1986

# Ellpack Project Status Report

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

Wayne R. Dyksen

Elias N. Houstis
*Purdue University*, enh@cs.purdue.edu

Calvin J. Ribbens

Report Number:
86-579

# ELLPACK PROJECT STATUS REPORT

John R. Rice
Wayne R. Dyksen
Elias N. Houstis
Calvin J. Ribbens

CSD TR-579
March 17, 1986

# ELLPACK PROJECT STATUS REPORT

John R. Rice
Wayne R. Dyksen
Elias N. Houstis
Calvin J. Ribbens

CSD-TR 579
March 17, 1986

## ABSTRACT

A status report is given for the ELLPACK system as described in the book *Solving Elliptic Problems Using ELLPACK*. Information given includes distribution statistics, relevant literature, status of modules, new modules, and errata for the book. Tentative plans for a PC implementation (of a smaller ELLPACK system) are also discussed. We describe five new research directions that will not directly affect ELLPACK but which will, someday, produce new systems of the same nature as ELLPACK. These are (1) *Interactive ELLPACK* which is already operational, (2) Use of multiple domains and *domain mappings*, which also has an operational prototype, (3) Development and use of *adaptive tensor product grids*, which also has an operational prototype, (4) An *expert system* for the use of ELLPACK, which has a very preliminary operational prototype, and (5) A *distributed elliptic expert system*, in the advanced design stage, to solve elliptic PDEs in a heterogeneous computing environment including multiprocessor and vector machines.

CONTENTS

## I. OVERVIEW

This report presents an overview of the ELLPACK system which is described and documented in *Solving Elliptic Problems Using ELLPACK* by John R. Rice and Ronald F. Boisvert (The *ELLPACK book*).

All but three of the 61 modules described in the ELLPACK book are now working reliably and distributed with the current system. In addition two new modules will be included in the distributions later this year. The first is *SPLINE COLLOCATION* (by E. N. Houstis, J. R. Rice and M. Vavalis) which uses a new method and is more efficient than any of the other finite element methods in ELLPACK. The second is *HODIE for general domains* (by Robert E. Lynch) which uses a new, high order difference method. It provides a dramatic increase in efficiency compared to other finite difference methods and is more efficient than most finite element methods (often dramatically so).

A brief discussion is given of plans to move a version of ELLPACK to personal computer (such as an IBM AT or equivalent). Errata for the ELLPACK book are given along with a complete bibliography.

The distribution data given shows that ELLPACK now is at 67 sites (46 university, 8 government lab and 13 industry) in the US and 17 foreign countries. It has been installed on equipment from at least 12 different computer manufacturers.

The second part of the report provides overviews of five new research directions based on ELLPACK. This will not directly affect the ELLPACK system itself but might, someday, produce new systems of the same very high level, powerful nature. These projects are:

1. *Interactive ELLPACK.* A prototype of this system is operational (Wayne Dyksen and Calvin Ribbens). It allows one to take a particular elliptic problem and interactively solve various instances of it. Parameters that can be varied include grid sizes, methods, and constants in the PDE or boundary. A sophisticated color graphics facility is used which allows one to view various solutions, grids and/or related functions. These items may be also saved for later use or printed immediately.

2. *Domain Mapping Methods and Facilities.* ELLPACK has been extended to allow multiple domains to be used for a single PDE. Mappings may be given explicitly or used from ELLPACK modules in order to create more efficient methods or to transform the problem to an instance where existing methods apply. A prototype of this system is operational. (Calvin Ribbens)

3. *Adaptive Tensor Product Grids.* Tensor product grids are of interest for two reasons: (a) Many methods are only defined or implemented on rectangular grids, (b) Parallel and vector methods are much easier to design for such grids. Adaptive methods have been developed to create such grids for a variety of problems (e.g. singular or boundary layer problems), some of these included in the prototype multidomain ELLPACK system. (Calvin Ribbens and John Rice)

4. *Elliptic-Expert and ELLPACK advisor.* The ELLPACK system user needs to be knowledgeable about numerical methods for solving elliptic problem. He is unlikely to choose HODIE HELMHOLTZ, REDUCED SYSTEM SI or DYAKANOV CG4 if he has never heard of these terms. A user might well use 5 POINT STAR just because he knows it and feels comfortable with it. The Elliptic-Expert system is designed to guide the user to the best (most efficient) method for his elliptic problem, it is a traditional application of an expert system front end backed up with ELLPACK's very large data base of information about how various modules perform on various problems. A preliminary prototype is operational (Wayne Dyksen).

5. *Distributed Elliptic-Expert.* This project encompasses all the preceding ones plus the problem of deciding what computer to use for solving an elliptic problem. If the computer chosen is a multiprocessor, then Distributed Elliptic-Expert will provide automatic help in

mapping the computational structure onto the hardware structure so as to achieve efficient use of the computer. This project is in the advanced designs and analysis stage (Elias Houstis, Wayne Dyksen and John Rice).

## II. THE ELLPACK SYSTEM

### II.A Status of Modules

There are 61 modules listed in the ELLPACK book. At this time all but three work with reasonable reliability. The capacitance matrix modules CMM EXPLICIT, CMM HIGHER ORDER and CMM IMPLICIT do not interface reliably with the data structure of the DOMAIN PROCESSOR and have not yet been included in the ELLPACK systems distributed. The three dimensional capability of HODIE will be installed for distributions later in 1986.

We occasionally receive reports of problems with modules. Sometimes these are simple defects in the documentation, the module interface or portability. So far, we have corrected these simple items. Sometimes there are problems internal to the modules themselves and we forward the problem to the module authors. In some cases, modules have been modified to correct these problems and later distributions of the ELLPACK system have the corrected modules.

### II.B New Modules

The ELLPACK group has developed two new methods which are being incorporated into the ELLPACK system. These will be included in ELLPACK systems distributed later in 1986. No modules have been submitted to us by others as candidates for inclusion in the ELLPACK system. The new modules are briefly described here, more details are given in the appendices using the format of the ELLPACK book.

*SPLINE COLLOCATION.* This is a collocation method using bicubic splines which is analogous to the method used in HERMITE COLLOCATION and INTERIOR COLLOCATION using Hermite bicubic polynomials. It also produces fourth order accuracy on rectangular domains. The derivation and analysis of the method is considerably more complex than that of collocation with Hermite bicubics, the equations derived are also rather complex. However, the resulting linear system to be solved is smaller for the same accuracy and the overall result is a more efficient method. This method and module are due to Elias N. Houstis, John R. Rice and Manolis Vavalis.

*HODIE (for general domains).* The HODIE (Higher Order Difference) methods have consistently been among the most efficient methods for rectangular domains. For some years Ron Boisvert and Robert Lynch have independently been studying ways to extend these methods to general PDE's on non-rectangular domains. The goal has been to obtain fourth order accuracy. The basic difficulty is how to accurately approximate mixed boundary conditions along curved sides. Fairly early a method to achieve second order accuracy on special boundaries was found by Boisvert. This is a great improvement over standard textbook approaches, but other second order methods based on finite differences have also been found. Recently Lynch has found a new approach using the HODIE method which gives high accuracy for general boundary conditions along curved boundaries. The HODIE module for general domains achieves fourth order accuracy for almost all elliptic problems and is dramatically more efficient than 5 POINT STAR for general domains. It is also usually more efficient than COLLOCATION for general domains.

### II.C ELLPACK on a Personal Computer

Inexpensive personal computers now have the resources (memory and speed) to solve substantial scientific problems. The ELLPACK group has initiated the design of a specialized version for IBM (or compatible) PC with a hard disk. This project is still in progress and will probably involve collaboration with some people outside the ELLPACK group. There are three main design issues: (1) How much should be removed from ELLPACK, (2) How much graphics

should be supported, and (3) Should the interactive facilities (see Section III.A) be included. It is expected that PC ELLPACK will have perhaps 20 of the 60+ ELLPACK modules, it will have graphics supported in only one environment and some interactive features will be included.

The initial choice of modules is as follows:

DIS. COLLOCATION
5 POINT STAR
INTERIOR COLLOCATION

IND. AS IS

SOL. BAND GE
JACOBI CG

TRI. HODIE FFT
MULTIGRID MG00
SET

PROC. LIST MODULES

MENU.

As the MENU segment indicates, the PC version will be interactive.

**II.D Errata for the ELLPACK Book**

Writing books is similar to writing big software systems in at least one respect, one never gets all the errors out. Appendix 1 contains errata of *Solving Elliptic Problems Using ELLPACK* complete as of January 1, 1986. Additions to this errata list are greatly appreciated.

**II.E Distribution Data**

ELLPACK is distributed by Purdue University for a nominal license fee of $250/year plus an initial shipping charge. For this one receives an ELLPACK tape, two copies of the ELLPACK book and occasional information about ELLPACK (such as this report). The fees in no way pay for the development of the ELLPACK system (it now has over 125,000 lines of Fortran and templates). It does provide some modest support for graduate students, travel and small equipment items.

ELLPACK is not marketed nor maintained as a commercial product would be. Thus the distribution is not very large and the support given users is low. We realize that many users do not have the expertise to install ELLPACK easily nor to make the local modifications sometimes required to achieve fast performance. We do not have the resources to provide much help to such people. We can say that ELLPACK has been installed on a wide variety of systems with a reasonable level of performance. A summary of the installations as of February 1, 1986 is given below.

