4.D1 Parameter study for Alloy Solidification.

The following elliptic boundary value problem is of interest in the study of the solidification of metallic alloys.

$$\nabla^2 u - (\beta/2)^2 u = c$$

$$\begin{array}{rll} u & =0 & \textit{on } y = y_\infty \\ u_n & =0 & \textit{on } y = 0 \text{ and } x=1/2 \\ u_n + \beta u/2 & =-\beta\, \textit{sinh}(\beta y/2) & \textit{on } y = y=w(x) \end{array}$$

The domain represents a liquid alloy behind a solidification front $w(x)$ moving at constant velocity $\dot{V}$ in the $-y$ direction. The coordinate system is taken to be moving at this velocity and the system is assumed at steady-state. The function u is then related to the concentration of solute in the liquid according to the formula

$$c(x,y) = c_0(y) + u(x,y)\, \textit{exp}(-\beta y/2)$$

where $\dot{c}_0(y) = 1 + \textit{exp}(-\beta y)$ is the concentration for an unperturbed solid-liquid interface (δ=0). The sides of the container are at $x = 1/2$ and $x = -1/2$, but the domain is truncated at x=0 due to symmetry. Also the boundary condition along the topmost edge is actually u --> 0 as y --> ∞, but is truncated to some

Figure 4.1. The domain for Example 4.1.

finite value y^*.

We wish to perform parameter studies with respect to ***beta***,δ, and y_∞. The parameter δ is the amplitude of the solid-liquid interface, and $\beta = VL/D$ where V is the solidification velocity, L is the actual half-width of the contaner, and D is the diffusivity of the solute in the liquid. In each case we wish to determine the solute distribution along the solid-liquid interface. For more information see: S.R. Coriell, R.F. Boisvert, R.G. Rehm, and R.F. Sekerka, Lateral solute segregation during unidiretional solidification of a binary alloy with a curved solid-liquid interface II; large departures from planarity, J. Crystal Growth, 54 (1981), pp. 167-175.

The following ELLPACK program solves this problem. Note the use of the GLOBAL segment to parameterize the program. The parameter values for the run are read and set in the Fortran segment.

```
*          *****************************************************
*          *                                                   *
*          *  EXAMPLE ELLPACK PROGRAM 4.D1                     *
*          *                                                   *
*          *  REMARKS                                          *
*          *      MODEL OF SOLUTE SEGREGATION DURING UNIDIRECTIONAL *
*          *      SOLIDIFICATION OF A BINARY ALLOY WITH A CURVED   *
*          *      SOLID-LIQUID INTEFACE                        *
*          *                                                   *
*          *****************************************************
*

EQUATION.  UXX + UYY - BDV2SQ*U = 0.0

BOUNDARY.  UX = 0.0 ON X=0.0    Y=T          FOR T=W(0.0) TO YINF
           U  = 0.0 ON X=T,     Y=YINF       FOR T=0.0 TO 0.5
           UX = 0.0 ON X=0.5,   Y=YINF-T     FOR T=0.0 TO YINF-W(0.5)

           -DW(X)*UX + UY + BOV2*U = -BETA*SINH(BOV2*Y)
                  ON  X=0.5-T, Y=H(0.5-T) FOR T=0.0 TO 0.5

GLOBAL.
      COMMON /PARAMS/ BETA,BOV2,BDV2SQ.YINF,DELTA,TWOP1,TWOPID

FORTRAN.
C
C   SET PROBLEM PARAMETERS
C
      READ(5,*) YINF,BETA,DELTA
      BOV2 = 0.5*BETA
      BOV2SQ = BOV2*BOV2
      TWOPI = 2.0*PI
      TWOPID = TWOPI*DELTA
```

```
GRID.
             30 X POINTS
             40 Y POINTS


*                   SOLVE PDE PROBLEM FOR THE FUNCTION U
DISCRETIZATION.     5-POINT STAR
INDEXING.           MINIMUM DEGREE
SOLUTION.           SPARSE

*                   PLOT CONTOURS OF SOLUTE CONCENTRATION
OUTPUT.             PLOT(C)

FORTRAN.
C
C   TABULATE CONCENTRATION ALONG INTERFACE
C
      READ(5,*) NPTS
      WRITE(6,2000)
 2000 FORMAT('1 TABLE OF CONCENTRATION ALONG INTERFACE'//
     *      '        X                 C(X,W(X))'/              )
      DX = 0.5*FLOAT(INPTS-1)
      DO 100 I=1,NPTS
         X = FLOAT(I-1)*DX
         CVAL = C(X,W(X))
         WRITE(6,2001) X,CVAL
  100 CONTINUE
 2001 FORMAT(1X,2E18.6)

SUBPROGRAMS.
      REAL FUNCTION W(X)
C
C   SHAPE OF THE SOLID-LIQUID INTERFACE
C
      COMMON /PARAMS/ BETA,BOV2,BOV2SQ,YINF,DELTA,TWOPI,TWOPID
      W = DELTA*COS(TWOPI*X)
      RETURN
      END
      REAL FUNCTION DW(X)
C
C   DERIVATIVE OF SOLID-LIQUID INTERFACE SHAPE
C
      COMMON /PARAMS/ BETA.BOV2,BOV2SQ,YINF,DELTA,TWOPI,TWOPID
      DW = -TWOPID*SIN(TWOPI*X)
      RETURN
      END
      REAL FUNCTION C(X,Y)
C
C   COMPUTES SOLUTE CONCENTRATION FROM PDE SOLUTION
C
      COMMON /PARAMS/ BETA,BOV2,BOV2SQ,YINF,DELTA,TWOPI,TWOPID
      C = 1.0 + EXP(-BETA*Y) + U(X,Y)*EXP(-BOV2*Y)
      RETURN
      END
      REAL FUNCTION SINH(Z)
C
C   DIRECT COMPUTATION OF HYPERBOLIC SINE FUNCTION
C
      SINH = 0.5*(EXP(Z)-EXP(-Z))
      RETURN
      END
END.
```

EXAMPLE 4.D2 Use of PROCEDURES to analyze a method.

One of the objectives of ELLPACK is to assist in the analysis of numerical methods for PDEs. Two ELLPACK tools for this purpose are the procedures

DISPLAY MATRIX PATTERN and EIGENVALUES. The first is useful in analyzing various data structures and techniques for Gauss elimination methods and the second is useful to analyzing the applicability of iterative methods. The following example shows the results of applying these tools to the 5 POINT STAR discretization with three indexings (NATURAL, RED BLACK and NESTED DISSECTION).

```
*         ****************************************************
*         *                                                  *
*         *  EXAMPLE ELLPACK PROGRAM 4.D2                    *
*         *                                                  *
*         *  REMARKS                                         *
*         *     STUDY OF ZERO PATTERNS AND EIGENVALUES OF    *
*         *     MATRICES OBTAINED FROM 5 POINT STAR WITH     *
*         *     DIFFERENT INDEXINGS OF THE EQUATIONS.        *
*         *                                                  *
*         ****************************************************
*
EQUATION.
.                 - UXX - UYY = 0.

BOUND.
                   U = 0.  ON X = 0.0
                           ON X = 1.0
                           ON Y = 0.0
                           ON Y = 1.0

GRID.              8 X POINTS $ 8 Y POINTS

OPTION.            TIME

DIS.               5 POINT STAR
INDEX.             AS IS

PROC.              DISPLAY MATRIX PATTERN (MATNBR=1,MATNBC=1,MATBLK=6)

FORT.
        HX = R1HXGR
        HY = R1HYGR
        WRITE(I1OUTP,1000)
        DO 10 M = 1, I1NGRX-2
           DO 10 N = 1, I1NGRY-2
              E = (4.-2.*COS(PI*HX*M)-2.*COS(PI*HY*N)) / (PI**2*HX*HY)
              WRITE(I1OUTP,1010) M, N, E
   10   CONTINUE
 1000   FORMAT(//4X,1HM,3X,1HN,5X,17HEXACT EIGENVALUES  /)
 1010   FORMAT(1X,2I4,4X,E15.6)

PROC.              EIGENVALUES (SCALE = 1./PI**2)

INDEX.             RED BLACK

PROC.              DISPLAY MATRIX PATTERN (MATNBR=1,MATNBC=1,MATBLK=18)

INDEX.             NESTED DISSECTION (NDTYPE=5)

PROC.              DISPLAY MATRIX PATTERN (MATNBR=1,MATNBC=1,MATZER=1H )

END.
```

```
        SYMBOL TABLE INPUT TIME  2.70 SECONDS
        PROGRAM PROCESSING TIME  1.27 SECONDS
           TEMPLATE OUTPUT TIME  2.15 SECONDS
                     TOTAL TIME  6.12 SECONDS
```

Output of ELLPACK run:

```
 -----------------------
 DISCRETIZATION  MODULE
 -----------------------

    5 - P O I N T    S T A R

    DOMAIN                                 RECTANGLE
    X INTERVAL                 .000E+00,  .100E+01
    Y INTERVAL                 .000E+00,  .100E+01
    DISCRETIZATION                          UNIFORM
    GRID                                    8 X   8
    HX                                     .143E+00
    HY                                     .143E+00
    B.C.S ON PIECES 1,2,3,4                 1,1,1,1
    OUTPUT LEVEL                                  1
    NUMBER OF EQUATIONS                          36
    MAX NO. OF UNKNOWNS PER EQ.                   5
    EXECUTION SUCCESSFUL


 -----------------
 INDEXING  MODULE
 -----------------

    N A T U R A L

    NUMBER OF EQUATIONS                          36
    EQUATIONS/UNKNOWNS NUMBERED
        IN ORDER GENERATED
    EXECUTION SUCCESSFUL


 -----------------
 PROCEDURE MODULE
 -----------------

    D I S P L A Y  M A T R I X  P A T T E R N



                1           2           3
     123456 789012 345678 901234 567890 123456

   1 DX.... X..... ...... ...... ...... ......
   2 XDX... .X.... ...... ...... ...... ......
   3 .XDX.. ..X... ...... ...... ...... ......
   4 ..XDX. ...X.. ...... ...... ...... ......
   5 ...XDX ....X. ...... ...... ...... ......
   6 ....XD .....X ...... ...... ...... ......

   7 X..... DX.... X..... ...... ...... ......
   8 .X.... XDX... .X.... ...... ...... ......
   9 ..X... .XDX.. ..X... ...... ...... ......
  10 ...X.. ..XDX. ...X.. ...... ...... ......
  11 ....X. ...XDX ....X. ...... ...... ......
  12 .....X ....XD .....X ...... ...... ......
 13 ...... X..... DX.... X..... ...... ......
 14 ...... .X.... XDX... .X.... ...... ......
 15 ...... ..X... .XDX.. ..X... ...... ......
 16 ...... ...X.. ..XDX. ...X.. ...... ......
 17 ...... ....X. ...XDX ....X. ...... ......
 18 ...... .....X ....XD .....X ...... ......

 19 ...... ...... X..... DX.... X..... ......
 20 ...... ...... .X.... XDX... .X.... ......
 21 ...... ...... ..X... .XDX.. ..X... ......
 22 ...... ...... ...X.. ..XDX. ...X.. ......
 23 ...... ...... ....X. ...XDX ....X. ......
 24 ...... ...... .....X ....XD .....X ......

 25 ...... ...... ...... X..... DX.... X.....
 26 ...... ...... ...... .X.... XDX... .X....
 27 ...... ...... ...... ..X... .XDX.. ..X...
 28 ...... ...... ...... ...X.. ..XDX. ...X..
 29 ...... ...... ...... ....X. ...XDX ....X.
 30 ...... ...... ...... .....X ....XD .....X

 31 ...... ...... ...... ...... X..... DX....
 32 ...... ...... ...... ...... .X.... XDX...
 33 ...... ...... ...... ...... ..X... .XDX..
 34 ...... ...... ...... ...... ...X.. ..XDX.
 35 ...... ...... ...... ...... ....X. ...XDX
 36 ...... ...... ...... ...... .....X ....XD
```

```
------------------
 PROCEDURE MODULE
------------------

    D I S P L A Y   M A T R I X   P A T T E R N

    EXECUTION SUCCESSFUL


  M    N       EXACT EIGENVALUES

  1    1          .198665E+01
  1    2          .472188E+01
  1    3          .870329E+01
  1    4          .131223E+02
  1    5          .171037E+02
  1    6          .198589E+02
  2    1          .472188E+01
  2    2          .747710E+01
  2    3          .114585E+02
  2    4          .158775E+02
  2    5          .198590E+02
  2    6          .228142E+02
  3    1          .870329E+01
  3    2          .114585E+02
  3    3          .154399E+02
  3    4          .198590E+02
  3    5          .238404E+02
  3    6          .285956E+02
  4    1          .131223E+02
  4    2          .158775E+02
  4    3          .198590E+02
  4    4          .242780E+02
  4    5          .282594E+02
  4    6          .310146E+02
  5    1          .171037E+02
  5    2          .198590E+02
  5    3          .238404E+02
```

```
     5    4          .282594E+02
     5    5          .322408E+02
     5    6          .349960E+02
     6    1          .198589E+02
     6    2          .226142E+02
     6    3          .265956E+02
     6    4          .310146E+02
     6    5          .349960E+02
     6    6          .377512E+02


 ------------------
 PROCEDURE MODULE
 ------------------

    C O M P U T E    E I G E N V A L U E S

    SCALE FACTOR      .101321E+00


    N          E I G E N V A L U E              MAGNITUDE        ANGLE/PI

    1        .377513E+02      .000000E+00      .377513E+02      .000000
    2        .349961E+02      .000000E+00      .349961E+02      .000000
    3        .349961E+02      .000000E+00      .349961E+02      .000000
    4        .322408E+02      .000000E+00      .322408E+02      .000000
    5        .310147E+02      .000000E+00      .310147E+02      .000000
    6        .310147E+02      .000000E+00      .310147E+02      .000000
    7        .282594E+02      .000000E+00      .282594E+02      .000000
    8        .282594E+02      .000000E+00      .282594E+02      .000000
    9        .265956E+02      .000000E+00      .265956E+02      .000000
   10        .265956E+02      .000000E+00      .265956E+02      .000000
   11        .242780E+02      .000000E+00      .242780E+02      .000000
   12        .238404E+02      .000000E+00      .238404E+02      .000000
   13        .238404E+02      .000000E+00      .238404E+02      .000000
   14        .226142E+02      .000000E+00      .226142E+02      .000000
   15        .226142E+02      .000000E+00      .226142E+02      .000000
   16        .198590E+02      .000000E+00      .198590E+02      .000000
   17        .198589E+02      .000000E+00      .198589E+02      .000000
   18        .198589E+02      .000000E+00      .198589E+02      .000000
   19        .198589E+02      .000000E+00      .198589E+02      .000000
   20        .198589E+02      .000000E+00      .198589E+02      .000000
   21        .198589E+02      .000000E+00      .198589E+02      .000000
   22        .171038E+02      .000000E+00      .171038E+02      .000000
   23        .171037E+02      .000000E+00      .171037E+02      .000000
   24        .158775E+02      .000000E+00      .158775E+02      .000000
   25        .158775E+02      .000000E+00      .158775E+02      .000000
   26        .154399E+02      .000000E+00      .154399E+02      .000000
   27        .131223E+02      .000000E+00      .131223E+02      .000000
   28        .131223E+02      .000000E+00      .131223E+02      .000000
   29        .114585E+02      .000000E+00      .114585E+02      .000000
   30        .114585E+02      .000000E+00      .114585E+02      .000000
   31        .870330E+01      .000000E+00      .870330E+01      .000000
   32        .870329E+01      .000000E+00      .870329E+01      .000000
   33        .747711E+01      .000000E+00      .747711E+01      .000000
   34        .472190E+01      .000000E+00      .472190E+01      .000000
   35        .472188E+01      .000000E+00      .472188E+01      .000000
   36        .196667E+01      .000000E+00      .196667E+01      .000000


 ------------------
 INDEXING  MODULE
 ------------------

    R E D - B L A C K

    RBNDX BEGINNING EXECUTION
    RBNDX EXECUTION SUCCESSFUL
```

```
-----------------
PROCEDURE MODULE
-----------------

   D I S P L A Y   M A T R I X   P A T T E R N



                 1          2         3
    123456789012345678 901234567890123456

  1 D................. ..............X..X
  2 .D................ .............X..XX
  3 ..D............... ............X..XX.
  4 ...D.............. ...........X.XX..X
  5 ....D............. ..........X.XX..X.
  6 .....D............ .........X..X..X..
  7 ......D........... ........X..X..X...
  8 .......D.......... .......X..XX.X....
  9 ........D......... ......X..XX.X.....
 10 .........D........ .....X.XX..X......
 11 ..........D....... ....X.XX..X.......
 12 ...........D...... ...X..X..X........
 13 ............D..... ..X..X..X.........
 14 .............D.... .X..XX.X..........
 15 ..............D... X..XX.X...........
 16 ...............D.. .XX..X............
 17 ................D. XX..X.............
 18 .................D X..X..............

 19 ..............X.XX D.................
 20 .............X.XX. .D................
 21 ............X..X.. ..D...............
 22 ...........X..X..X ...D..............
 23 ..........X..XX.X. ....D.............
 24 .........X..XX.X.. .....D............
 25 ........X.XX..X... ......D...........
 26 .......X.XX..X.... .......D..........
 27 ......X..X..X..... ........D.........
 28 .....X..X..X...... .........D........
 29 ....X..XX.X....... ..........D.......
 30 ...X..XX.X........ ...........D......
 31 ..X.XX..X......... ............D.....
 32 .X.XX..X.......... .............D....
 33 X..X..X........... ..............D...
 34 ..X..X............ ...............D..
 35 .XX.X............. ................D.
 36 XX.X.............. .................D




-----------------
PROCEDURE MODULE
-----------------

   D I S P L A Y   M A T R I X   P A T T E R N

   EXECUTION SUCCESSFUL


-----------------
INDEXING  MODULE
-----------------

   N E S T E D     D I S S E C T I O N
```

```
    5-PT NESTED DISSECTION BEGINNING EXECUTION
    5-PT NESTED DISSECTION EXECUTION SUCCESSFUL


-----------------
PROCEDURE MODULE
-----------------

    D I S P L A Y   M A T R I X   P A T T E R N



             1         2         3
    123456789012345678901234567890123456

  1 D  X    XX
  2  D X                              XX
  3   DX     X                       XX
  4 XXXD
  5     D  XXX
  6      D X X                      XX
  7       DX                       XX
  8     XXXD
  9 X   X   D
 10 X X XX   D
 11           D     X                  X
 12            D   XX                 XX
 13             D  X              X  XX
 14              D XX X
 15               DX  X          XX
 16            XXXXD
 17           XX X  D
 18                  DX         XX
 19              XX  XD
 20                    D   X      X XX
 21                     D  X X     XX
 22                      D X   X XX
 23                       DX X X
 24                    XXXXD
 25                         DX     X
 26                     X X XD
 27                           DXXX
 28                      XX   XD
 29                  X        X D
 30               X  X   X    X  D
 31             X X    X X        D
 32       X             X   X      D
 33      XX            XX           D
 34   X  X      X      X             D
 35  XX        XX                     D
 36  X        XX                       D



-----------------
PROCEDURE MODULE
-----------------

    D I S P L A Y   M A T R I X   P A T T E R N

    EXECUTION SUCCESSFUL
```

```
---------------------
 ELLPACK  77  OUTPUT
---------------------


        ++++++++++++++++++++++++
        +                      +
        +    EXECUTION  TIMES  +
        +                      +
        ++++++++++++++++++++++++

    MODULE NAME                     SECONDS
    -----------------------------------

    5-POINT STAR                        .12
    NATURAL                             .02
    DISPLAY MATRIX PATTERN              .45
    EIGENVALUES                        3.65
    RED-BLACK                           .02
    DISPLAY MATRIX PATTERN              .40
    NESTED DISSECTION                   .07
    DISPLAY MATRIX PATTERN              .35
    TOTAL TIME                         5.38
```

