1988

XELLPACK: An Interactive Problem Solving Environment for Elliptic Partial Differential Equations

Jophn P. Bonomo

Wayne R. Dyksen

Report Number:
88-839

XELLPACK: AN INTERACTIVE PROBLEM-SOLVING ENVIRONMENT FOR ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

John P. Bonomo
Wayne R. Dyksen

CSD-TR-839
December 1988

XELLPACK:
An Interactive Problem-Solving Environment for Elliptic Partial Differential Equations

John P. Bonomo [†]
Wayne R. Dyksen [‡]

Purdue University
Department of Computer Sciences

CER-88-47, CSD-TR-839

[†]Supported in part by National Science Foundation grant CCR-8619817

[‡]Supported in part by National Science Foundation grants CCR-8612676 and DCR-8602385

XELLPACK: An Interactive Problem-Solving Environment for Elliptic Partial Differential Equations

John P. Bonomo [†]
Wayne R. Dyksen [‡]

Abstract

ELLPACK is a very high-level language designed for solving second order, linear elliptic partial differential equations in two and three dimensions with Dirichlet, Neumann, mixed or periodic boundary conditions. The typical elliptic problem solving process is iterative; one repeatedly computes a solution, analyzes the results and then modifies the solution process. To better serve this process, we have developed XELLPACK, an extension of ELLPACK based on the X windowing environment. XELLPACK provides graphics input for constructing grids, pop-up menus for selecting solution techniques, and color graphics output for analyzing solutions. Using the X paradigm, a user can interface with XELLPACK from any X workstation while an XELLPACK client solves an elliptic problem on any machine or machines on the network.

[†]Supported in part by National Science Foundation grant CCR-8619817

[‡]Supported in part by National Science Foundation grants CCR-8612676 and DCR-8602385

1 Introduction

Over the last several years the amount of computer power available to the average scientific programmer has increased dramatically with the introduction of powerful parallel and vector machines into the market. Problems that were previously considered intractable or unmanageable are now being solved routinely on these machines in research and industrial institutions around the world. Similar advancements in scientific software have also been achieved, although at a significantly slower rate. *Very High-Level Languages* (VHLLs) have been developed to free scientists and engineers from the low-level details of scientific programming. Typically VHLLs present to the user a high-level interface which allows him to define his problem in a natural way and to access state of the art equipment and algorithms.

Two prime examples of VHLLs are the ELLPACK and Interactive ELLPACK systems. ELLPACK [5] was developed as a VHLL to solve elliptic partial differential equations (PDEs). Though designed initially as a testbed for elliptic algorithms, it also serves as a powerful problem solving tool. To aid in both these uses of ELLPACK, Interactive ELLPACK [4] was developed to allow run-time graphical interaction between the programmer and ELLPACK.

The advent of workstations and windowing software calls for a further advancement in scientific software. The computing power made available to the scientist has again increased enormously as it is now possible to open one or more windows on a workstation, each which may be running programs on other, more powerful machines. In order to take advantage of this increased access to sophisticated hardware, as well as other features offered by a windowing environment, we have developed *XELLPACK*, an elliptic problem solving environment based on the X Window System[1].

In this paper we describe the current state of XELLPACK. Sections 2 and 3 give brief summaries of ELLPACK and Interactive ELLPACK. Section 4 describes the X Window System. In Section 5 we describe the main features of XELLPACK. We close in Section 6 with a discussion of future modification and improvements planned for XELLPACK.

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 the following:

1. ELLPACK, a very high-level language for solving elliptic problems [5];
2. the Elliptic PDE Population, a population of 56+ parameterized elliptic problems (190+ instances) used by the Performance Evaluation System [6]; and,

[1]The X Window System is a trademark of MIT

3. the Performance Evaluation System, a system for the generation, collection and analysis of data on the performance of elliptic algorithms [1], [3].

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

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

can be solved by the ELLPACK program shown in Figure 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 1: Sample ELLPACK program.

An ELLPACK program consists of several *segments.* The elliptic problem is defined by the *equation* and *boundary* segments. The boundary segment allows description of non-rectangular, parameterized domains as well as simple rectangular ones. It also allows the placement of holes and arcs within a domain.

ELLPACK contains four basic types of problem-solving *modules. Discretization modules* discretized 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 these steps into one module.

ELLPACK contains several other segments and modules which are not explicitly involved in the PDE solution process. *Output* modules allow the user to graph various functions (computed solution, true solution, residual, etc.) in either two or three dimensions. *Procedure* modules provide routines which are useful in analyzing an elliptic problem or which assist in the solution process. For example one procedure module prints out the non-zero pattern of the generated linear system for a PDE; another initializes the unknowns if an iterative solution scheme is used. The *fortran* and *subroutine* segments allow the inclusion of FORTRAN code into the ELLPACK program. Fortran segments can appear anywhere after the declarative segments, while the subroutine segment contains user supplied FORTRAN subroutines which can be called from the ELLPACK program.

Though ELLPACK was initially developed as an environment for evaluating the performance of algorithms and software for elliptic PDEs, it is now recognized as a very powerful tool for solving a large class of problems. Its modular design allows it to be used to solve problems other than second-order, linear elliptic problems. For examples, we have used ELLPACK to solve coupled systems of elliptic equations, nonlinear elliptic problems and time-dependent problems.

3 Interactive ELLPACK

Solving PDEs with ELLPACK is typically an iterative process. One begins by computing an initial solution using an arbitrary grid and solution method. This solution is then analyzed and an estimation of its accuracy is determined. Once this is determined, a new grid may be constructed by either adding new lines or moving existing lines; grids are constructed both computationally and visually. The solution method may also be modified by adjusting module parameters, or a totally new method may be chosen. A new solution is then computed, and the above procedure is repeated until a satisfactory solution is obtained.