**Summary of Organizations with ELLPACK** February, 1986

| | US | Foreign | |
|---|---|---|---|
| University | 29 | 17 | |
| Government | 2 | 6 | |
| Industrial | 7 | 6 | |
| Total | 38 | 29 | = 67 |
| **Computer Makes** | | | |
| Amdahl | 1 | 0 | |
| ATT | 1 | 0 | |
| CDC | 9 | 5 | |
| Cray | 1 | 1 | |
| DEC | 8 | 10 | |
| Gould | 1 | 0 | |
| IBM | 11 | 8 | |
| ICL | 0 | 1 | |
| NAS | 1 | 0 | |
| Ridge | 1 | 0 | |
| Sperry | 0 | 1 | |
| Tektronix | 1 | 0 | |
| UNIVAC | 2 | 2 | |
| Unknown | 5 | 2 | |

The 18 foreign countries with ELLPACK are: Austria, Austrialia(2), Canada (3), China(2), Denmark, England(5), Finland(2), Germany(2), Greece, Israel, Italy, Japan, Netherlands, South Africa, Sweden(2), Switzerland, and Yugoslavia

### II.F ELLPACK Literature

We list publications (see Appendix 2) which discuss the ELLPACK system itself or its use in some general way (such as evaluating software or an application). This list does not include any items which refer to the creation, analysis or design of a particular numerical method or its associated module.

## III. NEW DIRECTIONS

This section presents brief summaries of the five new research directions associated with the ELLPACK project.

### III.A Interactive ELLPACK

A new *menu* segment has been added to ELLPACK to make it possible to write interactive ELLPACK programs. The menu segment allows the user to specify several different methods or procedures to use in solving a problem, and interactively choose from them at run time. The control program generated from an interactive ELLPACK program contains code to print these

choices in a menu, prompt for the desired choice, and execute the appropriate code.

### Syntax

An ELLPACK menu segment is given as:

```
menu.     '<menu name>'
          <menu item>
                .
                .
          <menu item>
```

The title for the menu is given by <menu name>. One or more <menu item>s follow, specifying the choices to be listed in the menu. Each <menu item> is of the form:

```
'[<key>] : [<label>]'    <item def>
```

where <key> is an optional key (the user enters this to select the item at run time), and <label> is an optional name for the item. The default for <key> is an integer such that the items in each menu are numbered sequentially. The default for <label> is a meaningful string from <item def>. The item definition <item def> may include one or more of the following ELLPACK segments: GRID, DISCRETIZATION, INDEXING, SOLUTION, TRIPLE, OUTPUT, PROCEDURE, or FORTRAN. An <item def> may extend over several lines. The only restriction is that segment names within menus do not appear in column one. Recall that segment names which appear outside of a MENU segment must begin in column one. In fact, the preprocessor assumes a menu specification continues until it finds a segment name which begins in column one.

### Example

The following example illustrates most of the features of MENU. Notice that the menu choices *continue*, *return* and *exit* are automatically included in every menu. In a UNIX environment, the user can escape to the shell by "!command". The Interactive ELLPACK program

```
**************************************************************************
*                                                                        *
*       sample interactive ellpack program                               *
*                                                                        *
**************************************************************************

equa.   uxx + uyy = 0.0

boun.   u = true(x,y) on x=0.
                      on x=1.
                      on y=0.
                      on y=1.

grid.   5 x points
        5 y points

menu.   'Discretization'
        '5ps:finite differences'              dis.  5 point star
        'hbc:hermite bicubic collocation'     dis.  interior collocation
        'hod:high order finite differences'   dis.  hodie

menu.   'Solution'
        ':'             sol.    band ge
        ':'             sol.    linpack spd band
        ':'             sol.    sor
```

```
menu.    'Output'
         ':maximums'    out.    max(true)
                                max(error)
                                max(residu)
         ':table u'     out.    table(u)

subprograms.
         function true(x,y)
         true = x*x - y*y
         return
         end
end.
```

produces the following interactive menus on a standard terminal:

```
 ****************************************
 *                                      *
 *   discretization                     *
 *                                      *
 ****************************************

 5ps :  finite differences
 hbc :  hermite bicubic collocation
 hod :  high order finite differences
 c   :  continue
 r   :  return
 x   :  exit

 Option? c


 ****************************************
 *                                      *
 *   solution                           *
 *                                      *
 ****************************************

 1 :  bandge
 2 :  linpackspdband
 3 :  sor
 c :  continue
 r :  return
 x :  exit

 Option? c


 ****************************************
 *                                      *
 *   output                             *
 *                                      *
 ****************************************

 1 :  maximums
 2 :  table u
 c :  continue
 r :  return
 x :  exit

 Option? x
 Are you sure (y/n)? y
```

### Interactive Graphics Output

Interactive ELLPACK has been implemented on a number of graphics devices. A new ELLPACK option *terminal* has been added; currently, terminal can be one of *dumb*, *tek4105*, *tek4107*, *tek4115* and *ridge*. Since ELLPACK is "template driven", the ELLPACK control program can be tailored for the specific graphics device on which it is to be run. On a dumb terminal, graphics output is written to a file in UNIX plot(5) file format. On the Tektronix terminals, color graphic output is written to predefined graphics views; the number of colors and views is terminal specific. Graphics output is stored in segments local to the terminal; hence, the views can be manipluted locally (moved, copied or deleted). Graphic output can be saved in a file either to be viewed during another Interactive ELLPACK session or to be printed on a color printer. On the Ridge display (a bit-mapped device), graphics output is displayed in windows which can be manipulated with a mouse. Menu items are selected from pop-up menus with a mouse.

### Interactive Graphics Input

Interactive ELLPACK uses interactive graphics input to allow the user to specify grids. If the keyword *interactive* appears in a GRID segment, the domain with the current grid is displayed; grid lines may be added or deleted by positioning a crosshair using a joystick, thumbwheels or a mouse. In a non-graphic environment, the user is simply prompted for the information which defines the grid.

Figure III.A.1 shows a screen from an Interactive ELLPACK session. The top four views show the true solution for a problem along with the error contours of three numerical solutions whose grids are shown in the second row of views. The lower right view is an enlargement of view 2. The MENU for manipulating plots is seen in the lower left view.

## III.B Domain Mapping Methods and Facilities

A second major new research direction which relates to ELLPACK is the development of *MULTIDOMAIN ELLPACK* (MDE). MDE is a problem solving system based on ELLPACK in which more than one domain can be used to solve an elliptic problem. Just as in ELLPACK, MDE allows the user to state an elliptic PDE in a convenient very high level language, and then choose from a wide array of problem solving modules. Standard ELLPACK computes approximate solutions to problems on general two-dimensional domains or in three-dimensional boxes. For two-dimensional problems, MDE extends the power of ELLPACK by allowing the use of more than one domain in the solution of a single problem. Mappings may be defined between these domains, and MDE handles the transformation of the PDE automatically. Obviously, the goal is to improve the solution process in some way through the use of these problem transformations. MDE allows the use of well-known mapping techniques; perhaps more importantly, it facilitates experimentation with new mapping techniques. One particular area of interest has been in the application of problem transformations which enhance the possibilities for vector or parallel algorithms. An experimental version of MDE has been built, and used extensively in this research.

*Domain mapping and problem transformations*

The idea of transforming a PDE through a mapping is classical. It is helpful however, to describe briefly a typical example. Figure III.B.1 shows a domain $\Omega$ in $(x,y)$ space, and its image $\Omega^*$ under a mapping $F$, in $(s,t)$ space. A uniform grid placed on $\Omega^*$ is mapped as shown to the curvilinear grid on $\Omega$.