EXAMPLE 4.D3 Nonlinear PDE Solution by Picard iteration.

An ELLPACK program is shown below for the problem

$$u_{xx} + u_{yy} = u^2(x^2+y^2)\,e^{-xy}$$

on the unit square with boundary values so the true solution is e^{xy}. The method of Picard (or fixed point iteration) is used to solve this non-linear problem through a sequence of linear ones. HODIE higher order finite differences and Gauss elimination are used and the solution is obtained quickly. See Section 5.B for another example of using Newton iteration with ELLPACK for a nonlinear problem.

```
*         ***************************************************
*         *                                                 *
*         *  EXAMPLE ELLPACK PROGRAM 4.D3                   *
*         *                                                 *
*         *  REMARKS                                        *
*         *      NONLINEAR POISSON PROBLEM WITH U**2 ON RIGHT *
*         *      SIDE.  FIXED POINT ITERATION IS USED.      *
*         *                                                 *
*         *                                                 *
*         ***************************************************
*

EQU.          UXX  +  UYY  =  (X**2 + Y**2) * EXP(-X*Y) * U(X,Y)**2

BOUND.        U = 1.0      ON  X = 0.0
```

```
                                    ON  Y = 0.0
                     U = EXP(Y)  ON  X = 1.0
                     U = EXP(X)  ON  Y = 1.0

 GRID.           7  X POINTS  $  7  Y POINTS

 OPT.            OLDU = 1 $ MEMORY

 FOR.
 C       INITIALIZE U AND INDEXING VECTORS
 C       ONE GETS BETTER RESULTS IF U IS INITIALIZED BY
 C
 C       TRIPLE.    INTERPOLATE BOUNDARY  CONDITIONS BY BLENDING
 C
 C       OR BY
 C
 C       TRIPLE.    INTIALIZE U(U=START)
 C
 C       WHERE START(X,Y) = 1 + XY
 C
 C       BUT THE CURRENT EXAMPLE SHOWS HOW ELLPACK INTERNAL VARIABLES
 C       MAY BE USED BY SOMEONE KNOWLEDGEABLE ABOUT ELLPACK
 C
 C       THIS WILL BE DONE WHEN POSSIBLE

         DO 10 I = 1,I1NEQN
            R1UNKN(I) = 1.0 + I*.001
            I1ENDX(I)  = I
            I1UNDX(I) = I
    10 CONTINUE

         DO 50 I = 1,6

            IF (I .GT. 1) I1LEVL = 0
            WRITE(6,20) I
    20      FORMAT(//6X,10(1H*),11HITERATION = ,I3,2X,20(1H*)/)

DIS.        HODIE ACF
IND.        AS IS
SOL.        BAND GE

OUT.         MAX(ERROR)

FOR.
            DIFMAX = 0.0
            DO 30 J = 1,I1NEQN
               R1WORK(J) = R1UNKN(J)-R1UNK1(J)
               DIFMAX = AMAX1(DIFMAX,ABS(R1WORK(J)))
    30      CONTINUE
            WRITE(6,40) DIFMAX,(J,R1WORK(J),J=1,I1NEQN)
    40      FORMAT(6X,25HMAX CHANGE IN UNKNOWNS = , E16.5,3X,
      A              18H  DIFFERENCES ARE / (5X,5(I3,F10.6)))

    50 CONTINUE

SUB.
        FUNCTION TRUE(X,Y)
           TRUE = EXP(X*Y)
        RETURN
        END

END.

     APPROXIMATE MEMORY REQUIREMENTS

     WORKSPACE               99        GRID LINES            15
     LINEAR EQNS            600        UNKNOWNS              50
     INTERPOLATION          209        DOMAIN INFO            0
     AMATRX,BVECTR          200        TOTAL MEMORY        1173
```

```
      SYMBOL TABLE INPUT TIME  2.63 SECONDS
      PROGRAM PROCESSING TIME  1.43 SECONDS
         TEMPLATE OUTPUT TIME  2.42 SECONDS
                   TOTAL TIME  6.48 SECONDS
```

Output of ELLPACK run:

```
       **********ITERATION =  1   *******************



 ------------------------
  DISCRETIZATION  MODULE
 ------------------------

    H O D I E - H E L M H O L T Z

    DOMAIN                                RECTANGLE
    X INTERVAL               .000E+00,   .100E+01
    Y INTERVAL               .000E+00,   .100E+01
    DISCRETIZATION                          UNIFORM
    GRID                                    7 X   7
    HX                                     .167E+00
    HY                                     .167E+00
    OUTPUT LEVEL                                  1
    METHOD CHOSEN                                41
    NUMBER OF EQUATIONS                          25
    MAX NO. OF UNKNOWNS PER EQ.                   9
    EXECUTION SUCCESSFUL


 -----------------
  INDEXING  MODULE
 -----------------

    N A T U R A L

    NUMBER OF EQUATIONS                          25
    EQUATIONS/UNKNOWNS NUMBERED
        IN ORDER GENERATED
    EXECUTION SUCCESSFUL


 -----------------
  SOLUTION  MODULE
 -----------------

    L I N P A C K    S P D    B A N D

    NUMBER OF ROWS                                7
    NUMBER OF COLUMNS                            25
    NUMBER OF UPPER CO-DIAGONALS                  6
    LINPACK BAND GIVES 2 TIMINGS
        SETUP TIME AND SOLUTION TIME
    EXECUTION SUCCESSFUL


 --------------------
  ELLPACK  77  OUTPUT
 --------------------

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                          +
    +  MAX( ABS(ERROR ) ) ON   7 X   7 GRID =  .7069170E-01    +
+                                                   +
+++++++++++++++++++++++++++++++++++++++++++++++++++++
   MAX CHANGE IN SOLUTION =        .20526E+01     DIFFERENCES ARE
  1  1.037505  2  1.074953  3  1.111603  4  1.145313  5  1.171595
  6  1.074953  7  1.150476  8  1.225777  9  1.297412 10  1.358045
 11  1.111603 12  1.225777 13  1.342963 14  1.459678 15  1.566834
 16  1.145313 17  1.297412 18  1.459678 19  1.630315 20  1.799959
 21  1.171595 22  1.358044 23  1.566834 24  1.799959 25  2.052578

   ***********ITERATION =  2  ********************
```

```
--------------------
ELLPACK  77  OUTPUT
--------------------

+++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                   +
+  MAX( ABS(ERROR ) ) ON   7 X   7 GRID =  .4932761E-02   +
+                                                   +
+++++++++++++++++++++++++++++++++++++++++++++++++++++
   MAX CHANGE IN SOLUTION =        .75624E-01     DIFFERENCES ARE
  1  -.009915  2  -.018959  3  -.026280  4  -.029503  5  -.023816
  6  -.018959  7  -.035174  8  -.047491  9  -.051865 10  -.040185
 11  -.026280 12  -.047491 13  -.063184 14  -.068624 15  -.053167
 16  -.029503 17  -.051865 18  -.068624 19  -.075624 20  -.060599
 21  -.023816 22  -.040185 23  -.053167 24  -.060599 25  -.052609

   ***********ITERATION =  3  ********************
```

```
--------------------
ELLPACK  77  OUTPUT
--------------------

+++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                   +
+  MAX( ABS(ERROR ) ) ON   7 X   7 GRID =  .3271103E-03   +
+                                                   +
+++++++++++++++++++++++++++++++++++++++++++++++++++++
   MAX CHANGE IN SOLUTION =        .52596E-02     DIFFERENCES ARE
  1    .000615  2    .001210  3    .001686  4    .001821  5    .001295
  6    .001209  7    .002366  8    .003281  9    .003521 10    .002486
 11    .001686 12    .003281 13    .004534 14    .004864 15    .003442
 16    .001821 17    .003521 18    .004864 19    .005260 20    .003778
 21    .001295 22    .002486 23    .003441 24    .003778 25    .002790

   ***********ITERATION =  4  ********************
```

```
--------------------
ELLPACK  77  OUTPUT
--------------------

+++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                   +
+  MAX( ABS(ERROR ) ) ON   7 X   7 GRID =  .1502037E-04   +
+                                                   +
+++++++++++++++++++++++++++++++++++++++++++++++++++++
   MAX CHANGE IN SOLUTION =        .34201E-03     DIFFERENCES ARE
  1  -.000040  2  -.000079  3  -.000110  4  -.000117  5  -.000081
  6  -.000079  7  -.000156  8  -.000216  9  -.000230 10  -.000159
 11  -.000110 12  -.000216 13  -.000300 14  -.000320 15  -.000221
```

```
     16  -.000117 17  -.000230 18  -.000320 19  -.000342 20  -.000238
     21  -.000081 22  -.000159 23  -.000221 24  -.000238 25  -.000167

       ***********ITERATION =  5  **********************


--------------------
 ELLPACK  77  OUTPUT
--------------------

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                    +
    +  MAX( ABS(ERROR ) ) ON   7 X   7 GRID =  .7033346E-05    +
    +                                                    +
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++
       MAX CHANGE IN SOLUTION =          .22173E-04      DIFFERENCES ARE
      1     .000003  2     .000005  3     .000007  4    .000008  5     .000005
      6     .000005  7     .000010  8     .000014  9    .000015 10     .000010
     11     .000007 12     .000014 13     .000020 14    .000021 15     .000014
     16     .000008 17     .000015 18     .000021 19    .000022 20     .000015
     21     .000005 22     .000010 23     .000014 24    .000015 25     .000010

       ***********ITERATION =  6  **********************


--------------------
 ELLPACK  77  OUTPUT
--------------------

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                    +
    +  MAX( ABS(ERROR ) ) ON   7 X   7 GRID =  .5722046E-05    +
    +                                                    +
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++
       MAX CHANGE IN SOLUTION =          .15497E-05      DIFFERENCES ARE
      1     .000000  2     .000000  3     .000000  4   -.000001  5     .000000
      6     .000000  7    -.000001  8    -.000001  9   -.000001 10    -.000001
     11     .000000 12    -.000001 13    -.000002 14   -.000002 15    -.000001
     16     .000000 17    -.000001 18    -.000001 19   -.000001 20    -.000001
     21     .000000 22    -.000001 23    -.000001 24   -.000001 25    -.000001
```

We discuss four of the more interesting points about this program.

1. **Treatment of the nonlinearity.** Note that the ELLPACK function $U(X,Y)$ is used directly in the right side of the PDE. This facility can be used for more complex PDEs, for example

```
7.*UXX + (U(X,Y)**2+1.)*UYY + SIN(UX(X,Y))*UX - UY(X,Y)*U = 0
```

for the equation

$$7u_{xx} + (1+u^2)u_{yy} + sin\,(u_x)\,u_x - u_y\,u = 0$$

The discretization module uses the current definition of $U(X,Y)$, $UX(X,Y)$, etc. in forming the linear system for the problem. This means that $U(X,Y)$, etc. must be initialized, see the next point for one simple approach.

There is a **word of caution about nonlinearities**, it is not guaranteed that the functions $U(X,Y)$ are not disturbed during the discretizations. This technique works normally in most situations, but there are cases where it fails because the discretization module may change something about how the solution $U(X,Y)$ is computed during its execution. For example, suppose that one changes the grid size between two uses of 5 POINT STAR. After the first use, the unknowns are stored in table corresponding to the first grid. Once the grid is changed this table no longer corresponds to the existing grid. When 5 POINT STAR is used again, it sets variables to determine how to evaluate the new solution. At some point in this sequence $U(X,Y)$ becomes improperly defined, this might occur right in the middle of a discretization module's execution. If one suspects there may be a problem like this, the information about $U(X,Y)$, etc should be moved to user defined arrays and then used from there. An example of this is given in Chapter 5.

The **fixed point iteration** method is used to handle the nonlinearity. This method is very simple to implement in ELLPACK, in this case it corresponds to the iteration

$$U_{xx}^{(n+1)} + U_{yy}^{(n+1)} = (U^{(n)})^2(x^2 + y^2)\,e^{-xy} \qquad n = 0,1,2,...$$

where one has a linear problem to obtain $U^{(n+1)}$ from $U^{(n)}$. One could also attempt to use the iterations

$$U_{xx}^{(n+1)} + U_{yy}^{(n+1)} - U^{(n)}(x^2 + y^2)\, e^{-xy}\ U^{(n+1)} = 0$$

in much the same way. The strength of fixed point iteration is its simplicity. Its weakness is that one cannot predict whether it will work and, if it does work, how well it will work. In this instance it works well. When one formulation of the iteration fails, others should be attempted.

2. **Initialization of the iteration.** The unknown function $U(X,Y)$ is initialized indirectly by setting the *ith* unknown to 1 + 1/1000. This is done here merely to illustrate how one can use detailed information about ELLPACK's internal structure to set various things. Here one knows that there are I1NEQN unknowns in the array R1UNKN; one also initializes the indexing arrays I1ENDX and I1UNDX to the identity. While this is a more complicated and less effective way for this example than discussed below, it is a useful technique in some more complex situations. One could use the ELLPACK triple INTERPOLATE BOUNDARY CONDITIONS BY BLENDING to initialize $U(X,Y)$. Once this TRIPLE is executed then $U(X,Y)$, $UX(X,Y)$, etc. are defined and, in fact, the blending function method used frequently produces very good approximations. If one knows a good approximation to $U(X,Y)$, say START then one can use the ELLPACK statement INITIALIZE UNKNOWN (U = START). Both of these TRIPLES define U just as though a numerical method were used to approximate U with a more standard set of ELLPACK modules.

