1986

Interactive ELLPACK

Wayne R. Dyksen

Calvin J. Ribbens

Report Number:

86-588

INTERACTIVE ELLPACK:

An Interactive Problem Solving Environment for Elliptic Partial Differential Equations

Wayne R. Dyksen
Calvin J. Ribbens

CSD-TR 588
April, 1986

Abstract

ELLPACK is a versatile very high level language for solving elliptic partial differential equations. Solving elliptic problems with ELLPACK typically involves a process in which one repeatedly computes a solution, analyzes the results, and modifies the solution technique. Although this process is best suited for an interactive environment, ELLPACK itself is batch oriented. With this in mind, we have developed *Interactive ELLPACK*, an extension of ELLPACK which provides true interactive elliptic problem solving by allowing the user to interactively build grids, choose solution methods, and analyze computed results. Interactive ELLPACK features a sophisticated interface with windowing, color graphics output, and graphics input.

INTERACTIVE ELLPACK:

An Interactive Problem Solving Environment for Elliptic Partial Differential Equations

Wayne R. Dyksen
Calvin J. Ribbens

1. Introduction

The size and complexity of feasible scientific computations has increased dramatically in the last twenty-five years as a result of both technological and algorithmic progress. 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.

A prime example of such a system is the ELLPACK system [Rice & Boisvert, 1985]. ELLPACK is a versatile *very high level language* (VHLL) for solving elliptic partial differential equations (PDEs). It serves as a testbed for elliptic algorithms as well as a model VHLL for other problem domains (e.g., [Dyksen, et al., 1984]).

Solving PDEs with ELLPACK often involves an iterative process. Typically one begins by computing an initial solution using a somewhat arbitrary grid/method combination. This solution is analyzed along with other related functions such as the residual or an estimate of the error. This analysis involves many techniques such as viewing graphic representations of the functions, computing maximum or minimum values, or considering function values at specific points. A new grid may be constructed by moving existing grid lines, and adding or deleting grid lines; grids are constructed both computationally and visually. Parameters in the PDE or the numerical method may be adjusted, or a totally different method may be chosen. A new solution is then computed using the new grid/method combination. This process continues until the user is confident that a satisfactory solution has been found.

When faced with the above scenario, a user in a batch-like environment looses time and, *a fortiori*, his train of thought. With this in mind, we have developed *Interactive ELLPACK*, an extension of ELLPACK which provides true interactive elliptic problem solving by allowing the user to interactively build grids, choose solution methods, and analyze computed results.

A brief overview of ELLPACK is given in Section 2. We describe Interactive ELLPACK and its implementation in Sections 3 and 4, respectively. Section 5 contains a summary along with future directions for ELLPACK.

2. ELLPACK

The objective of ELLPACK was to develop an environment for evaluating the performance of algorithms and software for elliptic PDEs. Three major results of this effort are:

1. ELLPACK
   A very high level language for solving elliptic problems. [Rice & Boisvert, 1985].
2. Elliptic PDE Population
   A population of 56+ parameterized elliptic problems (190+ instances) used by the Performance Evaluation System. [Rice, Houstis & Dyksen, 1981].
3. Performance Evaluation System
   A system for the generation, collection and analysis of data on the performance of elliptic algorithms. [Boisvert, Houstis & Rice, 1979], [Bonomo, Dyksen, & Rice, 1986].

ELLPACK can be used to solve a large class of elliptic problems: second order, linear elliptic PDEs in two and three dimensions with Dirichlet, Neuman, mixed or periodic boundary conditions. For example, the simple elliptic problem

$$
\begin{aligned}
-\nabla^2 u - 20\pi^2 u &= 0 & (x,y) \in (0,1)\times(0,1) \\
u &= 0 & x=0,1, \quad y=0 \\
u_y &= 4\pi\sin(2\pi x) & y=1
\end{aligned}
$$

can be solved by the ELLPACK program shown in Figure 2.1.

```
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

grid.            17 x points $ 17 y points
discretization. 5 point star
indexing.        as is
solution.        linpack band
output.          max(u) $ plot(u) $ max(residual) $ plot(residual)
end.
```

Figure 2.1. Sample ELLPACK program.

An ELLPACK program consists of several *segments*. The elliptic problem is defined by the *equation* and *boundary* segments; these segments are declarations to ELLPACK and as such are not "executed". The remaining segments are executed from top to bottom. Figure 2.2 shows further examples of equation and boundary segments.

Sample Equations Segments

```
uxx + uyy + (1.+sin(pi*x))ux - u = f(x,y)

(p(x,y)ux)x + (p(x,y)uy)y - q(x,y)u = f(x,y)

uxx + uyy + uzz = f(x,y,z)
```

Sample Rectangular Boundary Segment

```
periodic          on x = 0
                  on x = 1
u = 0             on y = 0
uy + 2u = g(x)    on y = 1
```

Sample Nonrectangular Boundary Segment

```
u = 0                       on line 0.2, 0.0
                                 to 0.0, 0.4
                                 to 0.0, 0.6
                                 to 0.2, 1.0

uy = 0                      on line 0.2, 1.0
                                 to 0.6, 1.0

a(x,y)ux + b(x,y)uy = 0 on x=1+0.4*cos(t), &
                           y=1+0.4*sin(t)  &
                           for t = pi to 3*pi/2

ux = 0                      on line 1.0, 0.6
                                 to 1.0, 0.4

a(x,y)ux + b(x,y)uy = 0 on x=1+0.4*cos(t), &
                           y=  0.4*sin(t)  &
                           for t = pi/2 to pi

uy = 0                      on line 0.6, 0.0
                                 to 0.2, 0.0
```

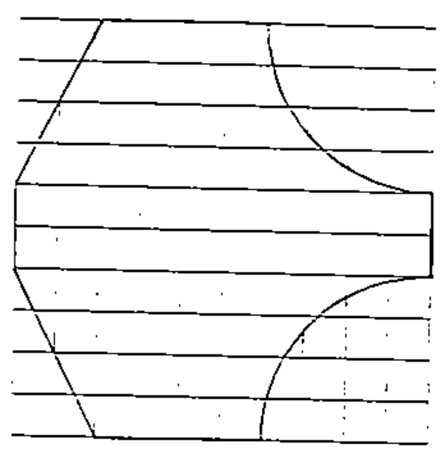

Figure 2.2. Examples of ELLPACK equation and boundary segments used to describe second order, linear elliptic PDEs in two and three dimensions with Dirichlet, Neuman, mixed or periodic boundary conditions.

ELLPACK contains four basic types of problem solving *modules*. *Discretization modules* discretize the continuous problem by generating a system of linear equations. *Indexing modules* are used to order the linear system which is then solved by a *solution module*. *Triple modules* incorporate all three of the above steps into one module. A summary of ELLPACK facilities is given in Appendix A.

Although ELLPACK was designed for second order, linear elliptic problems, its "software parts" design allows it to be used to solve problems from other domains. For example, we have used ELLPACK to solve coupled systems of elliptic equations, nonlinear elliptic problems, time dependent problems, and three dimensional problems on cylindrical domains with holes.