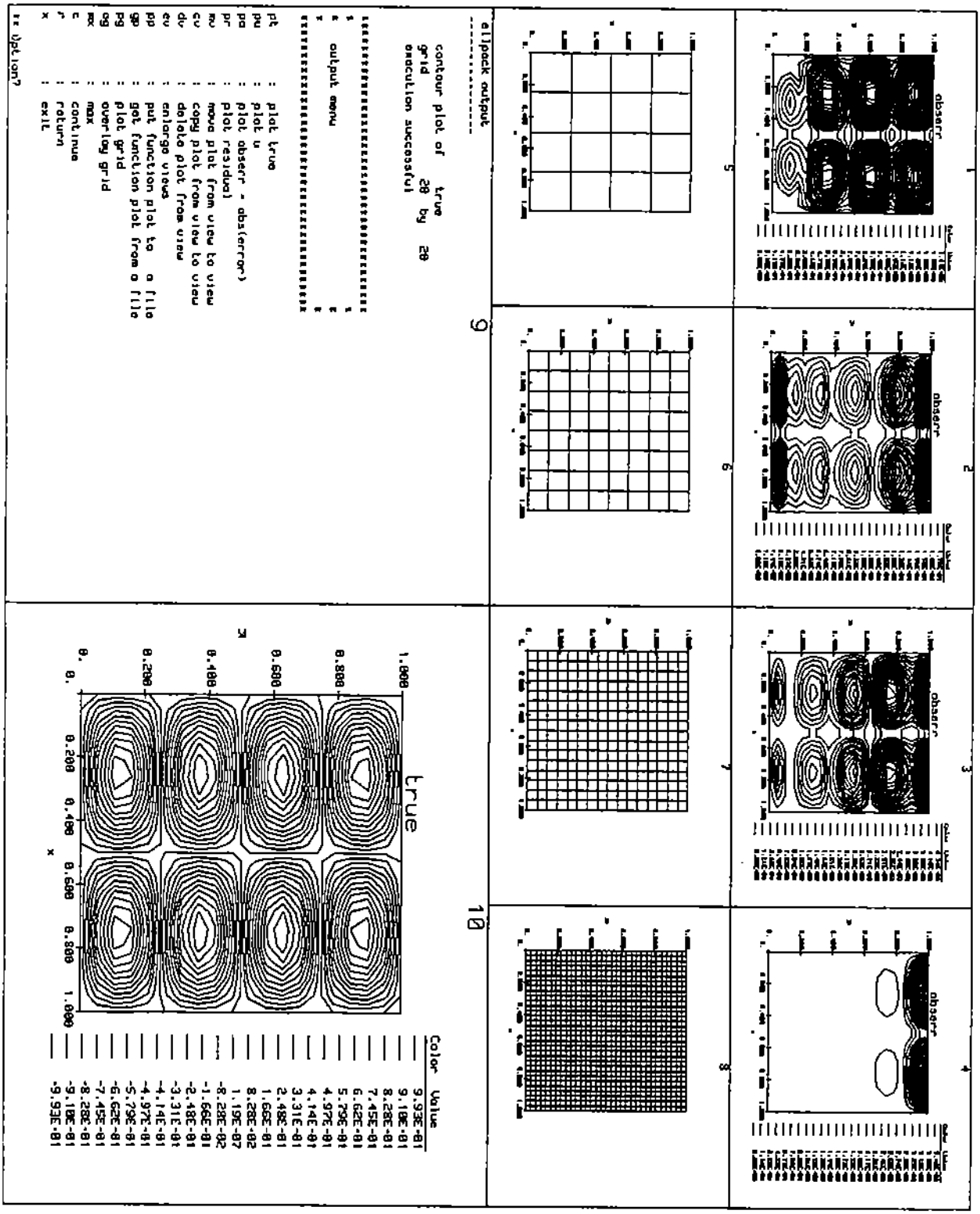


Figure III.A.1 Screen from a Tektronics 4115 during an Interactive ELLPACK session. The color coded contours are shown in black here, the screen resolution is good enough for all the numbers to be read.

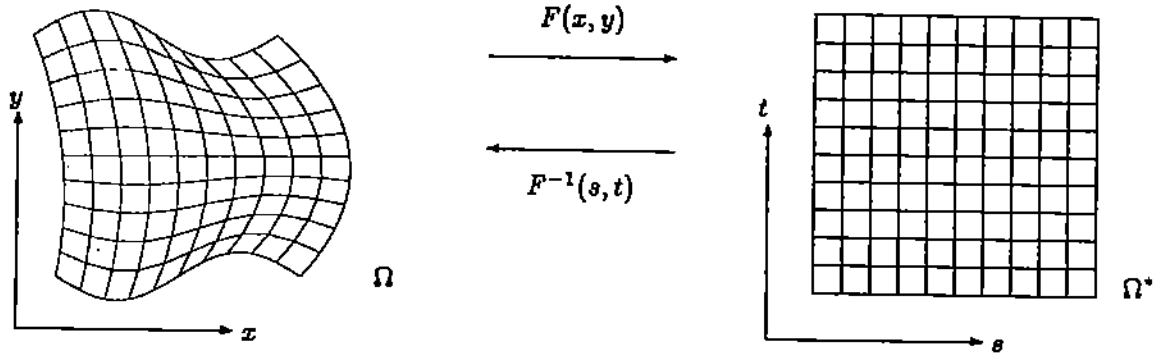


Figure III.B.1 Example of a domain transformation showing problem domain, transformed domain, and mapping between them.

Transforming the problem from $\Omega$ to $\Omega^*$ means a transformation in the PDE itself. Suppose we are given the two-dimensional elliptic problem

$$\begin{aligned} Lu &= f \quad in\ \Omega \qquad \text{(elliptic equation)}, \\ u &= g \quad on\ \partial\Omega \qquad \text{(Dirichlet boundary conditions)}. \end{aligned} \tag{1}$$

The mapping $F:\Omega\rightarrow\Omega^*$ may be represented by a change of coordinates $s=s(x,y)$ and $t=t(x,y)$. So we have $F(x,y)=(s(x,y),t(x,y))$. Similarly, $F^{-1}$ is represented by $x=x(s,t)$ and $y=y(s,t)$. Applying this change of variables to (1) yields

$$\begin{aligned} L^*v &= f^* \quad in\ \Omega^*, \\ v &= g^* \quad on\ \partial\Omega^*, \end{aligned}$$

where $v(s,t)$ is the new dependent variable and the superscript * indicates that a change of coordinates has been made in the function or operator.

*User Interface*

A new segment called MAP, has been implemented in MDE. With it a user can specify a mapping to a new domain. The syntax is similar to other problem solving segments in ELLPACK:

MAP. < module name > ( < optional parameters > )

Modules of this type differ in the type of map they define or in the way in which the map is computed. Implementation of the transformations requires that each MAP module provide some way of computing the mapping $F$ from $\Omega$ to $\Omega^*$, its inverse $F^{-1}$, and $\partial F$ and $\partial^2 F$, the first and second

order partial derivatives of $F$. Frequently, some of these functions are known only approximately. MAP modules installed or under development for the MDE prototype include EXPLICIT, BLENDING, REMOVE CORNER, POLAR, and LAPLACE. It is a simple task to add additional MAP modules to the system.

The new domain is defined implicitly as the image of the previous domain under the new map. Only the original domain is given explicitly in a BOUNDARY segment. Subsequent domains are defined by the map or sequence of maps used. When a MAP module is executed the new domain becomes the "current" one, and any subsequent solution of the PDE will operate on the newly transformed problem.

Output is available on any of the domains that have been defined. The domains are referred to by number; 1 is the first, 2 the second, and so on. The OUTPUT segment of standard ELLPACK has been extended to allow the specification of a domain number. A typical OUTPUT segment in MDE has the form

OUTPUT. < type > ( < function > , < grid > ) ON < domain number >

The default domain number is the current domain. MDE automatically handles the change of variables needed to evaluate a function on a particular domain when that function is defined on a different domain. A typical MDE program fragment is shown in Figure III.B.2.

```
equ.  uxx + uyy = 0.0
bou.  u = true(x,y)    on line 0.0,1.0 to 0.6,1.0 to 1.0,0.7
                           to 1.0,0.0 to 0.0,0.0 to 0.0,1.0
map.  remove corner(npc=5,angle=pi/2.)
gri.  9 x points -1.0 to 1.0  $  9 y points  0.0 to 1.5
dis.  collocation
sol.  band ge
out.  plot domain on 1        $  plot domain on 2
      max(error,20,20) on 1   $  plot(u)
```

Figure III.B.2 Section of a typical Multidomain ELLPACK program.

*Implementation*

As mentioned above, MDE is an extension of ELLPACK. As such it includes the library of pre-compiled problem solving modules available with ELLPACK. Of course, under the most general transformations many of the DISCRETIZATION or TRIPLE modules no longer apply because of the increased complexity of the transformed operator. The prototype MDE is actually an extension of Interactive ELLPACK (see Section III.A). In particular, the MENU segment and various graphics interfaces are available. Convenient and immediate graphics are particularly valuable in a multi-domain setting, where the changing geometry of the problem is often of great interest.

The new MAP segment is defined by a simple addition to the preprocessor grammar which defines the ELLPACK language. MAP modules are added to the ELLPACK Definition File in a way exactly analogous to other problem solving modules. These entries give the FORTRAN

code needed to call a MAP module from the ELLPACK control program, and information about array dimensions and any special declarations required by the new module.

Following the standard notion of "interfaces" between the various "software parts", MDE defines a new "mapping" interface. It contains variables and arrays which record information such as the current working domain, the domain where the most recent solution was computed, and data which is used to keep track of each domain and mapping present. In addition to these new data structures, the multiple domain capabilities of MDE require some modification to the standard subprograms which define the PDE. The subroutines which return problem and boundary operator coefficients have been modified to allow for domain transformations. Similar modifications were needed in the functions which evaluate the problem and boundary condition right side functions. The subroutine which contains the boundary parameterization was also modified to reflect the presence of maps. It is significant that these changes, as well as the new mapping interface, are all taken care of by the MDE preprocessor. No changes to existing ELLPACK library software were required. The multiple domains are handled in such a way that the existing problem solving modules can be used without modification. This was an important design criterion for MDE.

### III.C Adaptive Tensor Product Grids

This section describes work that is being done in the area of adaptive grid methods for elliptic PDEs. The idea of adapting the grid to a particular problem is very powerful and widely-used in many applications. A common approach is to add mesh points near where the problem is difficult. A more accurate term for this tactic might be grid "refinement". One of the major challenges of grid refinement is to keep track of the changing grid. The data structures and programming needed to handle fairly arbitrary grid refinements can be very complex. Yet this is frequently done, with good results.