3. **Testing for convergence.** A simple convergence test is used based on the maximum changes in the unknowns from one iteration to another. In order to make this test, one needs to have both the current and previous unknowns available. The option OLDU=1 creates the array R1UNK1 which always contains the values of the previous unknowns. This allows the test to be made even if one

does not understand how the unknowns are used by the discretization module to represent $U(X,Y)$.

An alternate (and simpler) convergence test would be to compute the maximum change on the grid of the solution itself. The option OLDU = 1 also creates the function $U1(X,Y)$ and one could replace the DO 30 loop by

```
     DO 30 I = 1,5
        X = I/6.
        DO 30 J = 1,5
           Y = J/6.
           DIFMAX = AMAX1(DIFMAX,ABS(U(X,Y)-U1(X,Y)))
  30 CONTINUE
```

One can easily modify this to store the individual differences in the workspace array R1WORK for printing with FORMAT 40.

4. **Use of Workspace.** The Fortran code is written at the end of the DO-loop to use a temporary array. The standard ELLPACK array R1WORK can be used, its size I1MWRK is obtained with the option MEMORY. If a larger temporary array is needed, the size of R1WORK can be set to n with the option MAX-WORKSPACE=n.

EXAMPLE 4.D4 Nonrectangular domain with a hole

This example illustrates the use of the HOLE segment and non-rectangular domains.

```
*         *******************************************************
*         *                                                     *
*         *  EXAMPLE ELLPACK PROGRAM 4.D4                       *
*         *                                                     *
*         *  REMARKS                                            *
*         *      THIS PROGRAM USES THE HOLE FEATURE IN ELLPACK.  *
*         *      THE REGION IS BETWEEN TWO CONFOCAL ELLIPSES.    *
*         *                                                     *
*         *******************************************************
*
EQ.
          UXX  +  UYY  =  0.0

BO.       U = 0.  ON  X = COSH(3.0)*SIN(T), Y = SINH(3.0)*COS(T)
                                        FOR  T = 0.0 TO 2*PI
```

```
HO.        U = 1.  ON  X = COSH(2.3)*SIN(T), Y = SINH(2.3)*COS(T)
                                          FOR  T = 0.0 TO 2*PI

GR.        17 X POINTS,   -COSH(3.0) TO COSH(3.0)
           17 Y POINTS,   -SINH(3.0) TO SINH(3.0)

DI.        5 POINT STAR
IN.        AS IS
SO.        BAND GE

OP.        TIME $ MEMORY

OU.        PLOT-DOMAIN
           MAX(TRUE) $ MAX(U) $ MAX(ERROR)

SUBPROGRAMS.
      FUNCTION TRUE(X,Y)
      R1    = SQRT((X-1.0)**2+Y**2)
      R2    = SQRT((X+1.0)**2+Y**2)
      U     = ACOSH(0.5*(R1+R2))
      TRUE = (3.0-U)/(3.0-2.3)
      RETURN
      END
      FUNCTION ACOSH(X)
      ACOSH = ALOG(X+SQRT(X**2-1.0))
      RETURN
      END

END.
```

```
    APPROXIMATE MEMORY REQUIREMENTS

    WORKSPACE           331       GRID LINES          35
    LINEAR EQNS        4624       UNKNOWNS           289
    INTERPOLATION       757       DOMAIN INFO        289
    AMATRX,BVECTR     13583       TOTAL MEMORY     19908


       SYMBOL TABLE INPUT TIME  2.50 SECONDS
       PROGRAM PROCESSING TIME   .85 SECONDS
          TEMPLATE OUTPUT TIME  2.20 SECONDS
                    TOTAL TIME  5.55 SECONDS
```

Output of ELLPACK run:

```
-----------------
 DOMAIN  PROCESSOR
-----------------

   DOMAIN PROCESSOR BEGINNING EXECUTION
   FOUND  63 BOUNDARY POINTS WHERE THE
    1 PIECES INTERSECT THE  17 X  17 GRID

   TIME TO PROCESS BOUNDARY              4.933
   TIME TO PROCESS INTERIOR               .100
   TOTAL PROCESSING TIME                 5.033


-----------------
 DOMAIN  PROCESSOR
-----------------

   DOMAIN PROCESSOR BEGINNING EXECUTION
   FOUND  32 BOUNDARY POINTS WHERE THE
    1 PIECES INTERSECT THE  17 X  17 GRID
    TIME TO PROCESS BOUNDARY              2.733
    TIME TO PROCESS INTERIOR               .033
    TOTAL PROCESSING TIME                 2.767


 ------------------------
  DISCRETIZATION  MODULE
 ------------------------

    5 - P O I N T    S T A R

    DOMAIN                      NON-RECTANGULAR
    UNIFORM GRID                       17 X  17
    HX                                .125E+01
    HY                                .125E+01
    OUTPUT LEVEL                              1
    BOUNDARY CONDITIONS
    PIECE  1                             TYPE 1
    PIECE  2                             TYPE 1
    NUMBER OF EQUATIONS                     146
    MAX NO. OF UNKNOWNS PER EQ.               5
    EXECUTION SUCCESSFUL


 ----------------
  INDEXING  MODULE
 ----------------

    N A T U R A L

    NUMBER OF EQUATIONS                     146
    EQUATIONS/UNKNOWNS NUMBERED
        IN ORDER GENERATED
    EXECUTION SUCCESSFUL


 ----------------
  SOLUTION  MODULE
 ----------------

    L I N P A C K    B A N D

    NUMBER OF ROWS                         40
    NUMBER OF COLUMNS                     146
    NUMBER OF LOWER CO-DIAGONALS           13
    NUMBER OF UPPER CO-DIAGONALS           13
    LINPACK BAND GIVES 2 TIMINGS
        SETUP TIME AND SOLUTION TIME
    EXECUTION SUCCESSFUL


 --------------------
  ELLPACK  78  OUTPUT
 --------------------

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                          +
    +  MAX( ABS(TRUE  ) ) ON  17 X  17 GRID =  .9797284E+00    +
    +                                                          +
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


 --------------------
  ELLPACK  78  OUTPUT
 --------------------

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                           +
    +  MAX( ABS(U      ) ) ON  17 X  17 GRID =  .9795400E+00    +
    +                                                           +
    +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


--------------------
 ELLPACK  78  OUTPUT
--------------------

    +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                           +
    +  MAX( ABS(ERROR ) ) ON  17 X  17 GRID =  .1475811E-02     +
    +                                                           +
    +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++



--------------------
 ELLPACK  77  OUTPUT
--------------------

        +++++++++++++++++++++++++
        +                       +
        +    EXECUTION  TIMES   +
        +                       +
        +++++++++++++++++++++++++

    MODULE NAME                    SECONDS
    ----------------------------------

    DOMAIN                            5.03
    HOLE                              2.78
    5-POINT STAR                       .28
    NATURAL                            .02
    LINPACK BAND SETUP                 .18
    LINPACK BAND                       .73
    PLOT DOMAIN                        .30
    MAX                                .27
    MAX                                .70
    MAX                                .45
    TOTAL TIME                       10.80
```

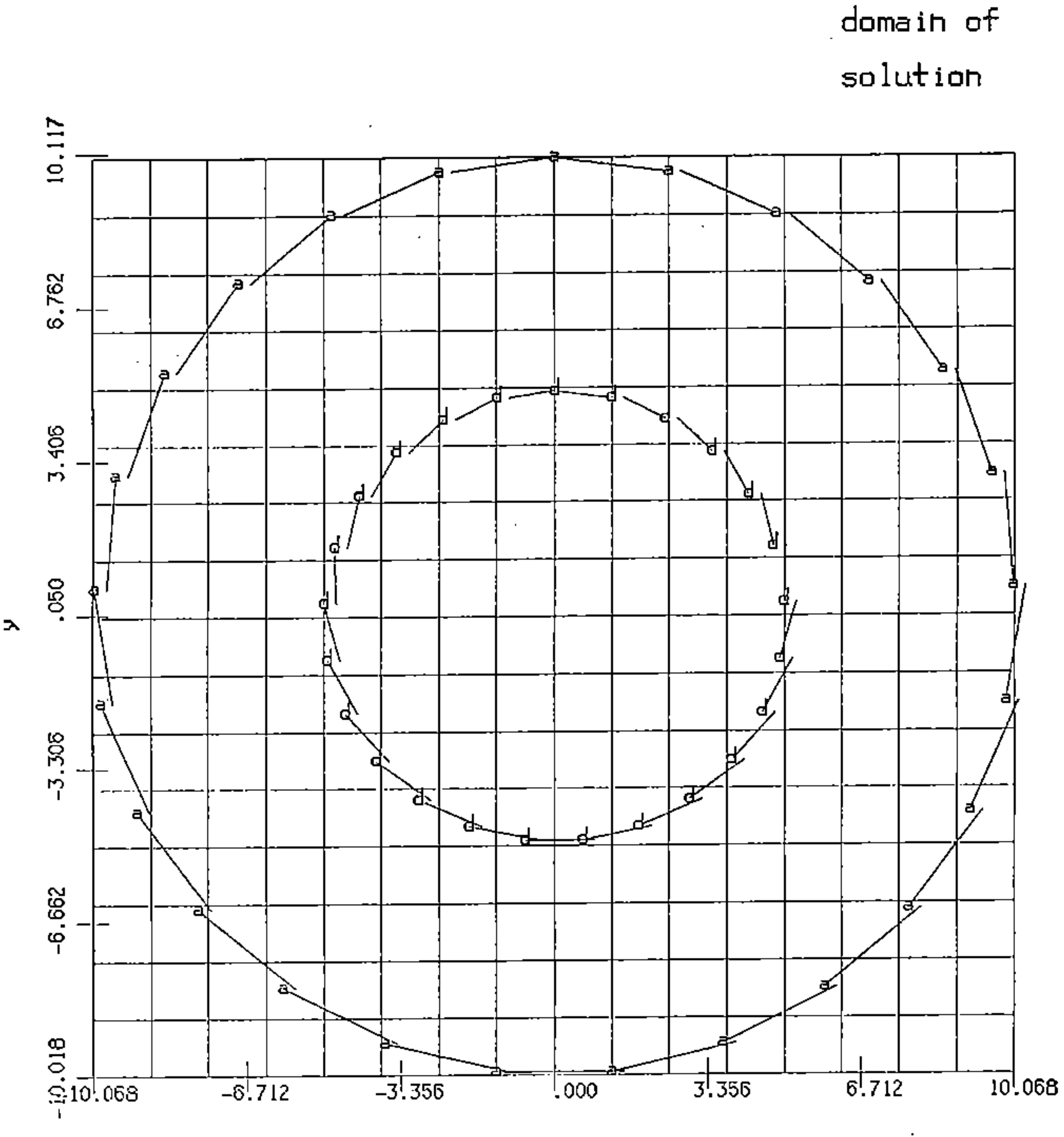

Figure 4.2. The contour plot produced by PLOT(DOMAIN) in the example ELLPACK program 4.D4.

CHAPTER 5: EXTENDING ELLPACK TO NON-STANDARD PROBLEMS

ELLPACK can be used to solve or study problems where its "automatic" problem solving capabilities do not apply. To use ELLPACK in this way requires that one has an understanding of both the ELLPACK system and the numerical methods involved. Even more complicated applications depend on knowing some of the details of the implementations in the ELLPACK modules. We illustrate this use of ELLPACK with five examples.

1. *A problem with a double-valued boundary condition along a slit.* **Such problems arise when their objects are placed in electrical fields. Many problems of this type can be solved using ELLPACK once one learns the technique.**

2. *Diffusion problem with interior interface conditions.* **The melting of a metallic alloy introduces an interface with a derivative jump condition. The 5-POINT STAR equations are modified along the liquid-solid interface to be this condition.**

3. *Three nonlinear problems.* **Newton iteration methods for handling nonlinearities can be implemented easily in ELLPACK The examples given show very rapid convergence and two are real world applications.**

4. *A time dependent problem.* **Consider the parabolic problem**

$$u_t = Lu$$

where L is an elliptic operator in two or three variables. Many numerical methods for such problems are implicit so they can take large time steps. At each time step these methods solve an elliptic boundary value problem. Several of these methods can be concisely formulated in ELLPACK to use its elliptic problem solving capabilities. We illustrate this technique for the PDE

$$u_t = uy^2u_{xx} + u_{yy} + 2+tan(x+y+t))uy + u + f(x,y,t)$$

5. *The transitor equations.* **This problem involves three simultaneous, highly nonlinear elliptic equations. This problem arises from a model of the electric field in a transistor and is difficult to solve. A particular iteration is defined which might loosely be called a Newton-Jacobi iteration. Jacobi iteration is used in going from equation to equation and Newton's method is used for the nonlinearities.**

5.A SPECIAL INTERIOR BOUNDARY CONDITIONS

ELLPACK has the capability to handle auxiliary conditions along curves inside the domain. These might be conditions on 'slits' as occurs at a thin plates is inserted in an electric field or at the interface between a solid and liquid. The ARC segment allows one to specify single valued boundary conditions in a straight forward way. This example also illustrates the various output that one can obtain from ELLPACK. The output TABLE-BOUNDARY is primarily useful for people who want to know how ELLPACK works internally. This technique is illustrated in program 5.A2.

```
*        *****************************************************
*        *                                                   *
*        *  EXAMPLE ELLPACK PROGRAM 5.A1                     *
*        *                                                   *
*        *  REMARKS                                          *
*        *      THIS PROGRAM USES THE ARC  FEATURE IN ELLPACK.  *
*        *      THE REGION IS BOUNDED BY CONFOCAL ELLIPSES.  *
*        *      THE SLIT IS A DEGENERATE ELLIPSE.            *
*        *                                                   *
*        *****************************************************

OPT.     TIME

EQ.      UXX  +  UYY  =  0.0

BOUND.   U = 0.  ON  X = COSH(2.0)*SIN(T), Y = SINH(2.0)*COS(T)
.                                          FOR  T = 0.0 TO 2*PI

ARC.     U = 1.  ON  LINE -1.0, 0.0  TO  1.0, 0.0

GRID.    11 X POINTS   -COSH(2.0)  TO COSH(2.0)
         11 Y POINTS   -SINH(2.0)  TO SINH(2.0)
```

```
DIS.      5-POINT STAR
IND.      AS IS
SOL.      BAND GE

OUT.      SUMMARY(U)    $ MAX(ERROR)
          TABLE-DOMAIN $ TABLE-BOUNDARY
          PLOT-DOMAIN   $ PLOT(U)

SUBPROGRAMS.
      FUNCTION TRUE(X,Y)
      R1   = SQRT((X-1.0)**2+Y**2)
      R2   = SQRT((X+1.0)**2+Y**2)
      U    = ACOSH(0.5*(R1+R2))
      TRUE = (2.0-U)/2.0
      RETURN
      END
      FUNCTION ACOSH(X)
      ACOSH = ALOG(X+SQRT(X**2-1.0))
      RETURN
      END

END.


        SYMBOL TABLE INPUT TIME  2.68 SECONDS
      PROGRAM PROCESSING TIME  1.00 SECONDS
         TEMPLATE OUTPUT TIME  2.72 SECONDS
                   TOTAL TIME  6.40 SECONDS
```