3. Interactive ELLPACK

3.1 Overview. Interactive ELLPACK is an extension of ELLPACK. Menus of traditional ELLPACK statements can be constructed using the newly added *menu* segment. Interactive ELLPACK uses both color graphics output and graphics input to provide a sophisticated user interface.

3.2 The Menu Segment. The segments in a traditional ELLPACK program are executed sequentially, from top to bottom. In order to allow the user to specify several different methods or procedures that might be used in solving a problem, and to interactively choose from them at run time, we added a new *menu* segment to ELLPACK. 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 definition>
```

where <key> is an optional key (the user enters this to select the item at run time), and <label> is an optional name for this item. The default <key> is an integer such that the items in each menu are numbered sequentially. The default <label> is a meaningful string from <item definition>. The <item definition> may include one or more of the following ELLPACK segments: *grid, discretization, indexing, solution, triple, output, procedure,* or *fortran*. An <item definition> may extend over several lines, as long as segment names within menus do not appear in column one. Since segment names which appear outside of a *menu* segment must begin in column one, the assumption is that a *menu* segment continues until another segment name occurs beginning in column one. Every menu automatically contains three standard items: *continue* to the next menu, *return* to the previous menu, and *quit* from the Interactive ELLPACK session. In a UNIX environment, the user can escape to the shell by "!command". Three examples which illustrate most of the features of the menu segment are given in Figure 3.1.

In a windowing environment, <menu name> is used as the window title, and the <key>'s are used to construct pop-up menus; in this case, the default keys 1, 2,... should be avoided and meaningful ones supplied.

3.3 Interactive Graphics Output. ELLPACK contains a number of *output* modules which produce graphics output. If *function* is a FORTRAN function, then *plot(function)* produces a two dimensional contour plot (level curves), and *plot3d(function)* gives a three dimensional rendering of *function.* A plot of the domain (perhaps non-rectangular) along with the current grid can be obtained by *plot domain.* In standard ELLPACK a small set of plotting primitives are called to produce the plots. These routines are system dependent, but examples using well-known plotting packages such as CALCOMP or DISSPLA are provided. Most installations then provide some way of sending these plot files to a number of output devices.

Interactive ELLPACK gives the user the ability to interactively view and manipulate multiple ELLPACK plots. The interface depends on the terminal type, which is specified using the Interactive ELLPACK option *terminal*; currently, terminal can be any one of *dumb*, *tek4105*, *tek4107*, *tek4115*, or *ridge.* If the terminal type is *dumb*, graphics output is merely written to a file in the standard ELLPACK way. On a graphics terminal, the interface consists of either a fixed number of static views or a variable number of dynamic views (i.e., windows), each of which is identified by number.

On the Tektronix terminals, color graphics output is written to predefined graphics views. The number of colors and views is terminal specific. Figure 3.2 shows a complete Interactive ELLPACK program; Figure 3.3 illustrates the Interactive ELLPACK interface on a Tektronix 4115. Each view is a rectangular area typically occupying some subregion of the terminal's graphic surface. Each graphic view is logically equivalent to a copy of the entire graphics surface; hence, the high level software does not need to know about the size or location of individual views. The terminals themselves translate and scale the graphics output to fit in a given view. One view is chosen as the default output view. Graphics output is stored in so-called segments which are stored in memory local to the

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


 ****************************************
 *                                      *
 *   discretization                     *
 *                                      *
 ****************************************

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


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


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

 1 :  bandge
 2 :  linpackspdband
 3 :  sor
 c :  continue
 r :  return
 q :  quit


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


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

 1 :  maximums
 2 :  table u
 c :  continue
 r :  return
 q :  quit
```

Figure 3.1. Examples of Interactive ELLPACK menu segments. Each menu segment is followed by the corresponding menu that would be produced at run time in a non-windowing environment.

```
*********************************************************************
*                                                                   *
*       I N T E R A C T I V E   E L L P A C K   P R O G R A M       *
*                                                                   *
*********************************************************************
options.
                max x points = 33 $ max y points = 33
                interpolation = splines
                terminal = tek4115

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

menu.   'Solution Menu'
        'ig : interactive grid'                 grid.  interactive
        '5p : ordinary finite diffs'            disc.  5 point star
                                                solu.  band ge
        'co : hermite collocation'              disc.  hermite collocation
                                                solu.  band ge
        'hh : high order diffs/linpack band'    disc.  hodie helmholtz
                                                solu.  linpack band
        'hf : high order diffs/fft'             trip.  hodie fft

menu.   'Output Menu'
        'pt  : plot true'                       out.   plot(true)
        'pt3 : plot true 3d'                    out.   plot3d(true)
        'pu  : plot u'                          out.   plot(u)
        'pu3 : plot u 3d'                       out.   plot3d(u)
        'pa  : plot abserr = abs(error)'        out.   plot(abserr)
        'pa3 : plot abserr 3d'                  out.   plot3d(abserr)
        'pr  : plot residual'                   out.   plot(residu)
        'mv  : move plot from view to view'     out.   move view
        'cv  : copy plot from view to view'     out.   copy view
        'dv  : delete plot from view'           out.   delete view
        'ev  : enlarge views'                   out.   enlarge views
        'pp  : put function plot to   a file'   out.   put plot
        'gp  : get function plot from a file'   out.   get plot
        'pg  : plot grid'                       out.   plot grid
        'og  : overlay grid'                    out.   overlay grid
        'mx  : max error'                       out.   max(error)

subprograms.
      function true(x,y)
      common / clrvgl /  rlepsg, rlepsm, pi
      true = sin(2*pi*x) * sin(4*pi*y)
      return
      end
      function abserr(x,y)
      abserr = abs(error(x,y))
      return
      end
end.
```

Figure 3.2. Interactive ELLPACK program. Two menus are specified in this example: one with a set of problem solving modules (*grid, discretization, solution, triple*), and one with a choice of *output* modules.