With the advent of various parallel and vector architectures however, this programming challenge is even greater. We believe it will be difficult, if not impossible, to develop efficient parallel and vector algorithms which use general grid refinement strategies. These grid refinement methods increase the complexity of the data structures and algorithms to such an extent that vectorization or parallelization becomes very difficult. The inherent parallelism of an algorithm based on a regular tensor product grid seems much easier to exploit. Yet the power of general grid refinement is clearly something that should not be given up so easily. One would like to develope adaptive grid methods which somehow preserve the tensor product structure of regular grids. We consider another common approach to grid adaption. Rather than adding new grid points, the original grid points can be moved so that more of them are located in the difficult regions. This preserves the tensor product nature of the grid, at least logically. In general, no new points are added, unless the entire mesh is refined uniformly. The result is a deformed tensor product grid which is adapted to the given problem. Strictly speaking, this grid is no longer uniform or regular either. The grid lines for example, are curves rather than horizontal or vertical lines. However, introducing a change of variables, we can view the grid as uniform and regular in terms of these new curvilinear coordinates. The next section expands on this idea. We then describe a prototype implementation of this scheme which is being used to investigate various adaptive tensor product grid methods.

#### Problem transformation

Our approach to tensor product grid adaption is to view it as a change of variables or problem transformation. Conceptually, we use two domains to solve a problem (see Figure III.C.1). The PDE is posed on a *problem domain* $P$. The problem is transformed via a mapping to the

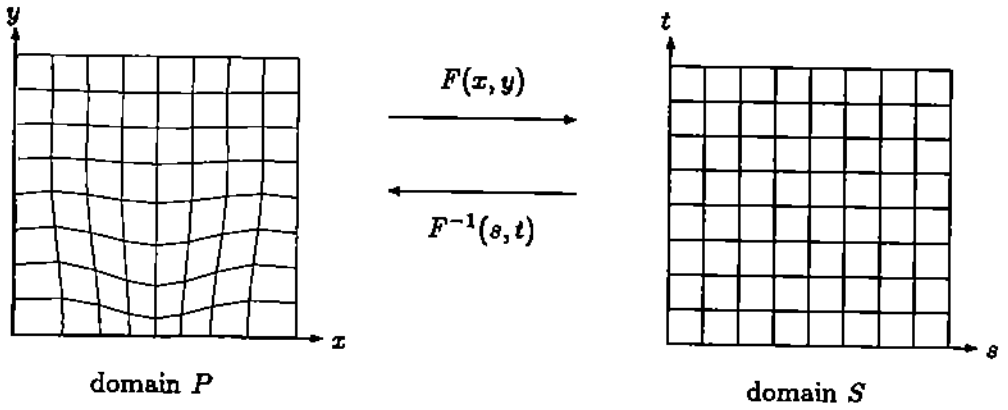


Figure III.C.1 Two Domains for Grid Adaption

*solution domain* $S$. A bivalued function $F$ maps points in $P$ to points in $S$; its inverse is $F^{-1}$. A fixed uniform tensor product grid is used to discretize the transformed PDE on $S$. $F^{-1}$ maps points in this uniform grid onto a curvilinear grid in $P$. The mapping is chosen so that this curvilinear grid is adapted, in some sense, to the given problem. Obviously, the task of chosing $F$ (or $F^{-1}$) to optimize the distribution of points is a nontrivial one. Once it is done however, we can proceed to solve the transformed problem in $S$. The brief discussion of transformed PDEs in Section III.B applies in this case as well.

**Multidomain ELLPACK implementation**

Having posed the grid adaption problem in terms of domain mappings, it fits naturally into the Multidomain ELLPACK (MDE) system described in Section II.B. Toward that end, a new MDE segment ADAPT has been implemented:

ADAPT. < module name > ( < optional parameters > )

Each of the ADAPT modules defines a mapping from the current problem domain to a new solution domain, where the problem can then be solved with a uniform tensor product grid. There are currently four such modules.

Each of the ADAPT modules takes basically the same approach to determining the map. Rather than searching directly for a suitable function $F$, they simply relocate grid points in $P$ according to some criterion based on the given problem. In practice, the grid used to select the adaptive mapping need not be as fine as the discretization grid. It need only be fine enough to allow a sufficiently flexible and wide array of mappings. Given this new curvilinear grid in $P$, the inverse mapping $F^{-1}$ is chosen to best approximate the new point distribution. We use a piecewise bicubic spline to represent each coordinate of $F^{-1}$. This choice provides enough flexibility to approximate a wide variety of point distributions while still maintaining the continuity needed to transform the problem smoothly. The coefficients of $F^{-1}$ are determined by least squares; there are typically many more grid points in $P$ than coefficients in $F^{-1}$. The least squares step also serves to smooth the mapping, in case the points were moved without much concern for smoothness. The tensor product formulation of the bicubic splines is exploited to make the least squares step inexpensive.

Once a new $F^{-1}$ is chosen, the transformed problem may be solved. The first and second order partial derivatives of $F^{-1}$ are used to compute the coefficients of the transformed problem. If $F$ at some point $(x,y)$ in $P$ is needed, we use a special two dimensional secant method to invert $F^{-1}$ numerically. This is not needed to discretize the transformed PDE; it is needed when

output on the original problem domain is desired. The numerical inversion step converges rapidly since a good initial guess is normally available.

The various ADAPT modules differ in how they determine the new point distribution. The least automatic method is an "interactive" one in which the user may move points or lines himself. It is based on the interactive GRID segment written by Wayne Dyksen for Interactive ELLPACK. More sophisticated methods under development take either a "local" or "global" approach. In both approaches, we seek to balance a "density" function over the problem domain. The density function may be any function that reflects the difficulties in the problem or the "goodness" of the solution. In a local method, points are moved according to the "influence" exerted on them by the densities at neighboring points. In a global method, an attempt is made to define a new point distribution based on density values taken over the whole domain.

### III.D Elliptic-Expert and ELLPACK Advisor

The principal goal of ELLPACK is to create a modern, very high level system for scientific computation. The size and complexity of feasible scientific computations has increased dramatically in the last twenty-five years, and yet, over the same time, the process by which scientists and engineers do scientific computing has changed relatively little. A significant improvement in the scientist/scientific computing interface can be realized via very high level systems such as ELLPACK . ELLPACK is a versatile *very high level language* (VHLL) or *problem solving environment* for solving elliptic PDEs. Although ELLPACK contains vast "raw" PDE solving power, there are two fundamental factors limiting its potential as a problem solver. First, ELLPACK is strictly "batch" oriented, and secondly, it takes an "elliptic expert" to make full use of ELLPACK's problem solving powers.

We already have a running prototype of *Interactive ELLPACK*. It features simple "problem definition" input, interactive grid generation, interactive elliptic problem solving and interactive color graphics. This results in a tremendous boost in our productivity and is one step in improving scientific computing (at least for elliptic problems).

#### Elliptic-Expert

Although interactive ELLPACK represents a significant improvement it provides no help in choosing a solution method. For a given scientific computing problem, the selection of the "best" solution algorithm is difficult for the average nonexpert. The need for confidence in results thus dictates electing inferior "known" algorithms over superior "unknown" ones.

*Elliptic-Expert* is an extension of Interactive ELLPACK which advises the user in the selection of the "best" algorithm for solving an elliptic problem. The interaction between the Elliptic-Expert system and the user includes the following areas:

1. Elliptic problem definition
2. Objectives and constraints for the computaions
3. Metaknowledge
4. Solution method (if the user wants to give guidance)
5. Output specifications

For example, a simple Ellipic-Expert problem is stated as follows:

```
OPTIONS.
        accuracy = 0.05
        total time < 1 hour
        solution is periodic, smooth
EQUATION.  - uxx - uyy - (20*pi**2)u = 0
BOUNDARY.  u = 0              on x = 0
                        on x = 1
                        on y = 0
        uy = 4*pi*sin(2*pi*x)   on y = 1   end.
```

The architecture of Elliptic-Expert is shown in Figure III.D1.

We already have a knowledge-base of performance of methods in over 10,000 elliptic problem solutions. A prime research effort for us is that of using our knowledge-base of the performance data plus an inference engine to create an expert system for elliptic problems.

The knowledge-base contains facts about the following three main areas:

1. Techniques to Approximate the Elliptic Problem
   - Applicability to elliptic problem
   - Effect of grid on discrete problem
   - Convergence properties
   - Discrete problem properties
   - Resource requirements
   - Relative ranking from performance data

2. Linear Equation Solution Modules
   - Effect of and/or need for indexing
   - Applicability to discretization
   - Resource requirements
   - Relative ranking from performance data
3. Module Documentation

The knowledge-base also contains an extensive set of rules for selecting the best ELLPACK elliptic solver for a particular problem.