Output of ELLPACK run (some output has been deleted for brevity):

```
------------------
 DOMAIN  PROCESSOR
------------------

    DOMAIN PROCESSOR BEGINNING EXECUTION
    FOUND  32 BOUNDARY POINTS WHERE THE
     1 PIECES INTERSECT THE  11 X  11 GRID

    TIME TO PROCESS BOUNDARY            2.533
    TIME TO PROCESS INTERIOR             .067
    TOTAL PROCESSING TIME               2.600


------------------
 DOMAIN  PROCESSOR
------------------

    DOMAIN PROCESSOR BEGINNING EXECUTION
    FOUND   4 BOUNDARY POINTS WHERE THE
     1 PIECES INTERSECT THE  11 X  11 GRID

    TIME TO PROCESS BOUNDARY             .033
    TIME TO PROCESS INTERIOR             .017
    TOTAL PROCESSING TIME                .050

*** 5 POINT STAR, AS IS AND BAND GE OUTPUT DELETED ***

-------------------
 ELLPACK  78  OUTPUT
-------------------

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                         +
    +  MAX( ABS(U      ) ) ON  11 X  11 GRID =  .1000000E+01   +
    +                                                         +
    +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


--------------------
 ELLPACK  78  OUTPUT
--------------------


        +++++++++++++++++++++++++++++++++++++++++++++++++++
        +                                                 +
        +    TABLE OF U       ON  11 X  11  GRID          +
        +                                                 +
        +++++++++++++++++++++++++++++++++++++++++++++++++++

*** TABLE(U) OUTPUT DELETED ***


--------------------
 ELLPACK  78  OUTPUT
--------------------

    +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                         +
    +  MAX( ABS(ERROR ) ) ON  11 X  11 GRID =  .8435743E-01   +
    +                                                         +
    +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


--------------------
 ELLPACK  78  OUTPUT
--------------------


    +++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                     +
    +    TABLE  OF  THE  POINT  TYPES  ON  11 X  11  GRID  +
    +                                                     +
    +++++++++++++++++++++++++++++++++++++++++++++++++++++++

         THE POINT XGRID(1), YGRID(1) IS AT THE LOWER LEFT.
         ---------------------------------------------------

11 *        0        0    -4029    -4030    -6001        1   -12001    -4003    -4004        0        0
10 *        0    -6028       29     9029     1031     1001     1002     3003        4   -12004        0
 9 *    -2028       28     9028      999      999      999      999      999     3004        5    -8005
 8 *    -2027     9027      999      999      999      999      999      999      999     3005    -8006
 7 *    -6025     8026      999      999     4035     4036     4037      999      999     2007   -12007
 6 *       25     8025      999     2034       35       36       37     8037      999     2009        9
 5 *    -3023     8023      999      999     1035     1036     1037      999      999     2010    -9009
 4 *    -2022    12021      999      999      999      999      999      999      999     6011    -8011
 3 *    -2021       21    12020      999      999      999      999      999     6012       12    -8012
 2 *        0    -3020       20    12019     4018     4017     4015     6013       13    -9012        0
 1 *        0        0    -1020    -1019    -3017       17    -9015    -1014    -1013        0        0
   *
   ******* 1        2        3        4        5        6        7        8        9       10       11


--------------------
 ELLPACK  78  OUTPUT
--------------------


    +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                                         +
 +   TABLE  OF  THE  BOUNDARY  POINT  TYPES ON  11 X  11  GRID     +
 +                                                                 +
 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

NUMBER    XBOUND          YBOUND          BPARAM        PIECE   BPTYPE   BGRID      BNEIGH

   1        .000000       3.626860         .000000        1      CORN    11006          0
   2        .752439       3.553583         .201358        1      VERT    10007      10007
   3       1.504879       3.324072         .411517        1      VERT    10008      10008
   4       2.257318       2.901489         .643502        1      BOTH    10009       9009
   5       3.009757       2.176116         .927296        1      BOTH     9010       8010
   6       3.448109       1.450744        1.159279        1      HORZ     8010       8010
   7       3.686186        .725372        1.369441        1      HORZ     7010       7010
   8       3.762196        .007547        1.568716        1      VERT     6011          0
   9       3.762196        .000000        1.570796        1      BOTH     6011       6010
  10       3.686184       -.725372        1.772154        1      HORZ     5010       5010
  11       3.448109      -1.450744        1.982313        1      HORZ     4010       4010
  12       3.009757      -2.176116        2.214298        1      BOTH     3010       3009
  13       2.257318      -2.901488        2.498092        1      BOTH     2009       3009
  14       1.504879      -3.324073        2.730076        1      VERT     1008       2008
  15        .752439      -3.553584        2.940237        1      VERT     1007       2007
  16        .006181      -3.626860        3.139950        1      HORZ     1006          0
  17        .000000      -3.626860        3.141593        1      BOTH     1006       2006
  18       -.752439      -3.553584        3.342949        1      VERT     1005       2005
  19      -1.504878      -3.324073        3.553109        1      VERT     1004       2004
  20      -2.257317      -2.901488        3.785094        1      BOTH     2003       3003
  21      -3.009757      -2.176116        4.068888        1      BOTH     3002       4002
  22      -3.448110      -1.450744        4.300872        1      HORZ     4001       4002
  23      -3.686184       -.725372        4.511032        1      HORZ     5001       5002
  24      -3.762196       -.007156        4.710416        1      VERT     5001          0
  25      -3.762196        .000000        4.712389        1      BOTH     6001       6002
  26      -3.686184        .725372        4.913747        1      HORZ     7001       7002
  27      -3.448110       1.450744        5.123905        1      HORZ     8001       8002
  28      -3.009757       2.176116        5.355890        1      BOTH     9002       9003
  29      -2.257317       2.901489        5.639684        1      BOTH    10003       9003
  30      -1.504876       3.324074        5.871669        1      VERT    10004      10004
  31       -.752439       3.553583        6.081828        1      VERT    10005      10005
  32       -.005453       3.626860        6.281736        1      HORZ    11005          0
  33        .000000       3.626860        6.283185        1      JUMP    11006          0
  34      -1.000000        .000000         .000000        2      CORN     6004          0
  35       -.752439        .000000         .123780        2      BOTH     6005       5005
  36        .000000        .000000         .500000        2      BOTH     6006       5006
  37        .752430        .000000         .876220        2      BOTH     6007       5007
```

```
--------------------
ELLPACK  77  OUTPUT
--------------------


      ++++++++++++++++++++++++
      +                      +
      +    EXECUTION  TIMES  +
      +                      +
      ++++++++++++++++++++++++


   MODULE NAME                   SECONDS
   ----------------------------------

   DOMAIN                          2.62
   ARC                              .07
   5-POINT STAR                     .22
   NATURAL                          .05
   LINPACK BAND SETUP               .10
   LINPACK BAND                     .27
   MAX                              .43
   TABLE                            .57
   MAX                              .35
```

```
     TABLE DOMAIN                    .23
     TABLE BOUNDARY                  .45
     PLOT DOMAIN                     .23
     PLOT                           5.60
     TOTAL TIME                    11.35
```

To solve problems with double valued boundary conditions one must use the HOLE segment to place a very thin hole (slit or arc) in the domain and then specify boundary conditions on each side of the hole. Care must be taken at the ends of the hole so that the domain processor can follow the boundary. One should make the ends of the hole pointed and the ends of different pieces of the boundary.

```
*         ***********************************************************
*         *                                                         *
*         *  EXAMPLE ELLPACK PROGRAM 5.A2                           *
*         *                                                         *
*         *  REMARKS                                                *
*         *      THIS PROGRAM IS FOR A PROBLEM WITH AN INTERIOR TWO *
*         *      VALUED BOUNDARY CONDITION ON A SLIT. THE ARC FACILITY *
*         *      OF ELLPACK DOES NOT APPLY SO A HOLE IN THE SHAPE OF A *
*         *      LONG, VERY THIN DIAMOND IS USED INSTEAD. CARE MUST BE *
*         *      TAKEN IN DEFINING THE SLITS THIS WAY SO THE ELLPACK *
*         *      DOMAIN PROCESSOR DOES NOT GET LOST. DEFINING THIS SLIT *
*         *      AS A LONG, VERY THIN RECTANGLE OR ELLIPSE WILL PROBABLY *
*         *      FAIL. THE ELLPACK PLOT ROUTINES ALSO ARE INACCURATE IN *
*         *      THE NEIGHBORHOOD OF TWO-VALUED BOUNDARY CONDITIONS. *
*         *                                                         *
*         ***********************************************************
OPT.      TIME

EQ.       UXX  +  UYY  =  0.0

BOUND.    U = 0.   ON  X = COSH(2.0)*SIN(T), Y = SINH(2.0)*COS(T)
.                                      FOR  T = 0.0 TO 2*PI

HOLE.     U = 1.         ON  LINE -1.0, 0.0  TO  0.0,  0.010  TO   1.0,0.0
          U = 2.-X**2    ON  LINE  1.0, 0.0  TO  0.0, -0.010  TO  -1.0,0.0

GRID.     21 X POINTS    -COSH(2.0) TO COSH(2.0)
          21 Y POINTS    -SINH(2.0) TO SINH(2.0)

DIS.      5-POINT STAR
IND.      AS IS
SOL.      BAND GE

OUT.      MAX(U) $ PLOT(U)

END.
```

```
          SYMBOL TABLE INPUT TIME   2.67 SECONDS
          PROGRAM PROCESSING TIME   1.22 SECONDS
             TEMPLATE OUTPUT TIME   2.70 SECONDS
                       TOTAL TIME   6.58 SECONDS
```

Output of ELLPACK run:

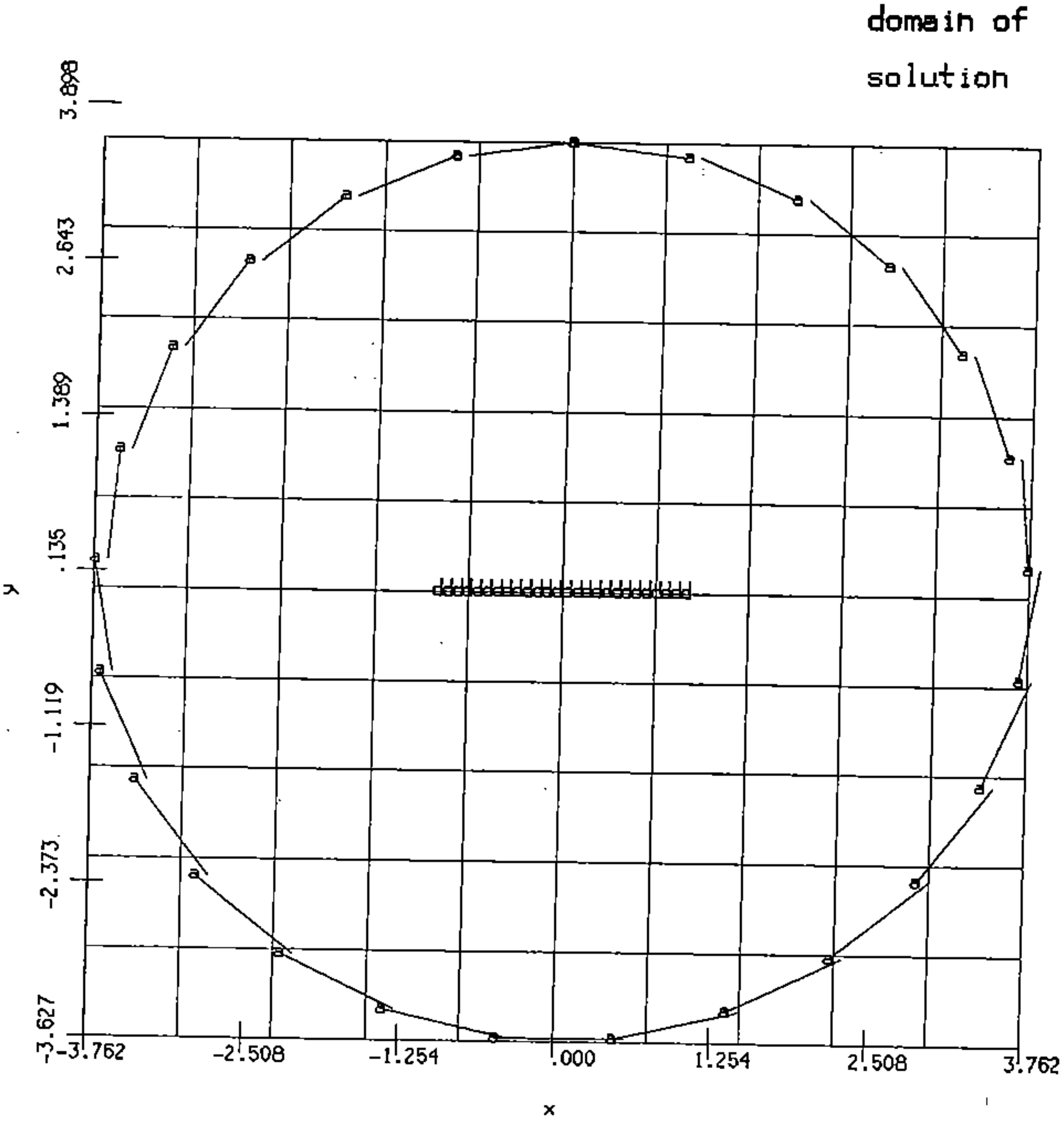

Figure 5.1 Problem domain for program 5.A1

```
------------------
 DOMAIN  PROCESSOR
------------------

    DOMAIN PROCESSOR BEGINNING EXECUTION
    FOUND  72 BOUNDARY POINTS WHERE THE
     1 PIECES INTERSECT THE  21 X  21 GRID

    TIME TO PROCESS BOUNDARY              5.167
    TIME TO PROCESS INTERIOR               .167
    TOTAL PROCESSING TIME                 5.333


------------------
 DOMAIN  PROCESSOR
------------------

    DOMAIN PROCESSOR BEGINNING EXECUTION
    FOUND  12 BOUNDARY POINTS WHERE THE
     4 PIECES INTERSECT THE  21 X  21 GRID

    TIME TO PROCESS BOUNDARY               .117
    TIME TO PROCESS INTERIOR               .017
    TOTAL PROCESSING TIME                  .133


----------------------
 DISCRETIZATION  MODULE
----------------------

    5 - P O I N T    S T A R

    DOMAIN                      NON-RECTANGULAR
    UNIFORM GRID                        21 X  21
    HX                                 .376E+00
    HY                                 .363E+00
    OUTPUT LEVEL                              1
    BOUNDARY CONDITIONS
    PIECE  1                             TYPE 1
    PIECE  2                             TYPE 1
    PIECE  3                             TYPE 1
    PIECE  4                             TYPE 1
    PIECE  5                             TYPE 1
    NUMBER OF EQUATIONS                     300
    MAX NO. OF UNKNOWNS PER EQ.               5
    EXECUTION SUCCESSFUL


----------------
 INDEXING  MODULE
----------------

    N A T U R A L

    NUMBER OF EQUATIONS                     300
    EQUATIONS/UNKNOWNS NUMBERED
        IN ORDER GENERATED
    EXECUTION SUCCESSFUL


----------------
 SOLUTION  MODULE
----------------

    L I N P A C K    B A N D

    NUMBER OF ROWS                           58
    NUMBER OF COLUMNS                       300
```

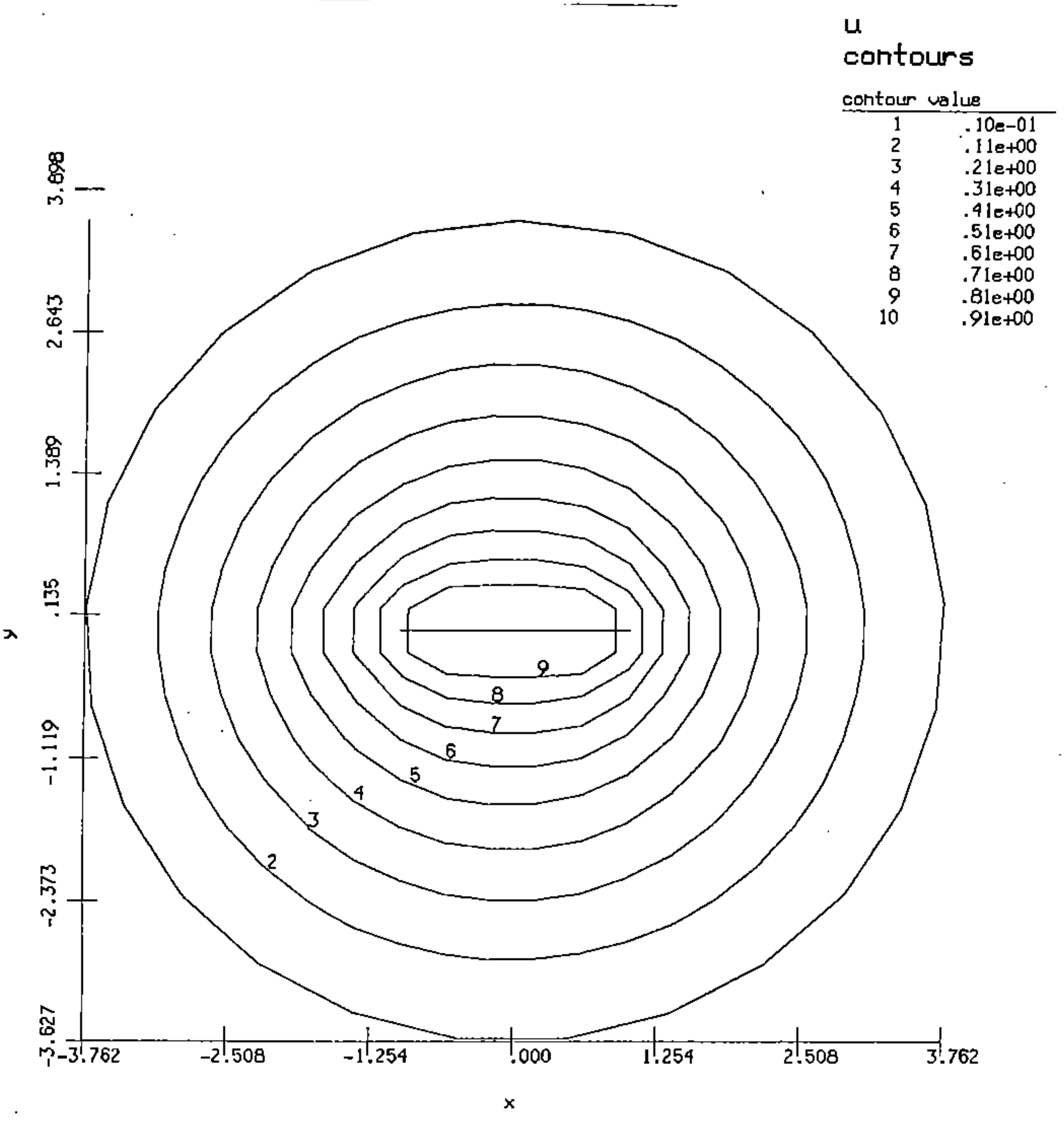

FIGURE 5.2 CONTOUR PLOT OF SOLUTION FOR PROGRAM 5.A1

```
    NUMBER OF LOWER CO-DIAGONALS            19
    NUMBER OF UPPER CO-DIAGONALS            19
    LINPACK BAND GIVES 2 TIMINGS
        SETUP TIME AND SOLUTION TIME
    EXECUTION SUCCESSFUL


 --------------------
 ELLPACK  78  OUTPUT
 --------------------

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    +                                                            +
    +  MAX( ABS(U      ) ) ON  21 X  21 GRID =  .1488397E+01     +
    +                                                            +
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++



 --------------------
 ELLPACK  77  OUTPUT
 --------------------

        +++++++++++++++++++++++++
        +                       +
        +    EXECUTION  TIMES   +
        +                       +
        +++++++++++++++++++++++++

   MODULE NAME                      SECONDS
   ------------------------------------

   DOMAIN                             5.33
   HOLE                                .17
   5-POINT STAR                        .43
   NATURAL                             .03
   LINPACK BAND SETUP                  .37
   LINPACK BAND                       3.03
   MAX                                1.00
   PLOT                               5.00
   TOTAL TIME                        15.52
```

5.B A TWO-PHASE DIFFUSION PROBLEM

Consider a rectangular container $0<x<1$, $-1/2<y<1$ filled with a metallic alloy. The sides of the container ($x=0$, $x=1$) are insulated, while the top of the container ($y=1$) is held at some fixed temperature above the melting point of the metal and the bottom ($y = 1/2$) is held at a constant temperature below the melting point. The vessel eventually contains both molten and solid metal, and we assume that the solid-liquid interface lies along the line $y=0$. In addition, we assume that the liquid metal is stirred by some external means. We wish to determine the steady-state temperature distribution of this system.