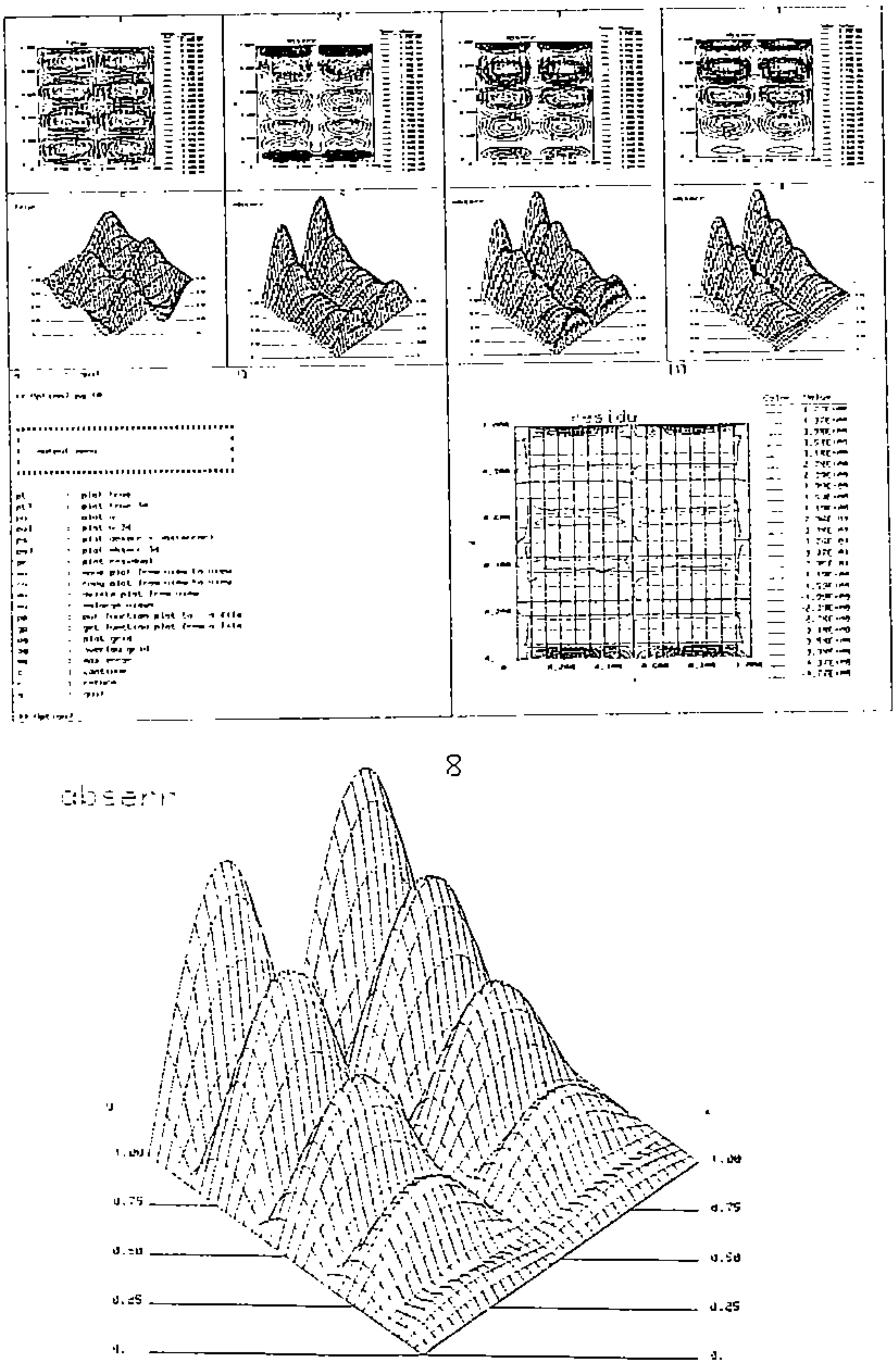

Figure 3.3. Interactive ELLPACK session on a Tektronix 4115 (top) and an enlargement of view 8 (bottom). This graphics interface includes ten fixed views. The color contours and surfaces are shown here in black. The screen resolution is good enough for all the numbers to be read easily.

terminal. These segments can be manipulated locally (i.e., with only a few bytes of communication from the host). Graphic output can be saved in a file to be viewed during subsequent sessions, or to be printed on a color printer. These terminals support a separate surface for dialog (non-graphics) output. This dialog area covers (transparently or opaquely) some number of views. It can be made invisible (and visible) from the keyboard so that all graphics views may be exposed.

On the Ridge display (a bit-mapped device), graphics output is displayed in windows which can be manipulated with a mouse. Figure 3.4 contains examples of an Interactive ELLPACK session on a Ridge display. One window is used for dialog; its shape and size change depending on the active menu. Menu items are selected from pop-up menus with the mouse. Graphics output is usually placed into a newly opened window, although it can be directed to any existing window. Once opened, the windows are managed by the Ridge window manager

A number of new output modules have been added to ELLPACK to implement the interactive graphics interface of Interactive ELLPACK. For example, *move view* moves the contents of one view to another. If a menu contains the item

```
'mv : move view'     output. move view
```

then after entering "mv", the user will be prompted for "From view?" and "To view?". Recall that views are identified by number. Parameters for these interactive output modules can also be specified directly on the command line as in "mv 3 7". A complete list of these modules is given in Table 3.1.

Table 3.1

Interactive ELLPACK output modules for manipulating the contents of views

Module	Parameters	Effect
move view	*view1 view2*	Moves the contents of *view1* to *view2*.
copy view	*view1 view2*	Copies the contents of *view1* to *view2*.
select view	*view*	Selects *view* as the active view.
delete view	*view*	Deletes the contents of *view*.
enlarge view	*view*	Enlarges the contents of *view* so that it fills the screen.
plot grid	*view*	Plots the current grid in *view*. The plot is not part of the retained segment in the view.
overlay grid	*view*	Plots the current grid in *view*. The plot is added to the retained segment in the view.
put plot	*view file*	Puts the plot displayed in *view* into *file*.
get plot	*file view*	Gets a plot from *file* and displays it in *view*.