Elliptic-Expert's inference engine has two modes. In *heuristic mode* the solution method is selected only on the basis of *a priori* knowledge; e.g., symbolic analysis of the elliptic problem and past performance knowledge. In *algorithmic mode*, dynamic strategies based on both *a priori* and *a posteriori* knowledge are used; e.g.,

1. side calculations to study the behavior of the coefficients, forcing function and boundary data,
2. trial, low accuracy, cheap solutions,
3. trial solutions using the 2 or 3 "most promising" methods, and

**Elliptic-Expert**

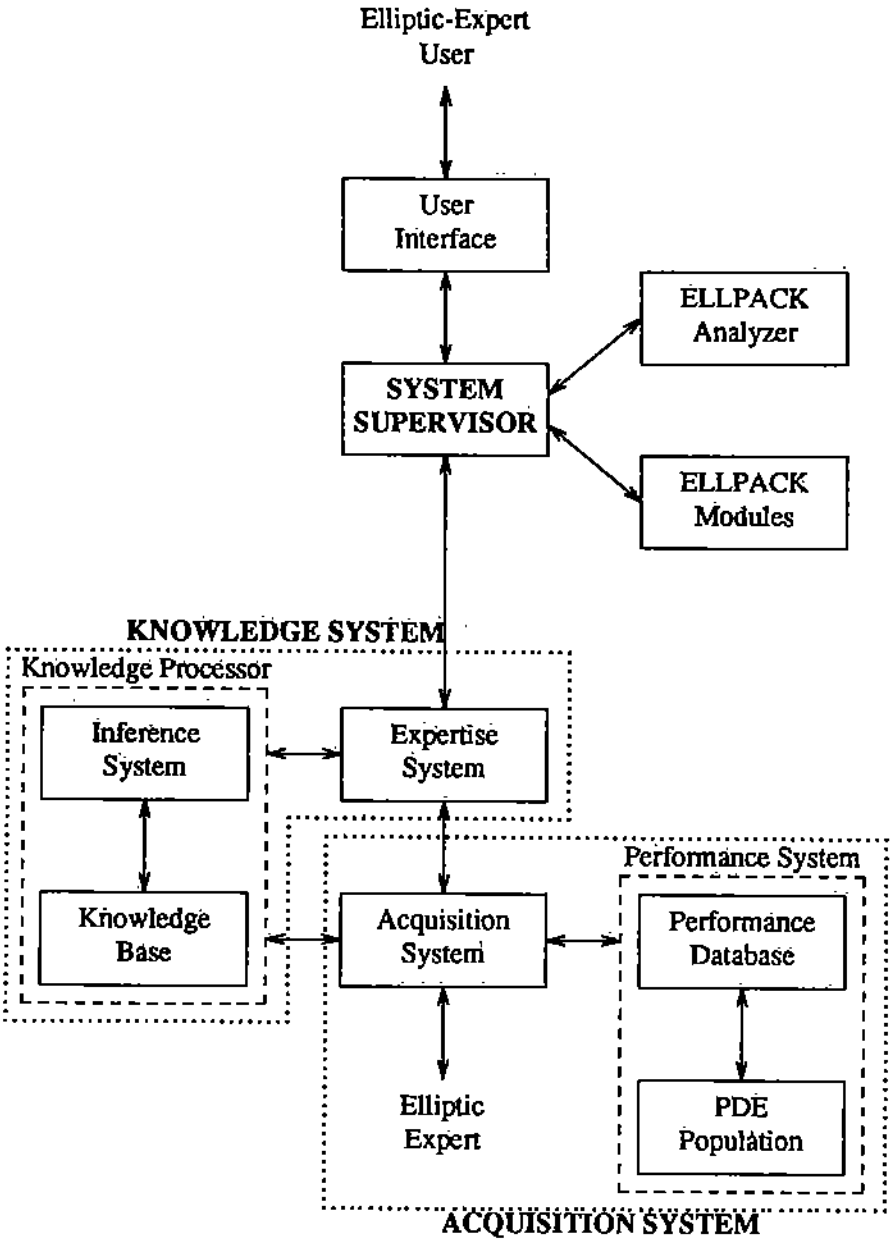


Figure III.D1. Schematic of Elliptic-Expert showing the user interface (top) and the expert system for analyzing PDE and selecting methods (bottom).

4. additional input/advice from the user after presentation of the initial results.

Elliptic-Expert has an Acquisition System to gain knowledge-base facts and inference rules for the Knowledge Processor. This knowledge comes from three sources: the elliptic expert, the Expertise System as it learns from its actions, and the ELLPACK Performance Evaluation System [BHR79], [RHD80].

The System Supervisor gives the problem definition to ELLPACK analyzer which returns the *problem interface*—that is, the encoded elliptic problem. This interface is then passes to the Expertise System along with requests for advice and/or explanation. Using the Knowledge System, the Expertise System then makes suggestions to the user. If no performance data is available for this type of problem, the Acquisition System is requested to generate some via the Performance Evaluation System. If the problem is deemed to be interesting and different, it is added to the PDE population. After the user chooses a solution method, the System Supervisor passes the *solution interface* to the appropriate ELLPACK module which returns the solution. Naturally, the Expertise System contains an explanation system to "justify" its actions.

References

[BHR79] Boisvert, R.F., Houstis, E.N., and Rice, J.R., "A system for performance evaluation of partial differential equations software," *IEEE Trans. Software Eng.*, 5(1979), 418-425.

[DYK85] Dyksen, W.R., "A tensor product generalized ADI method for elliptic problems on cylindrical domains," Purdue University, Department of Computer Science Report CSD-TR-495, March 1985, to appear, *J. Comp. Appl. Math.*

[RHD80] Rice, J.R., Houstis, E.N., and Dyksen, W.R., "A population of linear, second order, elliptic partial differential equations on rectangular domains, Parts 1 and 2," *Math. Comp*, 36(1981), 475-484.

### III.E Distributed Elliptic-Expert

This project is a direct continuation of the Elliptic-Expert project described above. To avoid repeating most details, we summarize the Elliptic-Expert system as follows:

1. It allows a natural statement of a complex scientific problem to be solved.
2. It allows simple statements about computational objectives (accuracy, output desired, resource constraints).
3. It then selects a method to solve the problem and produces the results or states that the objectives are infeasible.

The goal of such *problem solving environments (PSE)* is to provide the scientist with the benefit of a large body of sophisticated software (the problem solving modules in Elliptic-Expert consist of 100,000+ lines of Fortran) and the benefit of accumulated expertise (there are hundreds of rules used plus a database of performance measures for 10,000+ problem solutions.

However, Elliptic-Expert does nothing to assist with the equally formidable complexities of the next generation of supercomputers. Just as the potential for conventional computers is difficult to realize without elliptic-PDE-knowledgeable users, the potential of supercomputers is difficult to realize without supercomputer-knowledgeable users. The DE2 system is to do for hardware system complexity what Elliptic-Expert does for software/method complexity; hide it.

Figure III.D2 shows the overall structure of the DE2 system and Figure III.D3 expands upon the computer network that exists at Purdue. Without further generalities, we describe our approach to key problems: construction of an *architecture compiler* for a problem solving

envirnoment.

## Distributed Elliptic-Expert

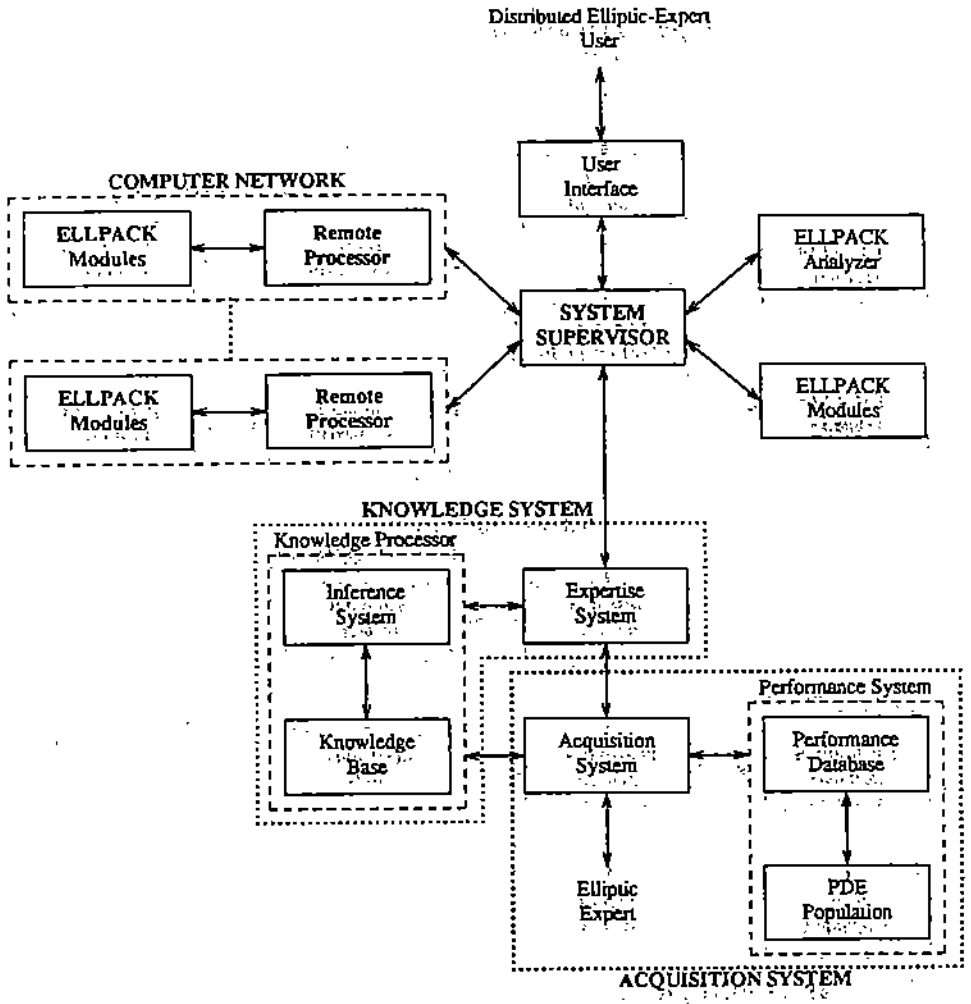


Figure III.D2 Schematic of DE2 showing the user interface (top right), the expert system for analyzing PDEs and selecting methods (bottom) and the connection to the computer network.