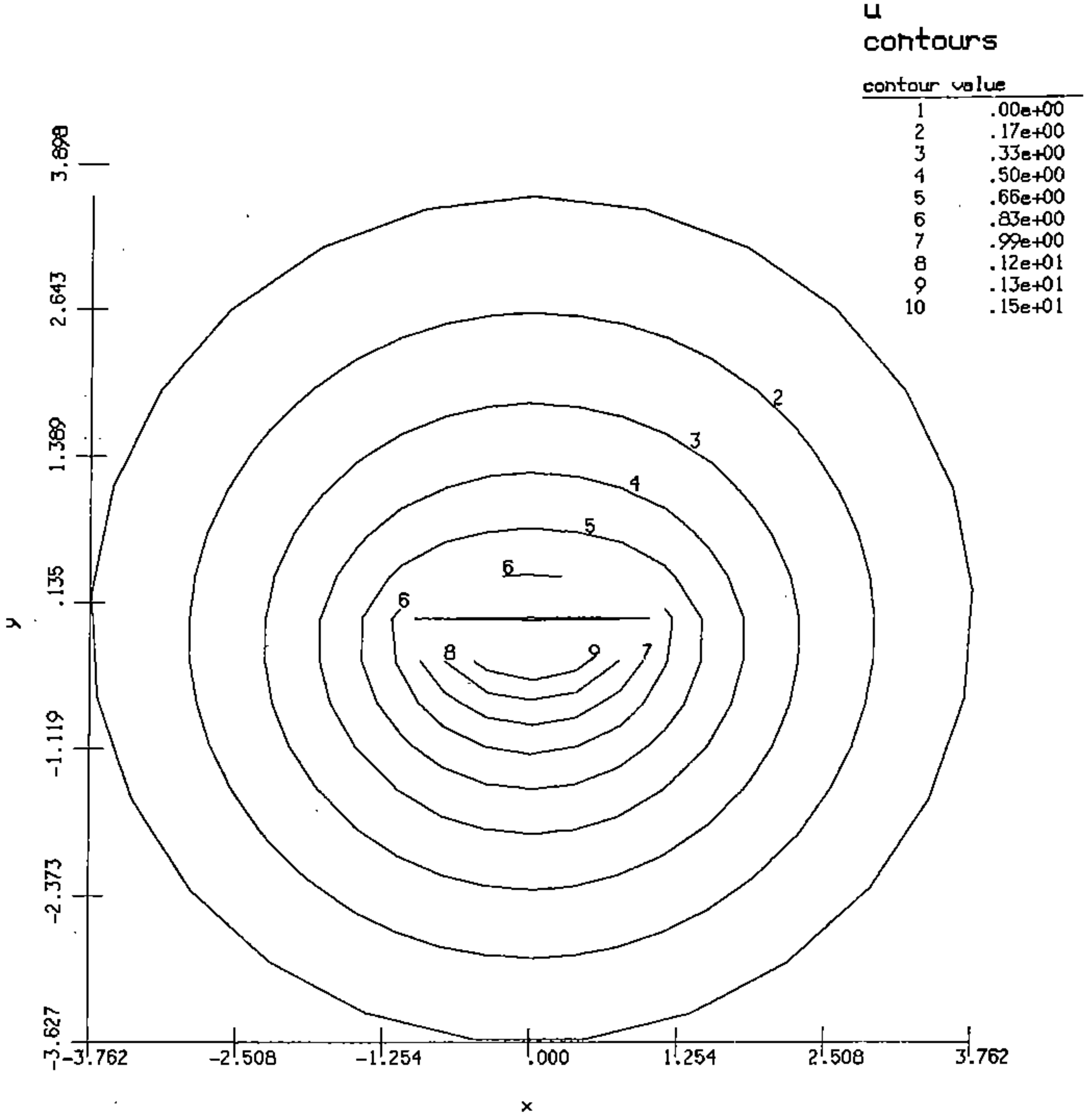

Figure 5.3 Contour plot of solution for program 5.A2

Let the functions u and v represent the temperature of the liquid and solid respectively. We then have the following models for the diffusion of heat in the two phases.

In the solid:

$$\begin{array}{lll} \nabla^2 v & = 0.0 & \text{for } 0<x<1,\ -1/2<y<0 \\ dv/dx & = 0.0 & \text{for } x=0,1,\ -1/2<y<0 \\ v & = 0.0 & \text{for } 0<x<1,\ y=-1/2 \end{array}$$

In the liquid:

$$\begin{array}{lll} \nabla^2 u & = f(x,y) & \text{for } 0<x<1,\ 0<y<1 \\ du/dx & = 0.0 & \text{for } x=0,1,\ 0<y<1 \\ u & = 1.0 & \text{for } 0<x<1,\ y=1 \end{array}$$

The function f is a source term that accounts for the heat introduced as a result of externally induced convection. For this example we $f(x,y)$ = $4y(1-y)\sin((3x+1/2))$.

The diffusion problems are coupled by two continuity conditions along the solid-liquid interface (for $0 < x < 1$, $y=0$):

$$\begin{array}{l} u = v \\ du/dy = k(dv/dy) \end{array}$$

The latter is a jump condition that results from the release of heat during solidification. The constant k is the ratio of thermal conductivity of the solid to the thermal conductivity of the liquid. We take $k = 1/2$.

If we ignore the jump condition then this problem is equivalent to a single phase steady state diffusion problem and is easily solved by the following ELLPACK program. Note that there is no need to distinguish between u and v in this program.

```
EQUATION.       UXX + UYY  =  F(X,Y)

BOUNDARY.       UX  = 0.0  ON X= 0.0
                UX  = 0.0  ON X= 1.0
                U   = 0.0  ON Y=-1/2
                U   = 1.0  ON Y= 1.0

GRID.            9 X POINTS
                13 Y POINTS

DIS.            5-POINT STAR
INDEX.          AS IS
SOL.            BAND GE
OUTPUT.         TABLE(U) $ PLOT(U)

SUBPROGRAMS.
      REAL FUNCTION F(X,Y)
      IF (Y .GE. 0.0) THEN
        F = 4.*Y*(1.-Y)*SIN(1.57080*(3.*X+0.5))
      ELSE
        F = 0.0
      ENDIF
      RETURN
      END
END.
```

This program produces the temperature distribution given in Figure 5.4.

One way to incorporate the jump condition is to modify the output of the 5 POINT STAR module. We wish to change the finite difference equations generated for points along the line $y=0$. We write a subprogram ADJUMP to do this and insert the following code after the existing OUTPUT statement.

```
FORTRAN.
      CALL ADJUMP(R1COEF,I11DCO,I1MNEQ,I1MNCO)
SOL.        BAND GE
OUTPUT.     TABLE(U) $ PLOT(U)
```

This will cause the equations to be modified and the problem solved again with the new discretization. To write the subprogram ADJUMP one must be familiar with the difference equations produced by 5 POINT STAR as well as with the sparse matrix storage scheme used by ELLPACK.

Let (xi, yj), $1 \leq i \leq 12$, denote the uniformly spaced grid point locations and let uij, vij, and fij denote functions evaluated at the point (xi, yj). The difference equation written by 5-POINT STAR for the point (xi, yj) in the interior of the domain is

Figure 5.4: Solution of diffusion problem without jump condition.

In the liquid:

$$u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1} + -4u_{i,j} = h^2 f_{ij} \qquad (1)$$

In the solid:

$$v_{i+1,j} + v_{i,j+1} + v_{i-1,j} + v_{i,j-1} - 4v_{ij} = 0 \qquad (2)$$

where $h = 1/8$ in this example. (The 5-POINT STAR module actually divides these equations by h^2.) Along the left and right sides of the domain these equations must also incorporate the boundary conditions $du/dx = 0$. For the liquid phase these equations become

$$u_{1,j+1} + 2u_{2,j} + u_{1,j-1} - 4u_{1,j} = h^2 f_{1,j} \qquad (3)$$

At the point (x_9, y_j):

$$v_{9,j+1} + 2v_{8,j} + v_{9,j-1} - 4v_{9,j} = h^2 f_{9,j} \qquad (4)$$

with similar equations for the solid phase.

Along the line $y=0$ we also wish these finite difference equations to satisfy the jump condition $du/dy = k(dv/dy)$. Replacing the derivatives by central differences at the point $(xi, y5)$ we get the discrete analogue

$$u_{i,6} - u_{i,4} = k(v_{i,6} - v_{i,4}) \qquad (5)$$

Note that we have introduced two fictitious quantities, $u5,j-1$ and $v5,j+1$, representing a liquid temperature in the solid and a solid temperature in the liquid respectively. We eliminate these using the relations (1) and (2) and use the continuity condition $ui,5 = vi,5$ to get

$$(1+k)u_{i+1,5} + 2u_{i,6} + (1+k)u_{i-1,5} + 2kv_{i,4} - 4(1+k)u_{i,5}$$
$$= h^2 f_{i,5}$$

for $i=1$ and $i=9$ we must use the boundary finite difference equations (3) and (4) and their solid analogues to eliminate the fictitious points from (5). Doing this, and using continuity, we obtain

At the point (x_1,y_5):

$$2u_{1,6} + 2(1+k)u_{2,5} + 2kv_{1,4} - 4(1+k)u_{1,5} = h^2 f_{1,5}$$

At the point (x_9,y_5)

$$2u_{9,6} + 2(1+k)u_{8,5} + 2kv_{9,4} - 4(1+k)u_{9,5} = h^2 f_{9,5}$$

Note again that there is no need to distinguish between u and v in the ELLPACK program since exactly one value is defined at each grid point.

In ELLPACK each equation and unknown is given a single index number from one to the number of equations and unknowns. Thus we must also know how 5-POINT STAR maps the double subscripts used above into the single subscripts used in ELLPACK (equivalently, how grid points are numbered). 5-POINT STAR uses the so-called natural ordering, so the (i,j)th point is given the index $9(j-2)+i$.

The coefficients of the kth finite difference equation are loaded into the kth row of the array R1COEF. The indices of the unknowns that these coefficients multiply are loaded into the corresponding locations of the array I1IDCO. (See Chapter 14 for details.) Note that as a result of the way in which we defined the function F we need not modify the right-hand sides of these equations. The following subprogram performs all these operations.

```
      SUBROUTINE ADJUMP (COEF,IDCO,MNEQ,MNCO)
C
C  CHANGE EQUATIONS ALONG Y=0 TO ACCOUNT FOR JUMP CONDITION
      REAL COEF(MNEQ,MNCO)
      INTEGER IDCO(MNEQ,MNCO)
C
```

```
      NX    = 9
      H     = 1.0/FLOAT(NX-1)
      H2    = H*H
      RK    = 0.5
      RK1   = 1.0 + RK
      INTER = 5
C
   COMPUTE NEW DIFFERENCE EQUATION COEFFICIENTS
C
      CO = -4.0* 1/H2
      CE = RK1/H2
      CW = RK1/H2
      CN = 2.0/H2
      CS = 2.0*RK/H2
C
C  LOAD COEFFICIENTS FOR MODIFIED INTERIOR POINTS
C
      ISTART = NX*(INTER-2) + 1
      ISTOP  = ISTART + NX-1
      DO 100 ISTART,ISTOP
         IN = I + NX
         IS = I - NX
         IE = I + 1
         IW = I - 1
         DO 50 K=1,MNCO
            INDEX = IDCO(I,K)
            IF (INDEX .EQ. I) THEN
               COEF(I,K) = CO
            ELSE IF (INDEX .EQ. IN) THEN
               COEF(I,K) = CN
            ELSE IF (INDEX .EQ. IS) THEN
               COEF(I,K) = CS
            ELSE IF (INDEX .EQ. IE) THEN
               COEF(I,K) = CE
            ELSE IF (INDEX .EQ. IW) THEN
               COEF(I,K) = CW
            ENDIF
   50    CONTINUE
  100 CONTINUE
C
C  LOAD COEFFICIENTS FOR MODIFIED BOUNDARY POINTS
C
      I  = ISTART
      IE = I + 1
      DO 110 K=1,MNCO
        IF (IDCO(I,K) .EQ. IE) COEF(I,K) = 2.0*COEF(I,K)

  110 CONTINUE
      I  = ISTOP
      IW = I - 1
      DO 120 K=1,MNCO
        IF (IDCO(I,K) .EQ. IW) COEF(I,K) = 2.0*COEF(I,K)
  120 CONTINUE
C
C   PRINT MODIFIED EQUATIONS
C
       DO 150 I=ISTART,ISTOP
       WRITE(I1OUTP) I
       DO 150 K=D,MNCO
         IF (IDCO(I,K) .NE. 0) WRITE(6,3001) K,IDCO(I,K),COEF(I,K)
   150 CONTINUE
       RETURN
C
 3000 FORMAT(' EQUATION ',I3 ' ************************')
 3001 FORMAT('    K=',I2,'  ID=',I3,' COEF=',1PE15.8)
      END
```

The result of the solution of the modified finite difference equations is shown in

Figure 5.5.

5.C NEWTON ITERATION FOR NONLINEAR PROBLEMS

Program 4.C1 in the previous chapter illustrates one way to solve nonlinear problems using ELLPACK. Fixed point iteration (also known as Picard's method) has a rate of convergence that is rarely fast. Newton's method usually converges usually very rapidly once one gets reasonably close to the solution and is very efficient when it works. If one visualizes the nonlinear elliptic PDE as just an equation for u (admittedly more complicated than usual) then we want to solve

$$F(u) = 0$$

Newton's method is to expand u in a Taylor's series at a point, say u_0, then discard all but the linear terms in $\delta = u - u_0$ and solve for δ. Symbollically, the Newton change δ satisfies

$$F(u_0) + F(u_0)\delta = 0$$

In the case of systems of nonlinear equations, δ is a vector and $F(u_0)$ is the Jacobian matrix (with entries $\partial F_i / \partial u_j$). For a partial differential equation the "derivative" $F(u_0)\,\delta$ the Frechet derivative at u_0. The Newton estimate u obtained at u_0 satisfies

$$F(u_0) + L(u_0, u) = 0$$

where L is a linear partial differential equation. Thus Newton's method for the nonlinear partial differential equation $F(u) = 0$ is as follows:

```
COMPUTE THE FRECHET DERIVATIVE $L(U, V)$
```

FIGURE 5.5: SOLUTION OF DIFFUSION PROBLEM WITH JUMP CONDITION

```
GUESS U = $U_0$
FOR K = 0 TO LIMIT DO
     SOLVE $F(U_K) + L(U_K, U_{K+1}) = 0$
     EXIT IF CONVERGENCE TEST IS PASSED
END-LOOP
PRINT RESULTS
```

Newton's method can be implemented directly in ELLPACK. The computation of L is straightforward (it is the linearized perturbation of $u+\delta$ in $F(u)$); a MACSYMA program is given below for this which is very helpful when the algebra becomes tedious. The technique is illustrated first for the simple example

$$F(u) = u_{xx} + u^2 u_{yy} - e^u - f = 0 \qquad 0 \le x, y \le 1$$

where $f(x,y)$ and the boundary conditions are chosen to make the true solution be $u(x,y) = sin(x)\, cos(y)$.