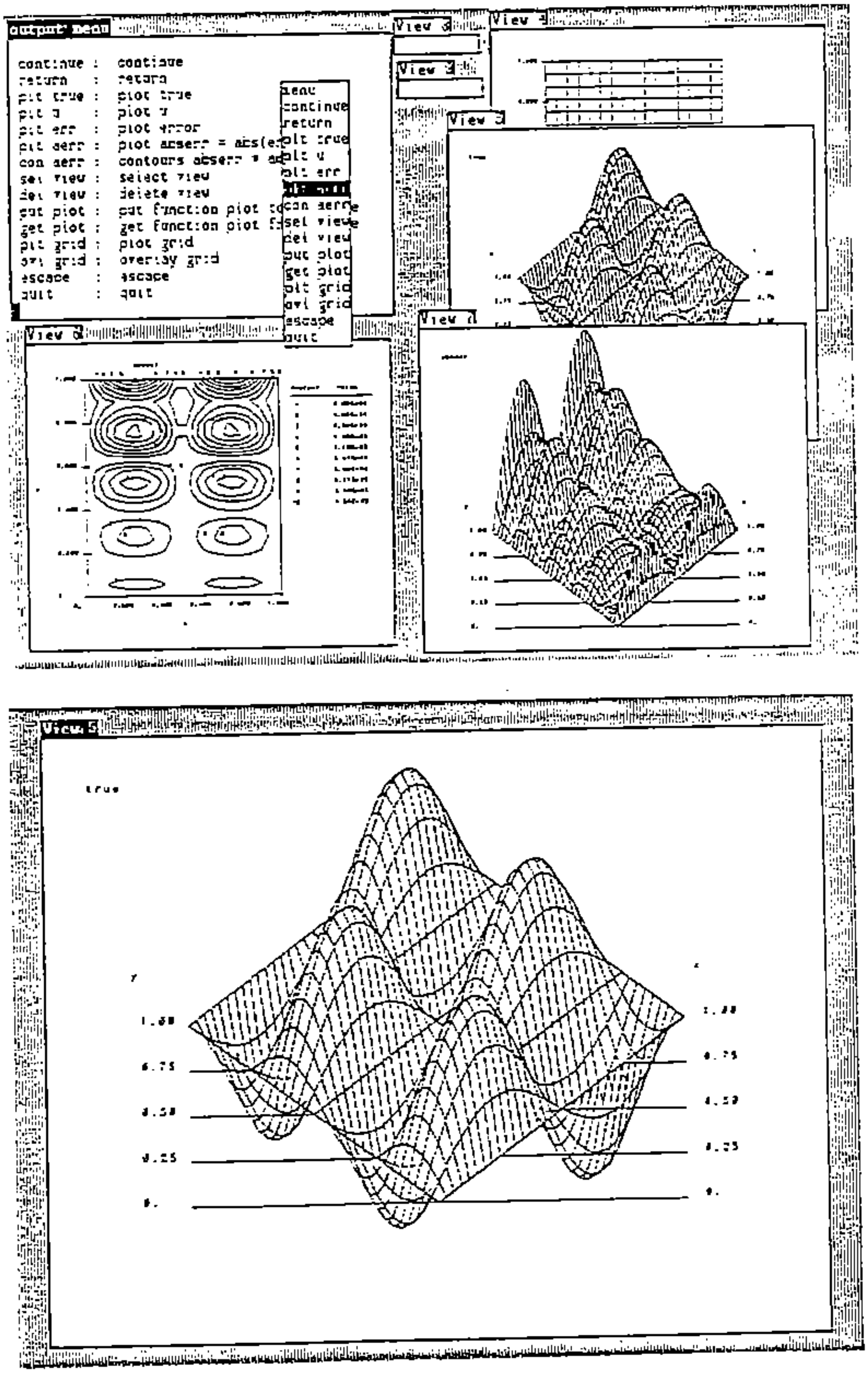

Figure 3.4. Interactive ELLPACK session on a Ridge display which supports windowing (top); individual views may be enlarged (bottom).

3.4 Interactive Graphics Input. The ELLPACK *grid* segment allows the user to specify either a uniform grid as in

```
grid.     5 x points
          5 y points
```

or a nonuniform grid as in

```
grid.     5 x points  0.0, 0.2, 0.5, 0.8, 1.0
          5 y points  0.0, 0.2, 0.5, 0.8, 1.0
```

In the nonuniform case, it is often desirable to place the grid lines with respect to some known function such as the right side of the PDE, a residual, a trial solution or an estimate of the error. This is done both computationally and visually.

Interactive ELLPACK contains a new grid segment *interactive* which allows the user to interactively construct a grid. In a non-graphic environment, the user is simply prompted for the appropriate grid information. On graphics terminals, the grid is constructed via a graphics input device. The interactive grid module displays the domain with the current grid in the default graphics view, and prints the following menu:

```
*************************************
*                                   *
*  Interactive Grid Module Commands *
*                                   *
*************************************

c : clear the grid and the screen
E : enlarge view
e : undo enlarge view
g : get a grid from a file
h : help
i : input a value for a grid line
m : make the grid uniform in x or y
n : print the number of grid lines
o : restore original grid
p : put a grid into a file
q : quit
r : redraw the screen
U : user defined via usrgrd
u : uniform grid in x or y
v : print the value of the nearest grid lines
x : add    an x grid line
X : delete an x grid line
y : add    a  y grid line
Y : delete a  y grid line
```

Vertical grid lines can be added and deleted by typing "x" and "X", respectively. The location of the grid line to be added or deleted is specified by a cross hair cursor which is positioned using either a joystick, thumbs wheels, or a mouse depending on the type of terminal. Typically, the grid is constructed over top of a plot of some function of interest. If a command requires input, the user is prompted for it in the dialog area. The command menu can be displayed by typing "h".

Interactive ELLPACK has proven to be a useful research tool. Interactive grid generation has been used extensively in several studies: the effect of grid aspect ratio on finite element methods [Rice, 1985]; adaptive tensor product grids for singular problems [Rice, 1986]; domain mappings and problem transformations [Ribbens, 1986]. A task for which Interactive ELLPACK is particularly well suited is that of parameter studies. For example, suppose one wanted to study the effects of introducing derivative boundary conditions on various PDE solving methods. To proceed, one might solve a family of problems with boundary conditions of the form

$$\alpha u + \beta u_N = g$$

for varied α and β. An Interactive ELLPACK program to accomplish this is given in Figure 3.5; note that values for α and β are entered interactively by selecting the "sp" option of the "Solution Menu".

4. Implementation of Interactive ELLPACK

4.1 Overview. ELLPACK can be viewed as a very high level interface to a library of scientific subroutines. The ELLPACK *preprocessor* reads an ELLPACK source program and generates a FORTRAN control program containing appropriate variable declarations and calls to ELLPACK modules. This control program is compiled and linked with the precompiled libraries to produce an executable program. An overview of the ELLPACK system is given in Figure 4.1. The development of Interactive ELLPACK required extending the ELLPACK preprocessor and the graphics interface.

4.2 The ELLPACK Preprocessor. The ELLPACK preprocessor was extended to recognize the new *menu* segment and a number of new modules, including the interactive *grid* module and various graphics *output* modules. Modifications such as these are relatively easy to make because of the design of the preprocessor. The language recognized by the preprocessor is actually defined by a *PG* program or *grammar*. PG is a FORTRAN-based preprocessor generator system [Rice & Boisvert,1985,Appendix B]. It takes as input a grammar consisting of *rules* and *actions* and generates as output a program which parses the language defined by the rules (in this case ELLPACK) and performs the associated actions (in this case writing the appropriate FORTRAN control program). By using PG we can generate a very complicated program (the ELLPACK preprocessor) consisting of over 13000 lines of FORTRAN by writing a comparatively simple 1200 line grammar. In order to extend the ELLPACK language to

```
*************************************************************
*                                                           *
*   M I X E D   B O U N D A R Y   C O N D I T I O N   S T U D Y   *
*                                                           *
*************************************************************

options.
               max x points = 33 $ max y points = 33
               interpolation = splines
               terminal = tek4115

global.
      common / alpbet / alpha, beta

equation.      uxx + uyy = f(x,y)

boundary.      u = true(x,y)                on x = 0
                                            on x = 1
               alpha*u + beta*uy = g(x,y)  on y = 0
                                            on y = 1

menu.   'Solution Menu'
        'sp : set parameters'             fort.
                                                print *, "alpha, beta?"
                                                read *, alpha, beta
        'ig : interactive grid'           grid. interactive
        'fi : finite diffs/linpack band'  disc. 5 point star
                                          solu. linpack band
        'hi : high order diffs/linpack band' disc. hodie helmholtz
                                          solu. linpack band
        'co : collocation'                disc. hermite collocation
                                          solu. band ge
menu.   'Output Menu
        'mx : max'                        out.  max(error)
        'pt : plot true'                  out.  plot(true)
        'pu : plot u'                     out.  plot(u)
        'pa : plot error'                 out.  plot(error)
        'pr : plot residual'              out.  plot(residu)
        'mv : move plot from view to view'  out.  move view
        'cv : copy plot from view to view'  out.  copy view
        'dv : delete plot from view'      out.  delete view
        'ev : enlarge views'              out.  enlarge views
        'pp : put function plot to   a file' out.  put plot
        'gp : get function plot from a file' out.  get plot
        'pg : plot grid'                  out.  plot grid
        'og : overlay grid'               out.  overlay grid

subprograms.
      function true(x,y)
      true = (x**4 + 0.5) * (2.*y**4 - 0.5)
      return
      end
      function f(x,y)
      f = 12.*x**2 * (2.*y**4 - 0.5) + (x**4 + 0.5) * 24. * y**2
      return
      end
      function g(x,y)
      common / alpbet / alpha, beta
      g = alpha*true(x,y) + beta*((x**4 + 0.5) * 8.*y**3)
      return
      end
end.
```