At the end of its analysis, the PSE knows a great deal about the computation; its size, its input/output, the software modules to be used, etc., as well as the heterogeneous network facility. Assume that this information is represented in a quantified form such as annotated graphs. One then has to map the computation graph into the architecture graph subject to all the constraints on size, sequencing and user specifications.

Once a machine is selected (e.g., a multiprocessor) for a computational module further restructuring is feasible. The PSE has not yet created the load module for execution, so it can restructure the module to fit the architecture better. An obvious, but very effective, tactic is as follows: we have a submodule that can do $K$ things, we have $N^2$ of these things to do and $P$ processors. We create a load module with $P$ copies of the submodule, each assigned to one processor

## Distributed Elliptic-Expert's Computer Network

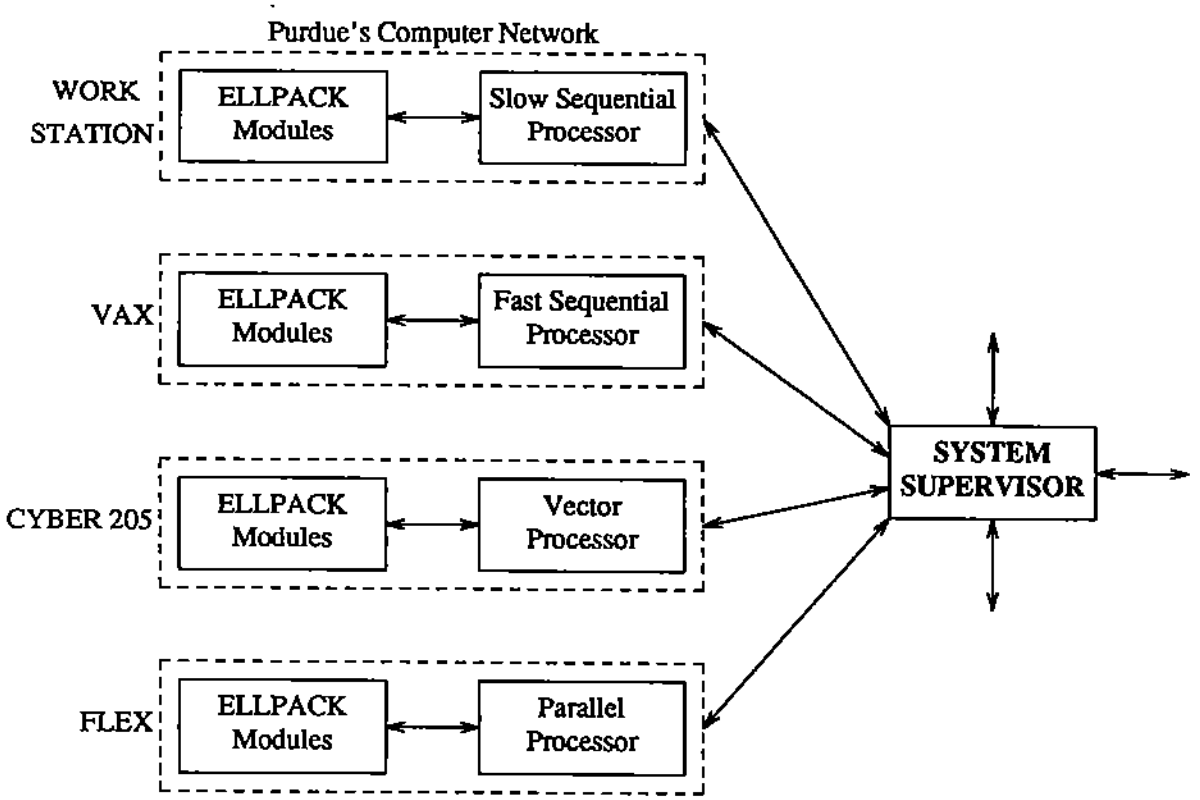


Figure III.D3 Detailed schematic of the computer network. The second expert system will analyze the computational requirements, help select appropriate machines and map the computation onto them.

and each doing $N^2/P$ of the things. Figure III.D4 shows a realistic example of an annotated graph for a DE2 application. The shaded boxes represent replicated modules that can be grouped in any convenient way.

Our operational plan is as follows. We work with relatively coarse grain structures so the graphs are of reasonable size. We translate all this information into a mathematical optimization problem; it will be mostly a linear program with perhaps a few nonlinear constraints and a

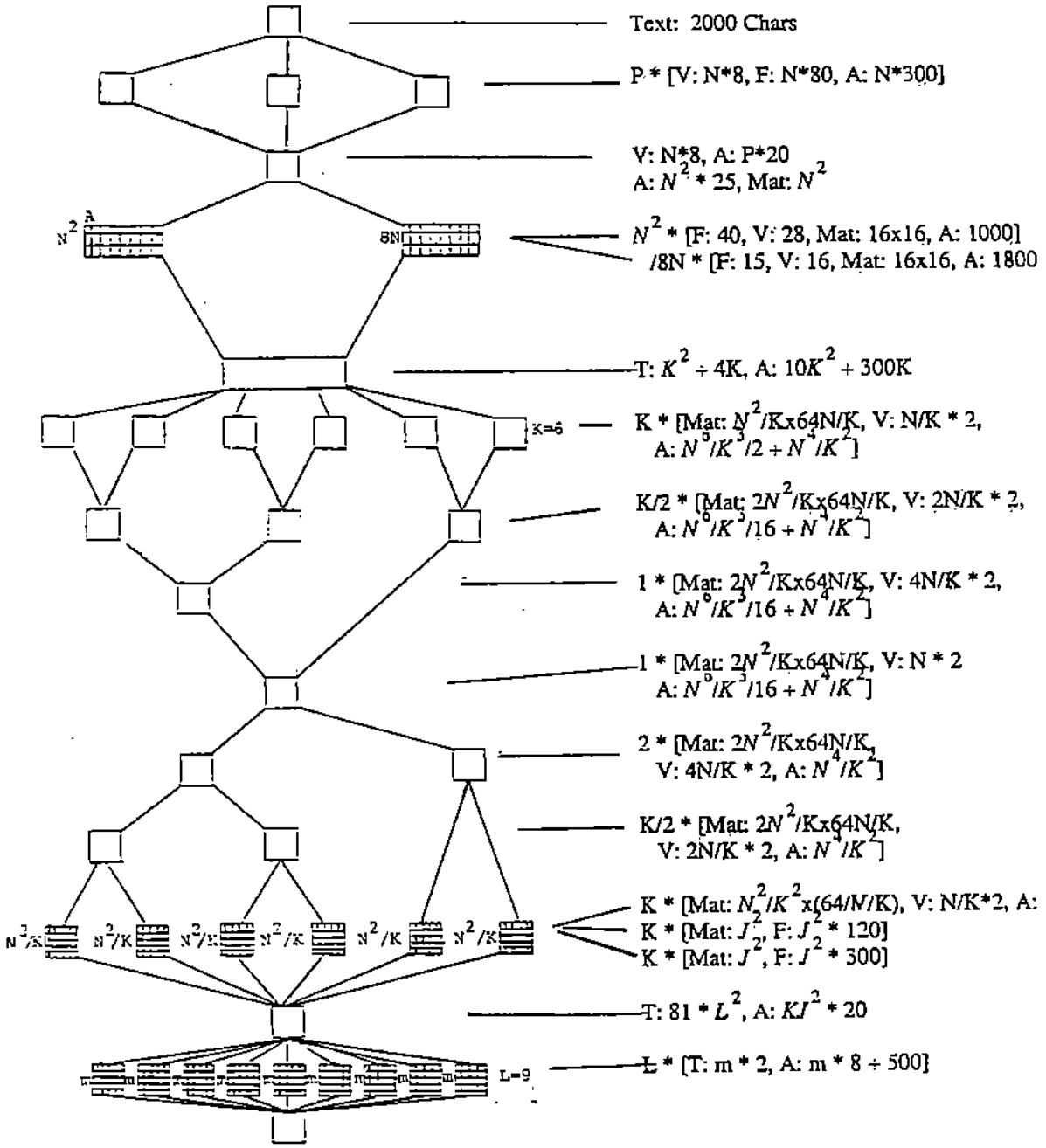


Figure III.D4 Graphical representation of a computation from the DE2 system. The annotations are encoded values of the memory and computation required at each node; another set of annotations shows the communication requirements between the nodes.

nonlinear objective function. We apply a fast heuristic algorithm to obtain a good – probably not optimal – solution of the optimization problem. This solution is then used to create the load module for the PSE run.

One can visualize from this description of DE2 that there are many specific technical problems to be solved. To keep the presentation brief, we list only eight of these.