If we make a perturbation δ of u in this example and discard all powers of δ beyond the first we obtain

$$\begin{aligned}
&(u+\delta)_{xx} + (u+\delta)^2(u+\delta)_{yy} - e^{u+\delta} - f \\
&= (u_{xx} + \delta_{xx}) + (u^2 u_{yy} + 2u\delta u_{yy} + u^2\delta_{yy}) - (e^u + \delta e^u) - f \\
&= (u_{xx} + u^2 u_{yy} - e^u - f) + \delta_{xx} + u^2\delta_{yy} + (2u - e^u)\delta \\
&= F(u) + L(u, u+\delta)
\end{aligned}$$

We change the notation to correspond to the iteration by setting $u = u_0$ and then let u in the ELLPACK notation denote the new iterate (u_{k+1}). Thus $\delta = u - u_0$ in the new notation and the above equation becomes

$$\begin{aligned}
&u_{xx} + (u_0)^2 u_{yy} + (2\, u_0 u_{0yy} - e^{u_0})u \\
&\qquad = 2(u_0)^2 u_{0yy} + e^{u_0}(1+u_0) + f \qquad 0 \le x, y \le 1
\end{aligned}$$

with boundary conditions $u(x,y) = sin(x)\, cos(y)$. Actually, in the ELLPACK program, both $u_0(x,y)$ and $u(x,y)$ are denoted by u. The u's in the coefficients of L are evaluated before the problem is solved and thus are the previous estimate

u_0; the estimate produced by solving the linearized problem is also u - and it becomes the u_0 for the next iteration.

An ELLPACK program for this example follows which has the initial guess $u(x,y) = 0$; solves the linearized problem by collocation with Hermite bi-cubics and limits the method to 5 iterations. Various other features of the program are explained in the comments. The only output we give is the table produced by the subroutine SUMMARY; it shows the convergence is quite fast.

```
*       *************************************************************
*       *                                                           *
*       *  EXAMPLE ELLPACK PROGRAM 5.C1                             *
*       *                                                           *
*       *  REMARKS                                                  *
*       *      APPLY NEWTON'S METHOD TO THE NONLINEAR PROBLEM       *
*       *                                                           *
*       *      UXX + U*UYY = EXP(U) + F(X,Y)                        *
*       *                                                           *
*       *************************************************************
*

DECLARATIONS.
        REAL ERRMAX(100)

*       USE THE PDE FOR THE NEW U(X,Y) OBTAINED BY LINEARIZING THE
*       NONLINEAR PROBLEM

EQUATION.
        UXX + (U(X,Y)**2)*UYY + (2.*U(X,Y)*UYY(X,Y)-EXP(U(X,Y)))*U =

.       2*(U(X,Y)**2)*UYY(X,Y) - EXP(U(X,Y))*(U(X,Y)-1.) + F(X,Y)

BOUNDARY.
         U = TRUE(X,Y)  ON X = 0.
                        ON X = 1.
                        ON Y = 0.
                        ON Y = 1.

GRID.   5 X POINTS $ 5 Y POINTS

*       INITIALIZE THE NEWTON ITERATION BY GUESSU = 0

TRIPLE. INITIALIZE U ( U = GUESSU )

*       USE FORTRAN TO CONTROL ITERATION AND OUTPUT

FORTRAN.
        I1LEVL = 1
        NITERS = 5
        DO 10 NITER = 1, NITERS

*          SOLVE THE LINEARIZED PROBLEM

DISCRETIZATION. HERMITE COLLOCATION
INDEXING        AS IS
SOLUTION        BAND GE

FORTRAN.

*          COMPUTE INTERMEDIATE MAX ERROR, SAVE FOR TABLED OUTPUT
*          TURN OFF ELLPACK OUTPUT
```

```
              CALL MAXERR(ERRMAX(NITER))
              I1LEVL = 0
   10      CONTINUE

*          PROCESS FINAL RESULTS

           CALL SUMARY(ERRMAX,NITERS)

SUBPROGRAMS.
      FUNCTION F(X,Y)
C
C     F IS CHOSEN TO MAKE THE TRUE SOLUTION SIN(X)*COS(Y)
C
      TRUE   = SIN(X)*COS(Y)
      TRUEXX = -TRUE
      TRUEYY = -TRUE
      F = TRUEXX + TRUE**2*TRUEYY + (2.*TRUE*TRUEYY-EXP(TRUE))*TRUE
     $  - 2.*TRUE**2*TRUEYY + EXP(TRUE)*(TRUE-1.)
      RETURN
      END
      FUNCTION GUESSU(X,Y)
      GUESSU = 0.
      RETURN
      END
      FUNCTION TRUE(X,Y)
      TRUE = SIN(X)*COS(Y)
      RETURN
      END
      SUBROUTINE MAXERR (ERRMAX)
C
C     COMPUTE THE MAXIMUM ERROR ON THE GRID, SAVE FOR LATER
C     ACCESS INTERNAL ELLPACK VARIABLES
C
      COMMON  / C1IVGR /  I1NGRX, I1NGRY, I1NGRZ, I1NBPT, I1MBPT
      COMMON  / C1GRDX /  R1GRDX(1)
      COMMON  / C1GRDY /  R1GRDY(1)
      ERRMAX = 0.
      DO 20 I = 1, I1NGRX
         X = R1GRDX(I)
         DO 10 J = 1, I1NGRY
            Y = R1GRDY(J)
            ERRMAX = AMAX1(ERRMAX,ABS(TRUE(X,Y)-U(X,Y)))
   10    CONTINUE
   20 CONTINUE
C
      RETURN
      END
      SUBROUTINE SUMARY (ERRMAX, NITERS)
C
C     PRINT SUMMARY OF RESULTS
C
      REAL ERRMAX(1)
      PRINT 100
      DO 10 NITER = 1, NITERS
         PRINT 110, NITER, ERRMAX(NITER)
   10 CONTINUE
      RETURN
  100 FORMAT('1       EXAMPLE ELLPACK PROGRAM 5.C1'//
     A        T8,'ITER',T16,'MAX ERROR',/T7,6('-'),2X,10('-'))
  110 FORMAT(T8,I4,1X,1P1E12.4)
      END

END.
```

The table produced by program 5.C1 is

PROGRAM 5.C1 ERRORS IN NEWTON ITERATES

ITER	MAX ERROR
1	.096433
2	.031135
3	.004753
4	.00010151
5	.000000298

Example program, 5.C1 illustrates the use of Newton's method in ELLPACK, it and the two following examples may be viewed as ELLPACK "template" as they show the general structure of such progmams.

Since linearizing the nonlinear operator can be tedious (and error prone) we give a MACSYMA program that produces the linear operator $L(u_0,u)$ automatically for this problem. This program can be adapted for nonlinear problems of all types; the linear operator $B(u_0,u)$ can also be obtained by a similar program.

To illustrate this technique and to show that ELLPACK can solve difficult real world problems, we provide two more example problems. See also G. Birkhoff and R. Lynch, *Numerical Methods for Elliptic Partial Differential Equations* SIAM, 1983 for the solution of Plateau's problem using this approach. The second example is from nonlinear, laminar, non-Newtonion flow [Ref: W.F. Ames, *Nonlinear Partial Differential Equations in Engineering* , Academic Press, 1965]. The nonlinear elliptic problem is

$$w(u)(u_{xx} + u_{yy}) + w_x(u)u_x + w_y(u)u_y = f(x,y)$$

$$u_x = 0 \quad on \quad x = 0,1$$

$$u = b(x) \quad on \quad y = 0,1$$

where the function $w(u)$ varies depending on the application. Set $a(u) = \sqrt{u_x^2 + u_y^2}$ then physically meaningful cases of $w(u)$ are

$$w(u) = [a(u)]^{\alpha} \qquad w(u) = 1/(\alpha+\beta a(u))$$
$$w(u) = e^{[a(u)/(\alpha+\beta a(u))]}/a(u) \quad w(u) = \alpha tanh(\beta a(u))/a(u)$$

This nonlinear problem is the source of problems 19 and 23 in the PDE population given in Appendix 3. We take one of the simplest possible cases here, $w(u) = a(u)$ (i.e. α=1). We choose $f(x,y)$ and $b(x)$ so that the true solution of the problem is

$$u(x,y) = (1+e^{-y})\cos(\pi x)$$

```
(C1) /* EXAMPLE PROGRAM 5.C2                    */
     /* LIST THE NONLINEAR PDE COEFFICIENTS */

     A(U) := W(U)$

(C2) B(U) := 0$

(C3) C(U) := W(U)$

(C4) D(U) := DIFF(W(U),X)$

(C5) E(U) := DIFF(W(U),Y)$

(C6) F(U) := 0$

(C7) G(U) := 0$

(C8) W(U) := SQRT ( DIFF(U,X)**2 + DIFF(U,Y)**2 )$

(C9) /* DEFINE DERIVATIVES OF U0, U0X, ETC. */

      GRADEF(U0,X,U0X)$

(C10) GRADEF(U0X,X,U0XX)$

(C11) GRADEF(U0X,Y,U0XY)$

(C12) GRADEF(U0Y,X,U0XY)$

(C13) GRADEF(U0,Y,U0Y)$

(C14) GRADEF(U0Y,Y,U0YY)$

(C15) GRADEF(U1,X,U1X)$

(C16) GRADEF(U1X,X,U1XX)$

(C17) GRADEF(U1X,Y,U1XY)$

(C18) GRADEF(U1Y,X,U1XY)$

(C19) GRADEF(U1,Y,U1Y)$

(C20) GRADEF(U1Y,Y,U1YY)$

(C21) /* WRITE THE NONLINEAR PDE */

      PDE(U) := A(U)*DIFF(U,X,2) + B(U)*DIFF(DIFF(U,X),Y) + C(U)*DIFF(U,Y,2)
                + D(U)*DIFF(U,X) + E(U)*DIFF(U,Y) + F(U)*U + G(U)$
```

```
(C22) /* DIFFERENTIATE AND COLLECT TERMS */

      DERIVATIVE:DIFF(PDE(U0+EPS*(U1-U0)),EPS)$

(C23) TSERIES:PDE(U0) + EV(DERIVATIVE,EPS=0)$

(C24) RATSIMP(TSERIES)$

(C25) TSERIES:EXPAND(TSERIES)$

(C26) COEU1XX:COEFF(TSERIES,U1XX)$

(C27) COEU1XY:COEFF(TSERIES,U1XY)$

(C28) COEU1YY:COEFF(TSERIES,U1YY)$

(C29) COEU1X:COEFF(TSERIES,U1X)$

(C30) COEU1Y:COEFF(TSERIES,U1Y)$

(C31) COEU1:COEFF(TSERIES,U1)$

(C32) RS:(COEU1XX*U1XX + COEU1XY*U1XY + COEU1YY*U1YY + COEU1X*U1X +
      COEU1Y*U1Y + COEU1*U1)-TSERIES$

(C33) RATSIMP(RS)$

(C34) /* DISPLAY THE COEFFICIENTS OF THE LINEARIZED PDE FOR NEWTONS METHOD */

      COEU1XX:RATSIMP(COEU1XX * W(U0));
                                    2        2
(D34)                            U0Y  + 2 U0X

(C35) COEU1XY:RATSIMP(COEU1XY * W(U0));
(D35)                              2 U0X U0Y

(C36) COEU1YY:RATSIMP(COEU1YY * W(U0));
                                      2      2
(D36)                            2 U0Y  + U0X

(C37) COEU1X :RATSIMP(COEU1X  * W(U0));
                3                      3                   2          3
             U0X  U0YY + 2 U0XY U0Y  + 3 U0X U0XX U0Y  + 2 U0X  U0XX
(D37)        ----------------------------------------------------------
                                      2      2
                                   U0Y  + U0X

(C38) COEU1Y :RATSIMP(COEU1Y  * W(U0));
                   3         2                  3          3
             (2 U0Y  + 3 U0X  U0Y) U0YY + U0XX U0Y  + 2 U0X  U0XY
(D38)        ------------------------------------------------------
                                    2      2
                                 U0Y  + U0X

(C39) COEU1   :RATSIMP(COEU1   * W(U0));
(D39)                                   0

(C40) RS      :RATSIMP(RS      * W(U0));
               2      2                  2                           2
(D40)   (2 U0Y  + U0X ) U0YY + U0XX U0Y  + 2 U0X U0XY U0Y + 2 U0X  U0XX
```

The algebra to derive the linearized problem is formidable even in this simplest possible case. The result from using the MACSYMA program is used in program 5.C4.

```
*       *                                                          *
*       *  EXAMPLE ELLPACK PROGRAM 5.C3                            *
*       *                                                          *
*       *  REMARKS                                                 *
*       *      APPLY NEWTON'S METHOD TO THE NONLINEAR PROBLEM      *
*       *                                                          *
*       *      W(U)(UXX + UYY) + WX(U)UX + WY(U)UY = F             *
*       *                                                          *
*       ************************************************************
*

DECLARATIONS.
        REAL ERRMAX(100)

*       USE THE PDE FOR THE NEW U(X,Y) OBTAINED BY LINEARIZING THE
*       NONLINEAR PROBLEM

EQUATION.
         ....

BOUNDARY.
        UX = 0.                          ON X = 0.
                                         ON X = 1.
        U =              2.*COS(PI*X)    ON Y = 0.
        U = (1.+EXP(-1.))*COS(PI*X)      ON Y = 1.

GRID.   5 X POINTS $ 5 Y POINTS

*       INITIALIZE THE NEWTON ITERATION BY INTERPOLATING THE
*       BOUNDARY CONDITIONS BY BLENDING FUNCTIONS

*RIPLE. INTERPOLATE BOUNDARY CONDITIONS BY BLENDING
TRIPLE. INITIALIZE U ( U = GUESSU )

*       USE FORTRAN TO CONTROL ITERATION AND OUTPUT

FORTRAN.
        I1LEVL = 1
        NITERS = 5
        DO 10 NITER = 1, NITERS

*          SOLVE THE LINEARIZED PROBLEM

DISCRETIZATION. HERMITE COLLOCATION
INDEXING        ASIS
SOLUTION        BAND GE

FORTRAN.

*          COMPUTE INTERMEDIATE MAX ERROR, SAVE FOR TABLED OUTPUT

           CALL MAXERR(ERRMAX(NITER))
           I1LEVL = 0
   10   CONTINUE

*       PROCESS FINAL RESULTS

        CALL SUMARY(ERRMAX,NITERS)

SUBPROGRAMS.
      FUNCTION W(X,Y)
      W = SQRT(UX(X,Y)**2 + UY(X,Y)**2)
      RETURN
      END
      FUNCTION WX(X,Y)
      WX = 2.*UX(X,Y)*UXX(X,Y)/SQRT(UX(X,Y)**2 + UY(X,Y)**2)
      RETURN
      END
      FUNCTION WY(X,Y)
      WX = 2.*UY(X,Y)*UYY(X,Y)/SQRT(UX(X,Y)**2 + UY(X,Y)**2)
      RETURN
      END
      FUNCTION F(X,Y)
C
C     F IS CHOSEN TO MAKE THE TRUE SOLUTION (1.+EXP(-Y))*COS(PI*X)
C
      COMMON  / C1RVGL /  R1EPSG, R1EPSM, PI
      F = ...
      RETURN
      END
      FUNCTION GUESSU(X,Y)
      GUESSU = 0.
      RETURN
      END
      FUNCTION TRUE(X,Y)
C     ACCESS PI = 3.14159... FROM ELLPACK COMMON
      COMMON  / C1RVGL /  R1EPSG, R1EPSM, PI
      TRUE = (1.+EXP(-Y))*COS(PI*X)
      RETURN
      END
      SUBROUTINE MAXERR (ERRMAX)
C
C     COMPUTE THE MAXIMUM ERROR ON THE GRID, SAVE FOR LATER
C     ACCESS INTERNAL ELLPACK VARIABLES
C
      COMMON  / C1IVGR /  I1NGRX, I1NGRY, I1NGRZ, I1NBPT, I1MBPT
      COMMON  / C1GRDX /  R1GRDX(1)
      COMMON  / C1GRDY /  R1GRDY(1)
      ERRMAX = 0.
      DO 20 I = 1, I1NGRX
         X = R1GRDX(I)
         DO 10 J = 1, I1NGRY
            Y = R1GRDY(J)
            ERRMAX = AMAX1(ERRMAX,ABS(TRUE(X,Y)-U(X,Y)))
   10    CONTINUE
   20 CONTINUE
C
      RETURN
      END
      SUBROUTINE SUMARY (ERRMAX, NITERS)
C
C     PRINT SUMMARY OF RESULTS
C
      REAL ERRMAX(1)
      PRINT 100
      DO 10 NITER = 1, NITERS
         PRINT 110, NITER, ERRMAX(NITER)
   10 CONTINUE
      RETURN
  100 FORMAT('1      EXAMPLE ELLPACK PROGRAM 5.C2'//
     A       T8,'ITER',T16,'MAX ERROR',/T7,6('-'),2X,10('-'))
  110 FORMAT(T8,I4,1X,1P1E12.4)
      END

END.
```

The next real application comes from gas lubrication, this is the effect that keeps high speed tapes and disks from making physical contact with read/write heads. Two views of the physical situations are shown in Figure 5.6. The separation between the disk and head is only a few thousandths of an inch. The high speed of the disk pulls the air into the gap; it is compressed as it goes through and this builds up a pressure to keep the two parts separated.