Figure 3.5. An Interactive ELLPACK program to study the effects of introducing derivative boundary conditions on various PDE solving methods.

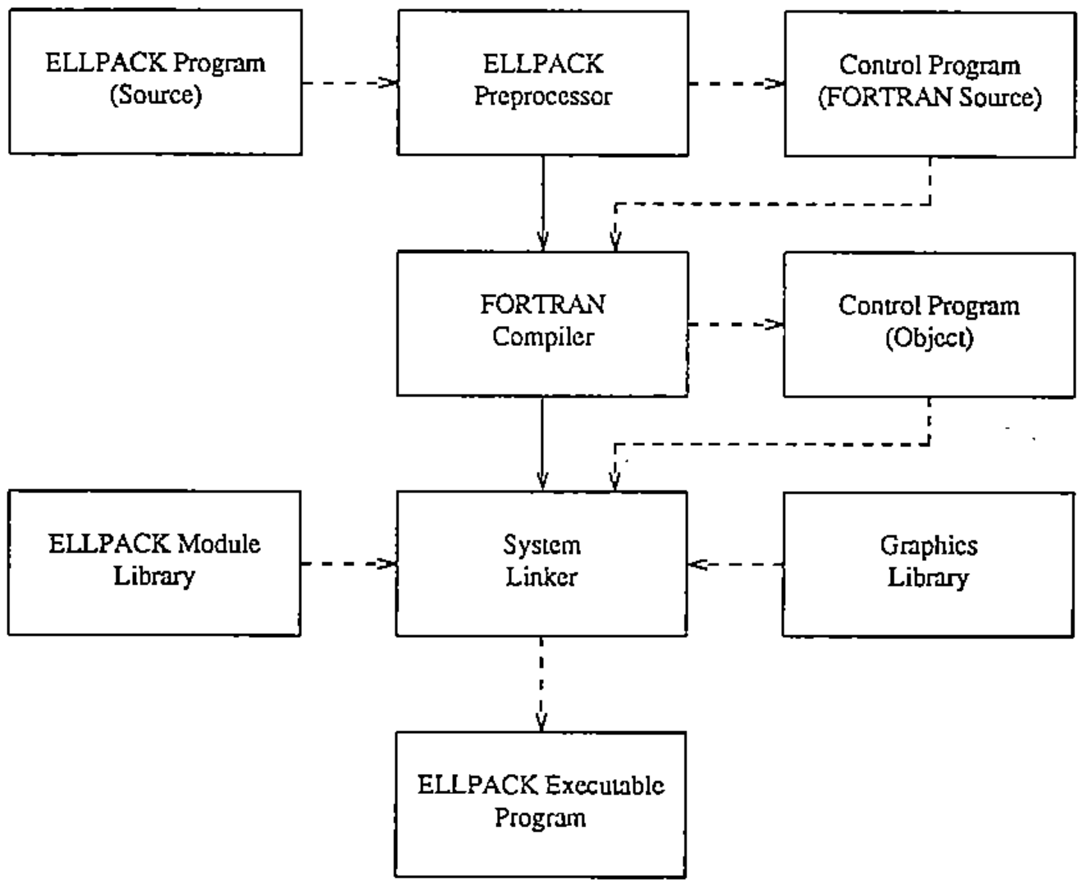

Figure 4.1. An overview of the ELLPACK system showing the three main steps in the generation of an executable ELLPACK program. The control program contains appropriate variable declarations and calls to ELLPACK system routines and modules. The development of Interactive ELLPACK required extending the ELLPACK preprocessor and the graphics interface.

recognize *menu* segments, we simply introduced new rules and actions to the preprocessor grammar and used PG to generate a new preprocessor. The rules specify the syntax of the new segment; the actions cause FORTRAN to be included in the control program to implement the menus, handle the interactive input/output, and provide for flow of control. A small section of the PG grammar defining Interactive ELLPACK is given in Figure 4.2.

Adding new modules to existing segments in ELLPACK is even easier than modifying the language itself. The preprocessor is "table-driven" in the sense that it reads from the ELLPACK *template* file to determine the characteristics of the current system and to define the basic structure of the control program. Hence, one can add to the set of modules recognized by the preprocessor or modify the structure of the control program by simply modifying the template. No change to the preprocessor itself is required. We were able to extend the set of *output* modules for example, by simply adding the necessary template variable definitions and sections of FORTRAN to the template.

The Template Processor described in [Ribbens, et al., 1984] is called by the preprocessor to process the template file. In addition to template variable or *macro* definitions, the template processor allows several forms of control directives, including iteration and conditional constructs. Hence, different sections of code may be included in the control program, depending on the modules selected and the graphics terminal being used. Recall that the terminal type is indicated by the Interactive ELLPACK option *terminal*. Figure 4.3 illustrates a typical section of the Interactive ELLPACK template.

4.3 The Graphics Interface. The graphics interface of Interactive ELLPACK consists of three layers of software. Figure 4.4 illustrates the relationships between the various software layers in an executing ELLPACK program. The highest layer is the library of ELLPACK modules such as the *grid* module *interactive* and the *output* modules *plot* and *plot3d*. At this layer, the software knows about the elliptic problem; that is, about the domain, the partial differential equation, the boundary conditions, the computed solution and any other related functions. For example, one routine at this layer draws the contour plots (level curves), completely independent of any graphics environment. Routines at this level which do graphics use a coordinate system which depends only on the problem domain.