**Technical Problems:**

1. *Obtain the module structure and resource use data.* We want more "coarse" structure and data that one obtains from a complete analysis of a program [Ri84]. We plan on software tools to help extract information and simplify programs (e.g., count the variables declared, reduce statements to arithmetic counts, ignore "small" loops or branches). It is premature to automate this completely and we also will use human interaction. Most submodules will reduce to one annotated node in the graph.
2. *Parallization of problem solving modules.* We have studied recasting methods so they are easily adapted to parallel computation. We have already parallalized some of the DE2 modules. We find that automatic approaches give helpful information but that human analysis is essential for really good results.
3. *Obtain the machine structure and resource capacities.* We will obtain this information by hand for our network and machines.
4. *Build the resource allocation problem.* A particular run combines a fair number of modules whose separate structure is known. These must be combined into one resource allocation problem. We will automate this process.
5. *Model the communications nonlinearties.* In a multiprocessor or multimachine environment the communication capacities behave nonlinearly as saturation is neared. Since this is the interesting range of operation, this nonlinearity must be modeled. We will measure this behavior ourselves for the FLEX 32 and for intermachine communication on the heterogeneous network.
6. *Solve the resource allocation problem.* The normal case will be to have only a few hundred constraints and a nonlinear objective function. The problem is too big to use time consuming standard optimization techniques, yet it is not a huge problem. We and others have developed fast heuristic methods for these problems and our initial experience is that the work to obtain a "good" solution heuristically grows linearly with the size of the problem.
7. *Architecture compiler: Dynamic software reconfiguration and program transformations.* Our long term goal is not only to map the computations onto the architecture, but to dynamically reconfigure the software to fit better on the architecture. We will operate at the procedure level (submodules).
8. *Develop performance estimators.* In selecting machines we need to be able to make a prior; estimates of routines based on the representations we have of the computations and machines [Ri82]. Performance prediction on multiprocessors is currently one of the leading open questions in high performance computations, extensive experimental work is planned to calibrate these estimators.

APPENDIX 1: **ERRATA (as of 1 March, 1986)**
SOLVING ELLIPTIC PROBLEMS USING ELLPACK

PAGE; line: CHANGE

x; 22 Transpose Rice and Ribbens as authors of Appendix C
19; 31 95 -> 103
19; 36 1074 -> 1082
36; 25 7-POINT STAR -> 7-POINT 3D
50; Fig. 3.1 Contour value are slightly changed due to new plot routines
57; 33 Need to add routines for CDXU and CDYU as ELLPACK uses the default routines of 0.0. Much better results are obtained
58; 22 With CDXU and CDYU given exactly, 1.1345798E-01 -> 8.9709E-03
59; 35 With CDXU and CDYU given exactly, 1.1345553E-01 -> 8.9113E-03
59; 5 With CDXU and CDYU given exactly, 1.1341274E-01 -> 8.8655E-03
65; -22,-18,-7 U= -> UEST=
68; 1 delete S from UNKNOWN
74; -2 1H) -> ' ')
78; 22 Until NDTYPE computation is corrected, we get a warning message here
80; -11 97 -> 105
80; -6 1249 -> 1257
84; 7 1471 -> 1487
84; 12 18882 -> 18898
86; Fig. 4.2 New plot routines produce slightly different tic-marks on axes
88; Fig. 5.1 Figure is for an 11 by 11 grid, but rest of example is 10 by 10 grid
89; -11 3 -> 4
92; 38 Should have another point (number 38) for the end of the arc
38 1.000E+00 0. E+00 1.000E+00 2 INTE 5007 0
93; Fig. 5.2 New plot routines produce different contours near arc
94; 6, 8 delete these lines
96; Fig. 5.3 Substantial changes due to new plot routines and different plot grid
96; -6, -7 Top and bottom conditions should interchanged to get plot Fig. 5.4
100; -25 I35PNU(I, -> I35PNU(1, , change I to 1
101; Fig. 5.5 see comment on page 96
101; -4 F′ ($u_0$) -> F ($u_0$)
102; 12 u + S -> u + δ
107; 12 5.C2 -> 5.C3
107; -3 Initial C for comment missing
109; 4 last term of the equation is
$3u_0(h_x u_{0_x} + h_y u_{0_y})/h$
111; Fig. 5.7 legend should have "c = 12" in italics
111; -4 x+y+t -> (x+y+t)/4.
112; 17 $L(x,y,t-\Delta t)$ -> $Lu(x,y,t-\Delta t)$
115; -23 TRUXYT -> ABS(TRUXYT)
117; 8-15 Values for MAX RELERR are wrong, magnitudes are about $10^{-3}$
125; -7, -10, -13 R1H1EV -> R3H1EV
155; 10 delete f,
187; - This module has been substantially revised. The parameters and performance paragraphs now read as follows:

PARAMETERS: PTHRS controls the stringency of the partial pivoting. It must be between 0.0 and 1.0 with a default value of 0.1. Pivoting is not done if the current diagonal entry is PTHRS (or more) times the largest element in the current row.
PERFORMANCE ESTIMATES: A workspace of length 10*I1MNCO*I1MNEQ +7*I1MNEQ+3 is required. The amount of workspace is not completely predictable and the moodule might at times stop and issue a message requesting more workspace.

195; -15 period -> periodic
203; 4 add the term "+ $u_{zz}$" to Lu
212; -11 it set -> it is set
; -9 Add the sentence: The default for NTRI is 8*NX*NY.
217; -0 Need reference
224;- Errors in page layout, indentation and margins
329; 31 Last five column headings should be
R1BPAR I1PECE I1BPTY I1BGRD I1BNGH
330; 29 Add another line
58 1.5000E+00 -1.0000E+00 2.5000E+00 4 BOTH 7004 14
345; 24,25 Add on the right: , L1REMV
; 32 IKWRK -> I1KWRK
346; 2 ACESS -> ACCESS
398; 26 I1BCST(4, 4) -> I1BCST(4, 6)
; -14 L1MEND, L1MUND -> I1MEND, I1MUND
; -1 Add on the right: , L1REMV
399; 1 Add on the right: , L1REMV
; 2 L1PAGE, L1INPT, L1OUTP, L1SCRA -> I1PAGE, I1INPT, I1OUTP, I1SCRA
; 6 C1RTABL -> C1TABL
; 30 31 -> 28
; 35 3 -> 2
; 36 Add a new first argument to Q0INT of .FALSE.
; 45 Q8GP00 -> Q0GP00
400; 25 After statement 162 add CALL Q1PCOE(0.0, 0.0, RICUXX)
; 46 After statement 183 add the same as above
; -1 Q8GP99 -> Q0GP99
401; -5 IBOV2 -> 0.0
403; 12,13 Add: , L1REMV
421;- Column "Boundary Conditions" has misalignments for 42P and 48P
The entry for 50P should be D
438; 3 6 - 0.7 -> y - 0.7
443; 17 subscript N should be n for Problem 42 boundary condition
447;- Problem 50 has an extraneous blank square
449; -1 None -> $\alpha$ varies the strength of the ridge in the solution among other things
450; 5 = g -> = G and = h -> = H
455; 21 Add the following to the second paragraph: Compiling the driver and these routines, then linking with the PGLIB (see Section B.6) yields an executable preprocessor
456; 9 Add the sentence (after "blanks.") In addition to rules and action, PG programs may include special segments
456; 10 built-in rule -> segment
456; 19 This line should appear as
PROG -> 'OPTION' ALFNUM+ '=' ALFNUM+

PROG -> 'OPTION' ALFNUM+ '=' ALFNUM+

456; -10, -11 Insert the term 'OPTION' (in quotes) after the ->

457; -16 the same as -> is the same as

460; 5 fules -> rules

463; 8 The DONE procedure has not been implemented

465; 7 Add the sentences (after "IQOUTF.") PG uses template processor routines to keep track of template variables. The template processor may also be called as a subroutine to process a template. The ELLPACK preprocessor does this twice; once to initialize many template variables and one to generate the Fortran control program.

491; -15 Fishpack -> Fishpak

491; -14 FISHPACK -> FISHPAK

## APPENDIX 2: ELLPACK LITERATURE

This does not include publications primarily about the numerical methods used in ELLPACK modules.

### BOOK

Solving Elliptic Problems Using ELLPACK
John R. Rice and Ronald F. Boisvert
Springer Verlag (1985), 497 pages

### PAPERS (in chronological order)

Houstis, E.N., Lynch, R. E., Papatheodorou, T.S., and Rice, J.R., Development, evaluation and selection of methods for elliptic partial differential equations. Ann. AICA, *11* (1975), 98-105.

Rice, J.R., ELLPACK: A research tool for elliptic partial differential equations software. In *Mathematical Software III*, (J. Rice, ed.), Academic Press, (1977), 319-341.

Rice, J.R., ELLPACK: A cooperative effort for the study of numerical methods for elliptic partial differential equations. In ARO Report 77-3: *Proceedings of the 1977 Army Numerical Analysis and Computer Conference*, pp. 165-169.

Houstis, E.N., Lynch, R.E., and Rice, J.R., Evaluation of numerical methods for elliptic partial differential equations. J. Comp. Physics, *27* (1978), 323-350.

Lynch, R.E.,and Rice, J.R., The HODIE method and its performance. In *Recent Advances in Numerical Analysis* (C. de Boor, ed.), Academic Press, (1978), 143-175.

Boisvert, R.F., Houstis, E.N., and Rice, J.R., A system for performance evaluation of partial differential equations software. IEEE Trans. Software Engineering, *5* (1979), pp. 418-425.

Rice, J.R., Methodology for the algorithm selection problem. In *Performance Evaluation of Numerical Software* (L. Fosdick, ed.), North- Holland (1979), pp. 301-307.