Figure 5.6. Top view (left) of a magnetic read had and side view (right) of the space between the head and disk.

The nonlinear elliptic problem to be solved on the domain shown in Figure 5.6a is

$$\begin{gathered}(uh^3 u_x)_x + (uh^3 u_y) + c\,(uh)_x = 0 \\ u(x,y) = 1 \;\; \textit{on the boundary}\end{gathered}$$

The function $h(x,y)$ is

$$\begin{aligned} h(x,y) &= 1 \qquad 0 \le x \le .5 \\ &= 1+2(x-.5) \;\; .5 \le x \le 1.5 \end{aligned}$$

and c is a physical constant. The expanded form of the elliptic operator is

$$u * u_{xx} + u * u_{yy} + (u_x + \frac{3h}{h} + \frac{c}{h^2})\, u_x + (u_y + \frac{3h}{u})\, u_y + \frac{ch_x}{h^3} u = 0$$

The linearized equation to be solved is

$$\begin{aligned} & u0 * u_{xx} + u0_{yy} + (2u0_x + 3h/u0 + c/h^2)u_x \\ & + (2u0_y + 3h/u0)u_y + (u0_{xx} + u0_{yy} + \frac{3}{h}(u0_x + u0_y) + \frac{ch_x}{h^3})u \\ & = u0(u0_{xx} + u0_{yy}) + u0_x^2 + u0_y^2 + 3h(u0_x + u0_y)/u0 \end{aligned}$$

Program 5.C4 uses this linearized equation to solve this problem with Newton's method. The principal result needed from this problem is the integral of $u(x,y)$ over the domain which is the load that the lubricant supports. The IMSL library routine DBLINT is used but not given in the subprograms.

```
*        ****************************************************
*        *                                                  *
*        *  EXAMPLE ELLPACK PROGRAM 5.C4                    *
*        *                                                  *
*        *  REMARKS                                         *
*        *      APPLY NEWTON'S METHOD TO THE NONLINEAR PROBLEM  *
*        *                                                  *
*        *          3          3                            *
*        *      (UH U )  +  (UH U )  + C(UH)  = 0           *
*        *           X X         Y Y        X               *
*        *                                                  *
*        *       THIS IS A FORM OF REYNOLD'S EQUATION FOR   *
*        *       COMPRESSIBLE FLUID LUBRICATION.            *
*       *                                                    *
*       ******************************************************
*

OPTIONS. OLDU = 1

DECLARATIONS.
        REAL DIFMAX(100)

*       USE THE PDE FOR THE NEW U(X,Y) OBTAINED BY LINEARIZING THE
*       NONLINEAR PROBLEM

EQUATION.
        ...

BOUNDARY.
         U = 1. ON ....


GRID.   5 X POINTS $ 5 Y POINTS

*       INITIALIZE THE NEWTON ITERATION BY GUESSU = 0

TRIPLE. INITIALIZE U ( U = GUESSU )

*       USE FORTRAN TO CONTROL ITERATION AND OUTPUT

FORTRAN.
        I1LEVL = 1
        NITERS = 5
        DO 10 NITER = 1, NITERS

*          SOLVE THE LINEARIZED PROBLEM

DISCRETIZATION. HERMITE COLLOCATION
INDEXING        ASIS
SOLUTION        BAND GE

FORTRAN.

*          COMPUTE INTERMEDIATE MAX DIFF, SAVE FOR TABLED OUTPUT

           CALL MAXDIF(DIFMAX(NITER))
           I1LEVL = 0
  10    CONTINUE

*       PROCESS FINAL RESULTS

        CALL SUMARY(DIFMAX,NITERS)

SUBPROGRAMS.
      FUNCTION GUESSU(X,Y)
      GUESSU = 0.
      RETURN
      END
      END
      SUBROUTINE MAXDIF (DIFMAX)
C
C     COMPUTE THE MAXIMUM U DIFFERENCES ON THE GRID, SAVE FOR LATER
C     ACCESS INTERNAL ELLPACK VARIABLES
C
      COMMON  / C1IVGR /  I1NGRX, I1NGRY, I1NGRZ, I1NBPT, I1MBPT
      COMMON  / C1GRDX /  R1GRDX(1)
      COMMON  / C1GRDY /  R1GRDY(1)
      DIFMAX = 0.
      DO 20 I = 1, I1NGRX
         X = R1GRDX(I)
         DO 10 J = 1, I1NGRY
            Y = R1GRDY(J)
            DIFMAX = AMAX1(DIFMAX,ABS(U1(X,Y)-U(X,Y)))
  10     CONTINUE
```

```
   20 CONTINUE
C
      RETURN
      END
      SUBROUTINE SUMARY (DIFMAX, NITERS)
C
C     PRINT SUMMARY OF RESULTS
C
      REAL DIFMAX(1)
      PRINT 100
      DO 10 NITER = 1, NITERS
         PRINT 110, NITER, DIFMAX(NITER)
   10 CONTINUE
      RETURN
  100 FORMAT('1       EXAMPLE ELLPACK PROGRAM 5.C4'//
     A       T8,'ITER',T16,'MAX DIFF',/T7,6('-'),2X,10('-'))
  110 FORMAT(T8,I4,1X,1P1E12.4)
      END

END.
```

5.D TIME DEPENDENT PROBLEM

ELLPACK can be used fairly directly for the following time dependent problem:

$$u_t = Lu + f \qquad\qquad u = u0(x,y) \text{ for } t=0$$
$$u = ubound(x,y,t) \text{ for } (x,y) \textit{ on boundary}$$

where L is a linear elliptic operator; an example of L is

$$Lu = 4y^2 u_{xx} + u_{yy} + (2 + tan(x+y+t))u_y + u$$

Note that the coefficients in L could depend on x,y and t as well as the forcing function f. ELLPACK does not automatically discretize the u_t term, so this must be done in the program explicitly. The simplest discretization is

$$u_t \sim \frac{u(t) - u(t-\Delta t)}{\Delta t}$$

which leads to the discrete equation

$$u(t) = u(t-\Delta t) + \Delta t * (Lu(x,y,t-\Delta t) + f(x,y,t-\Delta t))$$

ELLPACK can be used to discretize the $Lu(x,y,t)$ term, but this is not an attractive use of the ELLPACK facilities. It is better in most cases to use the more accurate Crank-Nicolson time discretization.

The Crank-Nicolson discretization uses the same approximation to u_t, but it is viewed as estimating u_t at $t-\Delta t/2$ instead of at $t-\Delta t$. The partial differential equation is then discretized to be

$$\begin{aligned} u(t) = u(t-\Delta t) + .5\Delta t\,[&Lu(x,y,t) + Lu(x,y,t-\Delta t) \\ &+ f(x,y,t) + f(x,y,t-\Delta t)] \end{aligned}$$

This discretization in time is always stable so that large time steps Δt can be taken. For each time step one must solve the elliptic problem

$$\begin{aligned} &Lu(x,y,t) - (2/\Delta t)u(x,y,t) \\ &\quad = -(2/\Delta t)u(x,y,t-\Delta t) - Lu(x,y,t-\Delta t) + f(x,y,t) + f(x,y,t-\Delta t) \end{aligned}$$

The terms on the right are known and on the left we have a linear elliptic equation which ELLPACK can solve.

Note that any ELLPACK method can be used to solve this problem, but there should be an interaction between the method chosen and the choice of Δt. To discretize space we choose an x,y grid and, for simplicity, we assume that x and y spacings are the same, h. We are essentially applying the methods of lines with one line (in time) for each grid node. However, we do not need to examine these lines individually or label the corresponding line solutions. The time discretization error from Crank-Nicolson is order $(\Delta t)^2$ and this should be similar to the space discretization error. If 5-POINT STAR is used with discretization error order h^2 then one should have h and Δt of about the same size. At least, if they are decreased, they should be decreased proportionally. If HERMITE COLLOCATION or SPLINE GALERKIN (DEGREE=3,SMOOTH=2) is used, then their discretization errors are order h^4 and one should have h^2 and Δt about the same size. With these discretizations one can take many fewer time steps for a given accuracy.

We give the ELLPACK program to solve this example. The functions $u0(x,y)$, $ubound(x,y,t)$ and $f(x,y,t)$ are chosen so that the true solution is

$$u(x,y,t) = sin(x+y+t)/4\, e^{-y^2-t}$$
$$1 \le t \le 2 \quad 0 \le x,y \le 1$$

The elliptic problem is solved with INTERIOR COLLOCATION which uses bi-cubic Hermite polynomials and has error of order h^4. We set $\Delta t = h^2/2$ and put the elliptic problem in a simple loop for the time steps.

```
*       *****************************************************
*       *                                                   *
*       *  EXAMPLE ELLPACK PROGRAM 5.D1                     *
*       *                                                   *
*       *  REMARKS                                          *
*       *      TIME DEPENDENT PROBLEM                       *
*       *      SEE THE ELLPACK PROGRAM TEMPLATE FOR GENERAL *
*       *      COMMENTS. COMMENTS ARE GIVEN FOR STATEMENTS  *
*       *      SPECIAL TO THIS PROGRAM.                     *
*       *                                                   *
*       *****************************************************
*
*   DECLARE FORTRAN ARRAYS FOR USE IN SUMMARY AT END.
DECLARATIONS.
        REAL TRUMAX(100), ERRMAX(100)

GLOBAL.
        COMMON / GCOMON / T, DELTAT, NSTEP

EQUATION.     (4.*Y**2)UXX + UYY + (2.+TAN((X+Y+T)/4.))UY
.             + (1.-2./DELTAT)U = PDERS(X,Y)

BOUNDARY.     U = UBOUND(X,Y) ON X = 0.
                              ON X = 1.
                              ON Y = 0.
                              ON Y = 1.

GRID.         3 X POINTS $ 3 Y POINTS

OPTIONS.      CONSTANT COEFFICIENTS=.FALSE.

FORTRAN.
        I1LEVL = 1
        TSTART = 1.
        TSTOP  = 2.
C
C   CHOOSE DELTA T = (DELTA X)**2 OVER 2
C
        DELTAT = R1HXGR**2/2
        NSTEPS = INT((TSTOP-TSTART)/DELTAT + .5)
        DELTAT = (TSTOP-TSTART)/NSTEPS

        DO 10 NSTEP = 1, NSTEPS
           T = TSTART + NSTEP*DELTAT
```

```
DISCRETIZATION.     HERMITE COLLOCATION
INDEXING.           ASIS
SOLUTION.           BAND GE

FORTRAN.
C
C    COMPUTE MAX ERROR FOR THIS EXAMPLE, SAVE FOR SUMMARY OUTPUT AT END
C
              CALL TMXEMX(TRUMAX(NSTEP),ERRMAX(NSTEP))
              I1LEVL = 0
   10     CONTINUE
C
C    PRINT SUMMARY OF RESULTS FOR THIS EXAMPLE
C
          CALL SUMARY(TRUMAX,ERRMAX,TSTART,NSTEPS)

SUBPROGRAMS.
      FUNCTION PDERS(X,Y)
      COMMON / GCOMON / T, DELTAT, NSTEP
C
      T = T - DELTAT
      IF (NSTEP .EQ. 1) THEN
            UOFT = U0(X,Y)
         ELSE
            UOFT = U(X,Y)
      ENDIF
C
      PDERS = - (2./DELTAT)*UOFT
     A        - (RLUXYT(X,Y) + F(X,Y,T))
     B        - F(X,Y,T+DELTAT)
C
      T = T + DELTAT
      RETURN
      END

      FUNCTION RLUXYT(X,Y)
C
      REAL COEFOF(6)
      COMMON / GCOMON / T, DELTAT, NSTEP
      INTEGER CUXX, CUXY, CUYY, CUX, CUY, CU
      DATA    CUXX, CUXY, CUYY, CUX, CUY, CU
     A        /  1,    2,    3,   4,   5,  6/
C
      CALL Q1PCOE(X,Y,COEFOF)
C
      IF (NSTEP .EQ. 1) THEN
            RLUXYT = COEFOF(CUXX) * U0XX(X,Y)
     A             + COEFOF(CUYY) * U0YY(X,Y)
     B             + COEFOF(CUY)  * U0Y(X,Y)
     C             + (COEFOF(CU) + 2./DELTAT) * U0(X,Y)
C
         ELSE
            RLUXYT = COEFOF(CUXX) * UXX(X,Y)
     A             + COEFOF(CUYY) * UYY(X,Y)
     B             + COEFOF(CUY)  * UY(X,Y)
     C             + (COEFOF(CU) + 2./DELTAT) * U(X,Y)
      ENDIF
      RETURN
      END

      FUNCTION F(X,Y,T)
      T1 = .25*(X+Y+T)
      T2 = EXP(-Y**2-T)
      F = - (.25*COS(T1) - 2.*Y*SIN(T1)) * T2 * (TAN(T1) + 2.)
     A  + (.0625 - 3.75*Y**2) * T2 * SIN(T1)
     B  + (.25 + Y) * T2 * COS(T1)
      RETURN
      END

      FUNCTION U0XX(X,Y)
      COMMON /GCOMON/ T, DELTAT, NSTEP
      U0XX = -( EXP(-Y**2-T) * SIN((X+Y+T)/4.))/16.
      RETURN
      END

      FUNCTION U0YY(X,Y)
      COMMON /GCOMON/ T, DELTAT, NSTEP
      U0YY = EXP(-Y**2-T) * ((4.*Y**2-2.0625)*SIN((X+Y+T)/4.)
     A     - Y*COS((X+Y+T)/4.) )
      RETURN
      END

      FUNCTION U0Y(X,Y)
      COMMON /GCOMON/ T, DELTAT, NSTEP
      U0Y = EXP(-Y**2-T) / 4. * (COS((X+Y+T)/4.)-8.*Y*SIN((X+Y+T)/4.))
      RETURN
      END

      FUNCTION U0(X,Y)
      COMMON /GCOMON/ T, DELTAT, NSTEP
      U0 = SIN((X+Y+T)/4.) * EXP(-Y**2-T)
      RETURN
      END

      FUNCTION UBOUND(X,Y)
      COMMON /GCOMON/ T, DELTAT, NSTEP
      UBOUND = TRUE(X,Y)
      RETURN
      END

      FUNCTION TRUE(X,Y)
      COMMON /GCOMON/ T, DELTAT, NSTEP
      TRUE = SIN((X+Y+T)/4.) * EXP(-Y**2-T)
      RETURN
      END

      SUBROUTINE TMXEMX (TRUMAX, ERRMAX)
C
C THIS ROUTINE FINDS THE MAX ABSOLUTE VALUE OF TRUE AND ERROR
C ON THE GRID AT THE CURRENT TIME STEP.  USE ELLPACK COMMON BLOCKS
C TO GAIN ACCESS TO VARIABLES DEFINING THE GRID.  ONE COULD EASILY
C COMPUTE THESE ONESELF.
C
      COMMON   / C1IVGR /  I1NGRX, I1NGRY, I1NGRZ, I1NBPT, I1MBPT
      COMMON   / C1GRDX /  R1GRDX(1)
      COMMON   / C1GRDY /  R1GRDY(1)
C
      TRUMAX = 0.
      ERRMAX = 0.
C
      DO 20 I = 1, I1NGRX
         X = R1GRDX(I)
         DO 10 J = 1, I1NGRY
            Y = R1GRDY(J)
            TRUXYT = TRUE(X,Y)
            TRUMAX = AMAX1(TRUMAX,TRUXYT)
            ABSERR = ABS(TRUXYT-U(X,Y))
            ERRMAX = AMAX1(ERRMAX,ABSERR)
   10    CONTINUE
   20 CONTINUE
C
      RETURN
      END

      SUBROUTINE SUMARY (TRUMAX, ERRMAX, TSTART, NSTEPS)
C
C THIS ROUTINE PRINTS A TABLE OF SOLUTION AND RELATIVE ERROR AT EACH TIME
C STEP.  THESE VALUES HAVE BEEN SAVED IN THE ARRAYS TRUMAX AND ERRMAX.
C
      REAL TRUMAX(1), ERRMAX(1)
C
C ACCESS GRID INFORMATION FROM ELLPACK VARIABLES.  THESE ALSO CAN BE
C COMPUTED EASILY WITHOUT REFERENCE TO ELLPACK.
C
```

```
      COMMON / GCOMON / T, DELTAT, NSTEP
      COMMON / C1RVGR /  R1AXGR, R1AYGR, R1AZGR, R1BXGR, R1BYGR,
     A                   R1BZGR, R1HXGR, R1HYGR, R1HZGR
C
C PRINT PROBLEM/METHOD INFORMATION
C
      TSTOP = TSTART + NSTEPS*DELTAT
      PRINT 100, R1HXGR, R1HYGR, TSTART, TSTOP, DELTAT
C
C PRINT HEADING
C
      PRINT 110
      DO 10 NSTEP = 1, NSTEPS
         T = TSTART + NSTEP*DELTAT
         IF (TRUMAX(NSTEP) .NE. 0.) THEN
               PRINT 120, NSTEP, T, TRUMAX(NSTEP),
     A                    ERRMAX(NSTEP)/TRUMAX(NSTEP)
            ELSE
               PRINT 120, NSTEP, T, 0.
         ENDIF
   10 CONTINUE
C
      RETURN
C
  100 FORMAT('1      TIME DEPENDENT PROBLEM'//
     A        T7,'HX       =',1P1E12.4/
     B        T7,'HY       =',1P1E12.4/
     C        T7,'TSTART   =',1P1E12.4/
     D        T7,'TSTOP    =',1P1E12.4/
     E        T7,'DELTA T =',1P1E12.4//)
  110 FORMAT(T8,'STEP',T18,'TIME',T28,'MAX TRUE',T39,'MAX RELERR'/
     A       T7,6('-'),3(2X,10('-')))
  120 FORMAT(T8,I4,1X,1P3E12.4)
C
      END

END.
```