The software at the intermediate level maps information from the ELLPACK device independent environment to a device specific graphics environment. These routines have knowledge of the attributes of the graphics device

```
*
*  a menu name is anything between single quotes
*
mname    ->        ''''  notqot*  ''''  eol
                        $(srcalls) <- '*set(menuname=''' // $2 // ''')$-'

*
*  a menu segment is one or more menu items
*
menseg  ->        mitem+
                        $l(hvmenu) = .true.
                        $(srcalls) <- '*set(i0nitm=' // $(i0nitm) // ')$-' &
                                   // '*set(keys=''' // $(mkey) // ''')$-' &
                                   // '*include(printmenu)$-'
                        $(i0nitm) = 0
                        inmenu = .false.

*
*  a menu item is a key, a label and one or more item statements
*
mitem    ->        blanks  ''''  notcol*  ':'  notqot*  ''''  newlin?  itstmt+
                        $i(i0nitm) = $i(i0nitm) + 1
                        if (inmenu)
                               if (lqmtch($3,''))
                                  $(mkey) <- $(i0nitm) // '$$/'
                               else
                                  $(mkey) <- $3 // '$$/'
                               endif
                        else
                               if (lqmtch($3,''))
                                  $(mkey) =  $(i0nitm) // '$$/'
                               else
                                  $(mkey) =  $3 // '$$/'
                               endif
                               inmenu = .true.
                        endif
                        if (lqmtch($5,''))
                           $(labels) <- $(itemlabel) // '$/'
                        else
                           $(labels) <- $5 // '$/'
                        endif
                        $(items) <- 'c$/'
                        initem = .false.
                        clear

*
*  an item statement may be a module, a grid segment, an output segment,
*  or a fortran segment
*
itstmt  ->        modtyp  restnm newlin?  mmodul           eol
        ->        'gr'    restnm newlin?  mgrseg ++ msep  mgrend
        ->        'ou'    restnm newlin?  motseg ++ msep  eol
        ->        'fo'    restnm          mftsmt+          eol
                          initem = .true.
                          $(itemlabel) = 'fortran'
```

Figure 4.2. Sample section of a PG grammar. Lines with arrows (->) are rules. Actions to be performed when a rule is matched are listed beneath each rule. Actions may use matched input: S1 is the first subrule matched, S2 the second, and so on. Strings may be assigned (=) or appended (<-) to template variables such as S(srcalls).

```
*if($def(hvp86))
*set(modname='plot')
*include(beginmodule)
*if (ttytype = 'dumb')
*else
      open(6,file='plot.$fcn',status='unknown')
*endif
*if (llrect)
      call q8plr2($fcn, '$fcn     ', $nx, $ny )
*else
      call q8plnr($fcn, '$fcn     ', $nx, $ny )
*endif
*if (ttytype = 'ridge')
      open(6,file='/dev/tty',status='old')
      call getplt('plot.$fcn',iddwin,menvew(mfdesc,'NewView'))
*endif
*if (ttytype = 'tek4115')
      open(6,file='/dev/tty',status='old')
      nxtseg = nxtseg + 1
      call seeplt (vewtab, vwgraf, nxtseg, 'plot.$fcn', 12)
*endif
*endif
```

Figure 4.3. Sample section of the template. A dollar sign prefixes a template variable. An asterisk in column one indicates a directive. Other lines are simply copied into the control program.

being used. For example, they know whether to used fixed views or windows for graphics. If fixed views, the number and location of each are known. This layer is aware of the availability of colors. The coordinate system of the highest layer is mapped to device dependent coordinates.

At the lowest level is the software which actually drives the graphics devices. They map commands used at the intermediate level such as *move* and *draw* to the escape sequences understood by the graphics terminal. Moreover, they may map the device dependent coordinates into some packed format. For example, on the Tektronix terminals the intermediate level command "move 848, 3072" gets mapped to "LFu0c@0".

Even though the graphics software is logically divided into three layers, Figure 4.4 shows that some intermediate level routines are called from the ELLPACK control program. Among the tasks of an Interactive ELLPACK control program are establishing the graphics environment, modifying it, and finally terminating it. All of these tasks are in the domain of the intermediate layer of the graphic software.

4.4 Other VHLL's. The techniques used to develop Interactive ELLPACK are applicable to any VHLL. The software tools used to build ELLPACK greatly reduced the implementation effort of Interactive ELLPACK. The preprocessor generator PG and the Template Processor can be used (and have been used) to generate natural interfaces to any suite of subprograms. Since the interactive graphics interface is divorced from the ELLPACK

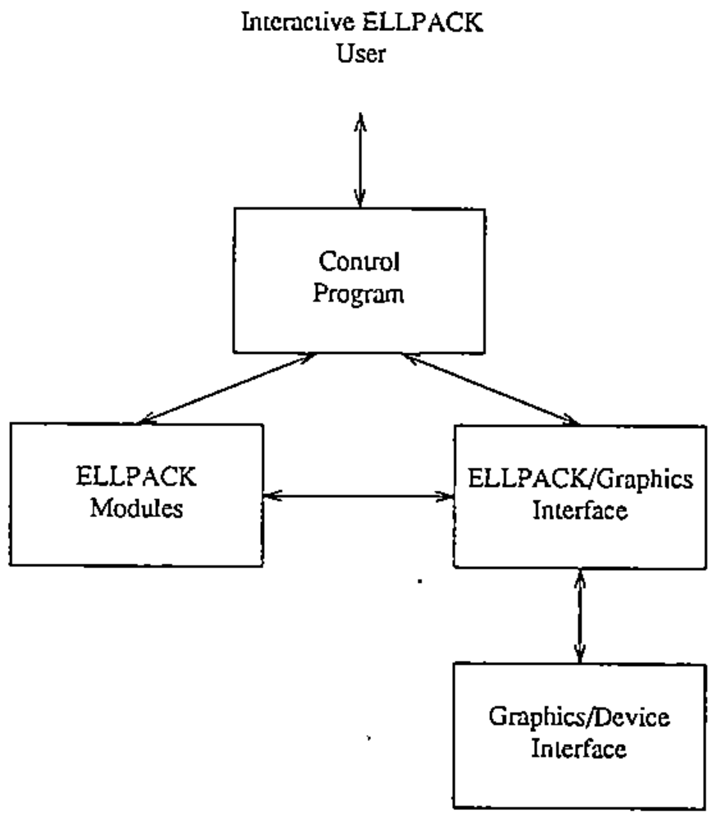

Figure 4.4. The relationship between the software layers in an executing ELLPACK program.

software, it can be used with any VHLL so developed.

5. Conclusions and Future Trends for ELLPACK

In the past twenty-five years, the scientific computing community has witnessed a revolution in computer hardware. Yet over this same time, scientific software has experienced only small uprisings here and there. Although the quality and quantity of available software has increased, the nature of the software has been rather constant—a library of FORTRAN callable routines.

We believe that problem oriented, very high level languages such as ELLPACK represent a first step toward the modernization of scientific computing. ELLPACK provides a natural interface to bridge the gap from the world of the scientist or engineer to the world of the numerical analyst. With such systems, new methods will be able to migrate from numerical analysis journals to users' hands in a natural, considerably faster way.