Rice, J.R., Programming effort analysis of the ELLPACK language, CSD-TR 288 (3 pages). SIGNUM Newsletter, *14* (1979), 109-111. Abstract in *Advances in Computer Methods for Partial Differential Equations*, III (Vishnevetsky and Stepleman, eds) IMACS, Rutgers University (1979), 28.

Gherson, P., Lykondis, P.S., and Lynch, R.E., Analytic study of end effects in liquid metal MHD generators. In *Seventh Inter. Conf. MHD Electrical Power Generation*, M.I.T. (1980), 590-594.

Houstis, E.N. and Rice, J.R., An experimental design for the computational evaluation of elliptic partial differential equation solvers. In *Production and Assessment of Numerical Software* (M. Hennell and L. Delves, eds.) Academic Press (1980), 55-65.

Coriell, S.R., Boisvert, R.F., Rehm, R. and Sekeuka, R., Solute segregations during unidirectional solidification of a binary alloy with curved solid-liquid interface II-large departure from planirity, J. Crystal Growth, *5* (1981), 167-175.

Rice, J.R., On the effectiveness of iteration for the Galerkin method equations. In *Advances in Computer Methods for Partial Differential Equations IV* (R. Vishnevetsky, ed.) (1981) IMACS, New Brunswick (1981), 68-73.

Rice, J.R., ELLPACK, progress and plans. In *Elliptic Problem Solvers* (M. Schultz, ed.) (1981) Academic Press (1981), 135-162.

Rice, J.R., Houstis, E.N. and Dyksen, W.R., A population of linear, second order, elliptic partial differential equations on rectangular domains, Part 1 and 2. Math Comp. *36*, (1981), 475-484 and microfiche supplement.

Houstis, E.N. and Rice, J.R., High order methods for elliptic partial differential equations with singularities. Int. J. Numer. Meth. Engr., *18* (1982), 737-754.

Rice, J. R., Machine and compiler effects on the performance of elliptic PDE software. Proc. 10th World Congress, IMACS, *1* (1982), 446-448.

Rice, J.R., Performance analysis of 13 methods to solve the Galerkin method equations, J. Lin. Alg. Appl. *52/53* (1983), 533-546.

Dyksen, W.R., Houstis, E.N., Lynch, R.E. and Rice, J.R., The performance of the collocation and Galerkin methods with Hermite bi-cubics. SIAM J. Numer. Anal., *21* (1984), 695-715.

Houstis, E.N. and Rice, J.R., Vector ELLPACK: Domain mappings and parallel geometric discretizations. In *Advances in Computer Methods for Partial Differential Equations V* (Vishnevetsky and Stepleman, eds.), IMACS, Rutgers University (1984), 195-198.

Ribbens, C., Rice, J.R. and Ward, W.A., Algorithm 622: A simple macro processor. ACM Trans. Math. Software, *10* (1984), 410-416.

Rice, J.R., Building elliptic problem solvers with ELLPACK. In *Elliptic Problem Solvers II* (G. Birkhoff and A. Schoenstadt, eds.) Academic Press (1984), 3-22.

Rice, J.R., Numerical computation with general two dimensional domains. ACM Trans. Math. Software, *10* (1984), 443-452.

Rice, J.R., Algorithm 624: A two dimensional domain processor. ACM Trans. Math. Software, *10* (1984), 453-562.

Rice, J.R., Software parts for elliptic PDE software. In *PDE Software: Modules, Interfaces and Systems* (B. Enquist and T. Smedaas, eds.), North-Holland (1984), 123-134.

Rice, J.R., ELLPACK: An evolving problem solving environment. In *Problem Solving Environments for Scientific Computing* (B. Ford, ed.) North-Holland (1986).

**TECHNICAL REPORTS (not preprints of papers)**

Rice, J.R., ELLPACK - A Cooperative Effort for the Study of Numerical Methods for Elliptic Partial Differential Equations, CSD-TR 203, October 15, 1976 (5 pages).

Rice, J.R., ELLPACK Contributor's Guide, CSD-TR 208, November 1, 1976 (Revised September 16, 1977) (45 pages).

Rice, J.R., ELLPACK 77 User's Guide, CSD-TR 226, March 18, 1977 (Revised January 20, 1978, corrected March 27, 1978) (34 pages).

Boisvert, R.F., ELLPACK Distribution Guide, CSD-TR 254, December 7, 1977 (Revised September 15, 1978) (22 pages).

Rice, J.R., ELLPACK 77 Contributor's Guide, CSD-TR 267, June 10, 1978 (34 pages).

Gherson, P., Lykoudis, P.S. and Lynch, R.E., The Use of ELLPACK 77 for Solving the Laplace Equation on a Region with Interior Slits, Application to a Problem in Magnetohydrodynamics, CSD-TR 275, July 1978.

Rice, J.R., ELLPACK Workshop, CSD-TR 285, August 1978 (14 pages).

Boisvert, R.F. and Bonnet, J.M., The Data Management Subsystem of the System for Performance Evaluation of PDE Software, CSD-TR 286, August 30, 1978 (22 pages).

Rice, J. R., ELLPACK 77 User's Guide, CSD-TR 289, September 15, 1978 (67 pages), Revised July 1, 1980 (75 pages).

Rice, J.R., ELLPACK 78 User's Guide - Preliminary version, CSD-TR 306, May 9, 1979 (18 pages).

Rice, J.R. and Sewell, Granville, On the effectiveness of iteration for the Galerkin method equations, CSD-TR 307, June 1, 1979 (10 pages).

Boisvert, R.F., Rice, J.R., and Warner, J., ELLPACK network documentation-preliminary version, CSD-TR 308, July 15, 1979.

Boisvert, R.F., ELLPACK Control Card Procedures: XEQ ELLPACK, XEQ GETELL, CSD-TR 310, August 27, 1979 (15 pages).

Rice, J.R. 1979 ELLPACK Workshop, Progress report and proposal for a 2-year program, CSD-TR 315, August 15, 1979 (Revised, November 15, 1979) (16 pages).

Boisvert, R.F., Brophy, J., Rice, J.R., and Warner, J., ELLPACK Network Users Guide, CSD-TR 319, December 1979 (35 pages).

Boisvert, R., Recommendation for ELLPACK module standards, NBS Scientific Computing Division Report, March 26, 1980 (14 pages).

Rice, J. R., ELLPACK User's Guide Supplement, CSD-TR 363, May 1, 1981 (11 pages).

Rice, J.R. and Ward, W.A., A Simple Macro Processor User's Guide. CSD-TR 403, May 20, 1982 (21 pages).

Rice, J.R., Expansion of the performance evaluation capabilities of ELLPACK, CSD-TR 451, September 4, 1983 (8 pages).

Brophy, J.F., PG-A Preprocessor Generator. CSD-TR 472, 1984 (13 pages).

Bonomo, J.P., Dyksen, W.R., and Rice, J.R., The ELLPACK performance evaluation system. CSD-TR 569, 1986 (23 pages).

Rice, J.R., Dyksen, W.R., Houstis, E.N. and Ribbens, C.J., ELLPACK status report. CSD-TR 579, 1986 (31 pages).

## APPENDIX 3: SPLINE COLLOCATION

Preliminary Draft

MODULE: SPLCOL (rectangular domains)

AUTHOR(S): E. N. Houstis and M. T. Vavalis

PURPOSE: Discretize the linear elliptic partial differential equation

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

on a two-dimensional rectangular region $R$ subject to homogeneous Dirichlet or Neumann boundary conditions on the boundary grid points of $\partial R$. The coefficients $a, b, \cdots, g$ are functions of $x$ and $y$.

METHOD: For uniform meshes a high order residual spline collocation method is applied to obtain optimal spline approximations to $u$ and its derivatives. The method is currently applicable for splines of order 3, 4, 6, and 8. In the case of order 3 splines superconvergence of order $O(h^{4-j})$ occurs at nodes $(j = 0)$, midpoints $(j = 0, 2)$ and points $(x_k + \lambda h, y_l + \lambda h)$, $(j = 1)$ where $\lambda = (3 \pm \sqrt{3})/6$.

PARAMETER: IORDER = 3, 4, 6, 8 (order of splines)

RESTRICTIONS:

2-dimensional rectangular regions
Grid size at least $3 \times 3$
$a$ and $c$ must be nonzero

APPENDIX 4: **HODIE (for general domains)**

**MODULE:** HODIE (general domain)

**AUTHOR:** Robert E. Lynch

**PURPOSE:** Discretize the linear elliptic partial differential equation

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

on a general two-dimensional region $R$ subject to conditions

$$Mu = pu + qu_x + ru_y = t$$

on the boundary $\partial R$ of $R$, where $a, b, \cdots, g, p, \cdots, t$ are functions of $x$ and $y$. This is one of three implementations of HODIE, the other two are (1) 2-dimensions, rectangular domain with Dirichlet boundary conditions and (2) 3-dimensions, box domain, Dirichlet conditions. The ELLPACK preprosessor selects the implementation appropriate for the user's problem.

**METHOD:** See the write up of the implementations (1) and (2).

**PARAMETERS:** IORDER = 2, 4 or 6 for the order of accuracy; default 4.

**RESTRICTIONS:**

2-dimensions
Grid of size at least 3×3
$a$ and $c$ must be nonzero
$p^2 + q^2 + r^2$ must be positive