We do not comment on the programming details here because a "template" for solving such problems is given later and the comments there explain most of these points. The bulk of the code is to evaluate $Lu(x,y,t-\Delta t)$ for both the previous time value and the initial conditions (which is a similar, but separate case). A small routine to measure the maximum error is included and the results are listed below for grids of 3×3(Δt = 1/8), 5×5(Δt = 1/32), and 9×9(Δt = 1/128).

Table 5.1. Behavior of the error in solving a time dependent problem with Crank-Nicolson and INTERIOR COLLOCATION.

Maximum Relative	(x,y)-Grid		
Error at t =	3x3	5x5	9x9
1 + 1/8			
1 + 1/4			
1 + 3/8			
1 + 1/2			
1 + 5/8			
1 + 3/4			
1 + 7/8			
2			

We end this section with a general "template" for solving time dependent problems in ELLPACK. The template is heavily commented to explain its use.

```
*
*       ELLPACK TIME DEPENDENT PROBLEM TEMPLATE
*
*       U  = LU + F(X,Y,T)
*        T
*
*       U = U0(X,Y)  FOR T = TSTART,  0 < X,Y < 1
*
*       U = UBOUND(X,Y,T)  FOR TSTART < T < TSTOP,  (X,Y) ON BOUNDARY
*
*       WHERE
*         L IS A LINEAR ELLIPTIC OPERATOR
*         U0 SPECIFIES THE INITIAL VALUES
*         UBOUND SPECIFIES THE BOUNDARY VALUES
*
*
* GLOBAL.  COMMON BLOCK GIVES FUNCTIONS ACCESS TO CURRENT TIME T, TIME
*          SPACING DELTAT, AND CURRENT STEP NUMBER NSTEP.
*
GLOBAL.
        COMMON / GCOMON / T, DELTAT, NSTEP
*
* EQUATION.  DEFINE EQUATION FOR EACH TIME T.  L IS THE LINEAR OPERATOR.
*            DEFINE RIGHT SIDE PDERS(X,Y) BELOW.
*
EQUATION.      LU - (2./DELTAT)U = PDERS(X,Y)
*
* BOUNDARY.  SPECIFY BOUNDARY VALUES.  DEFINE UBOUND(X,Y) BELOW.
*
BOUNDARY.      U = UBOUND(X,Y) ON X = 0.
                               ON X = 1.
                               ON Y = 0.
                               ON Y = 1.
*
* GRID.  CHOOSE GRID LINES FOR PROBLEM.
*
GRID.          5 X POINTS $ 5 Y POINTS
*
* OPTIONS.  FORCE ELLPACK TO EVALUATE COEFFICIENTS OF L FOR EACH TIME T
*           IF SOME COEFFICIENTS DEPEND ON T BUT NOT X OR Y.
*
OPTIONS.       CONSTANT COEFFICIENTS=.FALSE.
*
* FORTRAN.  SET TSTART, TSTOP.  SET DELTAT, DEPENDING ON DISCRETIZATION
*           METHOD.   IN THIS EXAMPLE, DELTAT IS SET TO HX (THE ELLPACK
*            VARIABLE R1HXGR).  COMPUTE NSTEPS (NUMBER OF STEPS), THEN
*            RECOMPUTE DELTAT TO MAKE THE STEPS COME OUT EVEN.
*
FORTRAN.
         I1LEVL = 1
         TSTART = 0.
         TSTOP  = 1.
         DELTAT = R1HXGR
         NSTEPS = INT((TSTOP-TSTART)/DELTAT + .5)
         DELTAT = (TSTOP-TSTART)/NSTEPS
C
C MAIN LOOP OVER TIME.  T IS THE TIME FOR THE CURRENT STEP.
C
         DO 10 NSTEP = 1, NSTEPS
            T = TSTART + NSTEP*DELTAT
C
C CHOOSE MODULES TO BE USED ON PROBLEM AT EACH STEP.  ONE OF MANY
C POSSIBLE COMBINATIONS IS SHOWN.
C
DISCRETIZATION.     5 POINT STAR
INDEXING.           AS IS
SOLUTION.           LINPACK BAND
OUTPUT.             MAX(ERROR)
FORTRAN.
C
C SET OUTPUT LEVEL=0 TO AVOID REPEATED OUTPUT FROM EVERY TRIP THROUGH LOOP.
C
            I1LEVL = 0
   10    CONTINUE
*
* SUBPROGRAMS.  DEFINE PDERS, RLUXYT, INITIAL VALUES, BOUNDARY VALUES,
*               AND TRUE (IF KNOWN).
*
SUBPROGRAMS.
      FUNCTION PDERS(X,Y)
C
C THIS FUNCTION EVALUATES THE PDE'S RIGHT SIDE FOR THE CURRENT TIME T.
C PDERS = (-2/DELTAT)*U(X,Y,T-DELTAT) - LU(X,Y,T-DELTAT)
C        - F(X,H,T-DELTAT) - F(X,Y,T).  NOTE THAT T IS PASSED IN GCOMON.
C
C VARIABLES:
C   X,Y     -  SPACE VARIABLES AT WHICH TO EVALUATE RIGHT SIDE
C   T       -  TIME AT WHICH TO EVALUATE RIGHT SIDE
C   DELTAT  -  TIME SPACING
C   UOFT    -  TEMPORARY VARIABLE; HOLDS U(X,Y) AT LAST TIME T.
C   PDERS   -  RETURNED VALUE OF RIGHT SIDE
C
      COMMON / GCOMON / T, DELTAT, NSTEP
C
C NEED U, LU, AND F AT (X,Y,T-DELTAT).  MOVE TIME T BACK ONE STEP SO ALL
C FUNCTIONS ARE EVALUATED AT THE PREVIOUS TIME STEP.
C
      T = T - DELTAT
C
C FIND U(X,Y,T-DELTAT); IT'S EITHER U0(X,Y) FOR THE INITIAL STEP,
C OR U(X,Y) WHERE U IS THE ELLPACK FUNCTION WHICH GIVES THE RESULT
C AT THE PREVIOUS TIME STEP.
C
      IF (NSTEP .EQ. 1) THEN
           UOFT = U0(X,Y)
         ELSE
           UOFT = U(X,Y)
      ENDIF
C
C EVALUATE RIGHT SIDE USING RLUXYT FOR LU AT PREVIOUS TIME STEP
C
      PDERS = - (2./DELTAT)*UOFT
     A        - (RLUXYT(X,Y) + F(X,Y,T))
     B        - F(X,Y,T+DELTAT)
C
C RESTORE T TO CURRENT VALUE.
C
```

```
      T = T + DELTAT
      RETURN
      END
      FUNCTION RLUXYT(X,Y)
C
C THIS FUNCTION EVALUATES LU(X,Y,T).  NOTE THAT T IS PASSED IN GCOMON, AND
C THAT (2/DELTAT) MUST BE ADDED TO THE COEFFICIENT OF U BECAUSE ELLPACK THINKS
C THE (-2/DELTAT)U IS PART OF LU.
C
C VARIABLES:
C   X,Y                   -  SPACE VARIABLES AT WHICH TO EVALUATE LU
C   COEFOF(6)             -  COEFFICIENTS OF L, EVALUATED AT TIME T
C   T                     -  TIME AT WHICH TO EVALUATE LU
C   NSTEP                 -  CURRENT STEP
C   CUXX,CUXY,...         -  INDICES INTO COEFOF
C   UXYSAV,UXSAV,UYSAV -  TEMPORARY VARIABLES
C   RLUXYT                -  RETURNED VALUE OF LU(X,Y,T)
C
      REAL COEFOF(6)
      COMMON / GCOMON / T, DELTAT, NSTEP
      INTEGER CUXX, CUXY, CUYY, CUX, CUY, CU
      DATA    CUXX, CUXY, CUYY, CUX, CUY, CU
     A        /  1,    2,    3,   4,   5,  6/
C
C CALL ELLPACK ROUTINE Q1PCOE TO EVALUATE THE COEFFICIENTS OF THE PDE AT
C TIME T AND FILL COEFOF.
C .
      CALL Q1PCOE(X,Y,COEFOF)
C
C IF ON 1ST STEP, NEED INITIAL VALUES (U0 AND ITS DERIVATIVES).  OMIT TERMS
C WITH ZERO COEFFICIENTS IN AN ACTUAL CASE.
C
      IF (NSTEP .EQ. 1) THEN
            RLUXYT = COEFOF(CUXX) * U0XX(X,Y)
     A               + COEFOF(CUXY) * U0XY(X,Y)
     B               + COEFOF(CUYY) * U0YY(X,Y)
     C               + COEFOF(CUX)  * U0X(X,Y)
     D               + COEFOF(CUY)  * U0Y(X,Y)
     E               + (COEFOF(CU) + 2./DELTAT) * U0(X,Y)
C
C ELSE, NEED RESULTS OF PREVIOUS TIME STEP (U,UX,UY,...).  OMIT THOSE
C TERMS WITH IDENTICALLY ZERO COEFFICIENTS IN AN ACTUAL CASE.
C
         ELSE
            RLUXYT = COEFOF(CUXX) * UXX(X,Y)
     A               + COEFOF(CUXY) * UXYSAV
     B               + COEFOF(CUYY) * UYY(X,Y)
     C               + COEFOF(CUX)  * UXSAV
     D               + COEFOF(CUY)  * UYSAV
     E               + (COEFOF(CU) + 2./DELTAT) * U(X,Y)
      ENDIF
      RETURN
      END
*
* DEFINE THE FUNCTION F.
*
      FUNCTION F(X,Y,T)
      F = . . .
      RETURN
      END
*
* DEFINE INITIAL VALUE U0 AND NECESSARY DERIVATIVES; OMIT DERIVATIVES
* NOT APPEARING IN PDE.
*
      FUNCTION U0XX(X,Y)
      COMMON / GCOMON / T, DELTAT, NSTEP
      U0XX = . . .
      RETURN
      END
      FUNCTION U0XY(X,Y)
      COMMON / GCOMON / T, DELTAT, NSTEP
      U0XY = . . .
      RETURN
      END
      FUNCTION U0YY(X,Y)
      COMMON / GCOMON / T, DELTAT, NSTEP
      U0YY = . . .
      RETURN
      END
      FUNCTION U0X(X,Y)
      COMMON / GCOMON / T, DELTAT, NSTEP
      U0X = . . .
      RETURN
      END
      FUNCTION U0Y(X,Y)
      COMMON / GCOMON / T, DELTAT, NSTEP
      U0Y = . . .
      RETURN
      END
      FUNCTION U0(X,Y)
      COMMON / GCOMON / T, DELTAT, NSTEP
      U0 = . . .
      RETURN
      END
*
* DEFINE THE BOUNDARY VALUES UBOUND(X,Y,T).  NOTE THAT T IS PASSED IN GCOMON.
*
      FUNCTION UBOUND(X,Y)
      COMMON / GCOMON / T, DELTAT, NSTEP
      UBOUND = . . .
      RETURN
      END
*
* DEFINE THE FUNCTION TRUE(X,Y,T), IF KNOWN.  NOTE THAT T IS PASSED IN GCOMON.
*
      FUNCTION TRUE(X,Y)
      COMMON / GCOMON / T, DELTAT, NSTEP
      TRUE = . . .
      RETURN
      END
END.
```

5.E THE TRANSISTOR EQUATIONS

*** * * DRAFT DEFERRED * * ***

CHAPTER 6: INTRODUCTION TO THE ELLPACK MODULES

The ELLPACK language and system described so far is only half of the story; the other half is the heart and muscle of ELLPACK, the ELLPACK **modules**. No problem solving system is better than its underlying programs. The design of ELLPACK allows the collection of modules to grow or shrink, so a particular ELLPACK system may have more or fewer modules than presented in Part 2 of this book, Chapters 6, 7 and 9.

In late 1982 the complete ELLPACK system had well over 40 modules; so many that some will find it difficult to choose among them. Part 1 of this book is written with reference to a smaller set, about 18 modules that comprise the basic set. This set includes the more important methods as well as one example of each "variety" of problem solving module. Part 3 of this book, Chapters 11, 12 and 13, illustrate the performance of many ELLPACK modules on a set of 9 model problems. This performance data gives some guidance in choosing modules for a particular problem, but one must keep in mind that it is not possible to predict reliably the relative performance of the modules for any untested problem. This is particularly so if the problem has any unusual features - as most real problems do.

Within the ELLPACK system there are two important collections of modules, the ITPACK software and the YALEPACK software, described in Chapters 7 and 8, respectively. The purposes of these two chapters is to present an overall view of the design, capabilities and methods in the packages. These two chapters are written by some of the developers of these packages; David Kincaid and David Young for ITPACK, and Stanley Eisenstat and Martin Schultz for YALEPACK. User instructions for specific modules in these collections are given in Chapter 9 along with instructions for all the other ELLPACK modules.

The user instructions for ELLPACK modules given in Chapter 9 are intended to include a summary of the modules properties and restrictions. However, there is not enough space to describe the design and methods for each module so references are given to more detailed descriptions. The information for each module is written by the authors of that module except for a few standard programs or simple methods that have been incorporated into ELLPACK. The format for each module description is:

Module Name	
Author's Name or Module Source	
Purpose	A brief statement of what the module does.
Method	A brief summary of the method used. References to more detailed description are usually given.
Parameters	Definiton of the parameters (arguments) of the module.
Restrictions	Summary of the restrictions on the applicability of the method or module.
Performance Estimates	Indicators of the amount of computer resources one can expect the module to use.
References	

The modules are put into five groups, three of which - **Discretization, Indexing and Solution** - correspond to a modular viewpoint of solving elliptic problems as illustrated in Figure 6.1. The fourth group, called **Triples** are modules which solve an elliptic problem entirely by themselves. Triples correspond to methods where it is either inefficient or unnatural to divide the problem solutions into three separate phases. The final group, called **Procedures**, do not correspond to a step in solving the elliptic problem, but rather to some supporting computations. Examples include computing matrix eigenvalues (perhaps to analyze the convergence of an interative method), displaying the pattern of non-zeros in a matrix, and initializing the unknowns (perhaps to initiate some iteration method).

Figure 6.1. Modular viewpoint of solving an elliptic problem. The interfaces between the modules are precisely defined which allows modules to be used in various combinations. The **triple** modules go from Interface 1 to 4 directly.