Interactive ELLPACK is designed to take advantage of the powerful personal workstations currently available. We believe that it represents a significant step in the development of scientific software since it makes full use of interactive color graphics output and input. It is currently being used as a research tool on a daily basis at Purdue.

Obviously, very large scale scientific computing is not interactive in the sense of obtaining immediate results. Someone whose task runs overnight on a CRAY might claim that systems like Interactive ELLPACK would be of no use to them. However, Interactive ELLPACK is a true extension of ELLPACK. Hence, we can use Interactive ELLPACK to "set up" a traditional ELLPACK run; e.g., interactively build a grid, etc. This program can be run as a traditional, batch oriented ELLPACK program for as long as the solution process takes. The results can then be analyzed within the Interactive ELLPACK environment.

ELLPACK views an elliptic problem solver as either a sequence of a discretization module followed by an indexing module followed by a solution module, or a single triple module. As a result, for a given problem ELLPACK contains 1147 (=9*7*18+13) distinct solution paths. Although its interface represents a significant improvement over ELLPACK, Interactive ELLPACK provides no help in choosing a solution method. To that end, we have begun work on *Elliptic Expert*, an extension of Interactive ELLPACK which uses knowledge-based decision making technology to advise the user in the selection of the "best" solution path (algorithm) for solving an elliptic problem.

6. References

1. R. F. Boisvert, E. N. Houstis, and J. R. Rice, "A system for performance evaluation of partial differential equations software", *IEEE Trans. Software Eng.*, 5(1979), 418-425.
2. J. P. Bonomo, W. R. Dyksen and J. R. Rice, "The ELLPACK performance evaluation system" Purdue University, Computer Science Department Report CSD-TR 569, January 1986.
3. C. J. Ribbens, J. R. Rice, and W. A. Ward, "A simple macro processor", *ACM Trans. Math. Software*, 10(1984),410-416.
4. C. J. Ribbens, "Domain transformations: a tool for vector algorithms for the numerical solution of partial differential equations", Ph.D. Thesis, Purdue University, 1986.
5. J. R. Rice, E. N. Houstis, and W. R. Dyksen, "A population of linear, second order, elliptic partial differential equations on rectangular domains, Parts 1 and 2", *Math. Comp*, 36(1981), 475-484.
6. J. R. Rice, and R. F. Boisvert, *Solving Elliptic Problems Using ELLPACK*, Springer-Verlag, New York, 1985.
7. J. R. Rice, "Is the aspect ratio significant for finite element problems?", Purdue University, Computer Science Department Report CSD-TR 535, September 1985.
8. J. R. Rice, "Adaptive tensor product grids for singular problems", *Algorithms for the Approximation of Functions and Data*", (James Mason, ed.), Oxford University Press, 1986.
9. J. R. Rice, W. R. Dyksen, E. N. Houstis, and C. J. Ribbens, "ELLPACK project status report", Purdue University, Computer Science Department Report CSD-TR 579, March 1986.

Appendix A

ELLPACK is an extension of Fortran that allows one to simply state and solve elliptic partial differential equations in 2-dimensions on general domains or in 3-dimensions on rectangular domains. For example, the elliptic problem

$$
\begin{array}{ll}
u_{xx} + u_{yy} + 3u_x - 4u = \exp(x+y)\sin(\pi x) & 0 < x < 1,\ -1 < y < 2 \\
u = 0 & x = 0,\ -1 < y < 2 \\
u = x & 0 < x < 1,\ y = 2 \\
u = y/2 & x = 1,\ -1 < y < 2 \\
u = \sin(\pi x) - x/2 & 0 < x < 1,\ y = -1
\end{array}
$$

is solved using ordinary finite differences and Gauss elimination by the ELLPACK program

```
OPTIONS.   TIME $ MEMORY
EQUATION.  UXX + UYY + 3.*UX - 4.*U = EXP(X+Y)*SIN(PI*X)
BOUNDARY.
           U = 0.0                ON X =  0.
           U = X                  ON Y =  2.
           U = Y/2.               ON X =  1.
           U = SIN(PI*X) - X/2.   ON Y = -1.
GRID.      6 X POINTS $ 12 Y POINTS
*
DISCRETIZATION.   5 POINT STAR
INDEXING.         AS IS
SOLUTION.         LINPACK BAND
*
OUTPUT.           TABLE(U) $ PLOT(U)
END.
```

GENERAL CONVENTIONS

The independent variables are X, Y (and Z in 3-dimensions), the solution is U with derivatives *UX*, *UY*, *UXX*, etc. If the solution is used as a function (not seen in the above example), then *U*(*X*,*Y*), *UX*(*X*,*Y*), etc. provide the corresponding values at (*X*,*Y*). All expressions are in Fortran. The keywords OPTIONS, GRID, etc. start in column 1, otherwise column placement is optional. A long expression may be continued to the next line by ending the line with an ampersand &. The normal structure of an ELLPACK program is

OPTIONS.	Specifies options of the run.
EQUATIONS.	Specifies the elliptic PDE.
BOUNDARY.	Gives domain and boundary conditions.
GRID.	Specifies lines of a rectangular grid.

problem solving, output and Fortran statements

SUBPROGRAMS.	Fortran functions and subroutines used
END.	

OTHER DOMAIN DEFINITION SEGMENTS

HOLE.	Specifies a hole in the domain with boundary conditions. Must follow BOUNDARY segment.
ARC.	Specifies a slit or interface in the domain. Must follow BOUNDARY segment.

The domain and boundary conditions are specified by a list of items like

condition ON *piece*

where *condition* is one of

PERIODIC

A*UX + B*UY + C*U = D

with *A*, *B*, *C* and *D* Fortran expressions. Terms with zero coefficients may be omitted. The domain boundary is given in parameterized pieces so that *piece* appears as

X = *Expression*(*t*), Y = *Expression*(*t*) FOR *t* = a TO b

Here, *t* is a parameter with X and Y given as expressions; a and b must be constants. The pieces are given in counter-clockwise order and must join up to form a closed curve. There is a special simple form for lines illustrated by

U = X-Y+2 ON LINE 0,0 TO 1,1 TO 3.6,2.4 TO 1,-1 TO 0,0

The points to be joined by straight lines are listed in order. Finally, there is an even simpler special form for rectangles illustrated by the example above.

The grid is specified by

k variable POINTS *point-list*

where *k* is normally a constant, *variable* is X, Y, or Z and *point-list* is the set of values defining the grid lines. For uniformly spaced grids *point-list* may be *a* TO *b* where *a* and *b* are constants; for uniformly spaced grids on rectangles the *point-list* may be omitted entirely.

BASIC PROBLEM SOLVING MODULES

Most numerical methods consist of the three steps seen in the above example. The basic facilities in ELLPACK are

DISCRETIZATION:

5 POINT STAR	Ordinary finite differences
SPLINE GALERKIN	Galerkin method with piecewise polynomials, rectangles only
HERMITE COLLOCATION	Collocation with Hermite bi-cubic polynomials on rectangles
HODIE	High order finite differences for operator without u_{xy} term on rectangles
COLLOCATION	Collocation with Hermite bi-cubics

INDEXING:

AS IS	Ordering not changed (optional)
RED-BLACK	Checker board ordering
NESTED DISSECTION	Nested dissection ordering
MINIMUM DEGREE	Minimum degree ordering

SOLUTION:

BAND GE	Band Gauss elimination
LINPACK SPD BAND	Cholesky decomposition for symmetric, positive definite band matrices
JACOBI CG	Jacobi iteration with conjugate gradient acceleration
SOR	SOR iteration
SPARSE -LU COMPRESSED	Sparse matrix method

TRIPLE:

FFT-9 POINT	Fast Fourier Transform for rectangles

There are more than 25 other problem solving modules in the complete ELLPACK system.

The ELLPACK system is available from John R. Rice, Department of Computer Science, Purdue University.

OTHER ELLPACK SEGMENTS

OPTIONS: (Fortran option variables given where available)

- MAX GRID = NX, NY or NX, NY, NZ to set max grid size for variable grids.
- INTERPOLATION = QUADRATICS (default) = SPLINES
- LEVEL = 0 to 5 governs amount of output (I1LEVL).
- MEMORY produces memory use estimate.
- NO EXECUTION suppresses execution.
- PAGE = 0 to 2 sets page advances from none to 1 before each module (I1PAGE).
- SELF-ADJOINT = logical switch (L1SELF).
- TIME produces time use estimate (L1TIME).
- MAX WORKSPACE = Set dimension I1KWRK of R1WORK to indicated value.
- MIN WORKSPACE = Dimension I1KWRK of R1WORK is at least indicated value.
- CLOCKWISE for boundary pieces given in clockwise order.

OUTPUT:

- MAX, RMS, NORM are the same, used as MAX(f) or MAX(f, grid) where f is any Fortran function with 2 (or 3) arguments and grid is like 20, 20.
- PLOT gives contour plot, arguments are as in MAX.
- PLOT DOMAIN displays domain with grid lines.
- TABLE gives table of values, arguments are as in MAX.
- SUMMARY = TABLE & MAX with arguments f or f, grid.

FORTRAN: Fortran code follows.

DECLARATIONS: Fortran declarations follow.

ADVANCED ELLPACK SEGMENTS

GLOBAL: Inserts Fortran declarations in all ELLPACK subprograms Q1PCOE, R1PRHS, Q1BCOE, R1BRHS, Q1BCOR.

PROCEDURE: Provides other useful facilities

- EIGENVALUES: Computes eigenvalues of discretization matrix.
- DISPLAY MATRIX PATTERN: Gives pattern of non-zeros in discretization matrix.
- SET UNKNOWN FOR 5 POINT STAR (U = *name*)
- SET UNKNOWN FOR HODIE HELMHOLTZ (U = *name*)
- REMOVE BLENDED BC: Subtract blending function interpolant of boundary conditions from U.
- REMOVE BICUBIC BC: Subtract bi-cubic interpolant of boundary conditions from U.
- REMOVE (V=*name*) Subtract named function from U.

OPTIONS:

- Array dimension control allows one to set Fortran array dimensions independent of what the ELLPACK system thinks they should be.
- Problem characteristics may be set by assigning .TRUE. or .FALSE. to the logical variables L1LAPL, L1CSTC, L1POIS, L1HMEQ.
- TABLE followed by PROBLEM, EQUATIONS, INDEXES, UNKNOWN, DOMAIN or BOUNDARY provides data about internal ELLPACK variables.

Preprocessor Variables: Certain preprocessor variables can be used in the ELLPACK program, some are $I1KBAN, $I1KWRK, $I1MEND, $I1MIXD, $I1MNCO, $I1MXEQ, $I1MXPT, $I1NBND, $I1MXPT, $I1NGRX, $I1NGRY, $I1NGRZ.

TRIPLE: The following triples directly set U(X,Y) for use in initializing iterations or homogenizing boundary conditions. One can use U(X,Y), UX(X,Y), UXX(X,Y), etc. after these triples.

- SET U BY BLENDING Use blending functions from interpolating boundary conditions.
- SET U BY BICUBICS Use Hermite bicubics from interpolating boundary conditions.
- SET (U = *name*) Define U from function named.

REST OF PROBLEM SOLVING MODULES

DISCRETIZATIONS: (for rectangles unless otherwise noted)

HODIE	High order finite differences for nonrectangular domains
HODIE HELMHOLTZ	High order finite differences for $u_{xx} + u_{yy} + f(x,y)u = g(x,y)$
7 POINT 3D	Ordinary finite differences in 3D
INTERIOR COLLOCATION	Variant of HERMITE COLLOCATION which provides better performance for uncoupled boundary conditions
SPLCOL	Collocation with splines of order 3, 4, 6 and 8

INDEXING:

HERMITE COLLORDER	reorders collocation matrix to have nonzero diagonal
REVERSE CUTHILL MCKEE	reorders equations to possibly help envelope or band solvers

SOLUTION: (GE means Gauss elimination)

BAND GE NO PIVOTING	Band GE without pivoting
LINPACK BAND	LINPACK routine for band GE
JACOBI SI	Jacobi iteration with semi-iteration
SYMMETRIC SOR CG	Symmetric SOR with conjugate gradient acceleration
SYMMETRIC SOR SI	Symmetric SOR with semi-iteration
REDUCED SYSTEM CG	Iteration on RED-BLACK with conjugate gradient acceleration
REDUCED SYSTEM SI	Iteration on RED-BLACK with semi-iteration
SPARSE	
-LU PIVOTING	GE, sparse matrices with pivoting
-LU UNCOMPRESSED	GE, nonsymmetric system
-LU COMPRESSED	GE, nonsymmetric with storage compression
-GE NO PIVOTING	GE, fast version, no pivoting
-LDLT	Symmetric version of SPARSE
ENVELOPE	
-LDU	GE, system in envelope form
-LDLT	GE, symmetric system

TRIPLE: (for rectangles unless otherwise noted)

FISHPAK HELMHOLTZ	Fast solver using ordinary finite differences for $u_{xx} + u_{yy} + \lambda u = f(x,y)$
HODIE 27 POINT 3D	Fast solver for 6^{th} order finite differences for Poisson problem on a cube
MARCHING ALGORITHM	Ordinary finite differences with generalized marching algorithm for separable, self-adjoint problem
DYAKANOV CG	Ordinary finite differences with preconditioned CG iteration for nonseparable, self-adjoint problem
DYAKANOV CG 4	Richardson extrapolation for DYAKANOV CG to obtain fourth order accuracy
P2C0 TRIANGLES	Galerkin method with 6-nodes quadratic triangular elements, only for non-rectangular 2D domains.
CMM EXPLICIT	Capacitance matrix methods for Poisson problem on non-rectangular 2D domains
CMM IMPLICIT	
MULTIGRID MG00	Multigrid method for $au_{xx} + cu_{yy} - fu = g$ with mixed boundary conditions