When using the batch oriented ELLPACK system, the above procedure is often unduly long and tedious, resulting in much wasted time. To speed up this process, *Interactive ELLPACK* was developed. Interactive ELLPACK [4] is an extension of ELLPACK which includes several important new features:

1. a *menu* segment to build user designed menus to allow the run time selection of ELLPACK modules;

2. an *interactive grid* module which allows the user to view, specify and change grids via interactive graphical devices throughout the execution; and,

3. new two and three dimensional color graphics output modules.

These features greatly speedup the iterative solution process described above. The graphics output modules allow the user to instantly view the results of the current grid/solution combination. The user-defined menus allow selection of multiple solution methods as well as allowing the run time modification of parameterized modules. To aid in the modification of the grid, the interactive grid module allows the current grid to be superimposed over any two dimensional graphics output such as the residual or some other error estimating function.

4 The X Window System

X is a windowing system developed as part of the Athena project at MIT. It is network transparent and designed to run under 4.3BSD UNIX[2] and Ultrix[3] Version 1.2. X runs on computers with either monochrome or color bitmap displays. We give a brief overview here for readers unfamiliar with the X Window System.

X uses the *server-client* paradigm. A set of screens for a single user with one keyboard and one motion device (such as a mouse) is called a *display*. The display is managed by an X *display server* which distributes user input to, and accepts output request from various client programs. These client programs can be local to the server's machine or may reside elsewhere on a computer network connected to the server's machine. Multiple clients connected to the same X server can exist on any machine. Figure 2 illustrates an X server which has six clients connected to it; two of these clients are local and four are located on remote machines, connected to the X server over a network.

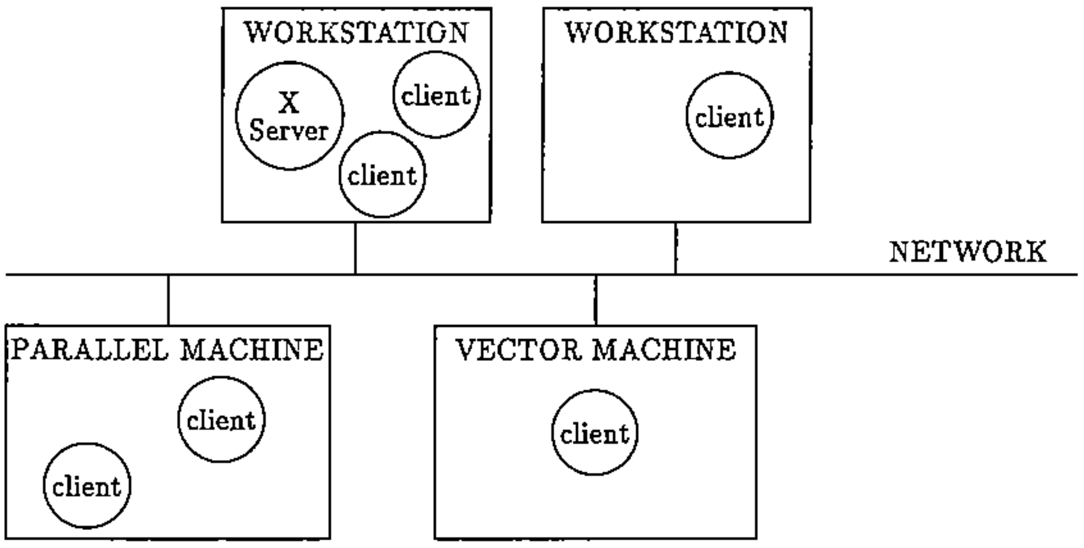


Figure 2: An X server and six clients distributed over a network.

[2]UNIX is a trademark of AT&T Bell Laboratories

[3]Ultrix is a trademark of Digital Equipment Corporation

An X server can handle multiple displays, with multiple, overlapping windows within each display. The main window for each screen is called the *root window* and it covers the entire screen. Multiple windows on the screen are stored as a tree structure, with the root window as the root of the tree. This hierarchy is used to specify the inheritance of window properties and the propagation of input events. Trees can be of arbitrary length and bushiness. Although a child window can be defined to extend outside the extent of it's parent's window, output to that child window is constrained to lie within the bounds of the parent. Any clipping necessary is performed automatically by the server.

If several sibling windows overlap, one of them is considered to be on top and is not obscured; output to the other siblings is suppressed if it falls within the overlap region. At any time one of these lower sibling windows may be *raised* and brought to the top, in which case it's contents must be restored. The X server does not retain the contents of windows and leaves the responsibility of *repainting* the window contents to the client application. When any part of a window becomes unobscured, the X server informs the client which part of the window needs to be repainted.

The X server provides some off screen resources to aid in repainting windows. Any window or part of a window may be stored as a *pixmap* in which the actual pixel values which make up the designated area are saved off screen in the server's address space. Pixmaps also exist as entities of there own apart from windows; for example, you can draw directly onto a pixmap and then later copy the contents of that pixmap to a window. Windows and pixmaps are both referred to as *drawables*.

The X server communicates with its client applications in two ways; either through replies to requests from a client (e.g. a query on window characteristics) or through *events*. Events are generated asynchronously by the X server in response to device activity (e.g. moving a mouse or pressing a key) or as side effects to client requests (e.g. raising or moving a window). Each window has associated with it a set of events that it recognizes and responds to; these events are said to be *selected* by the window. If an event is generated in some window which does not recognize it, it is passed up through that window's ancestors until it finds a window which has selected that event, or until it is explicitly discarded by some window. The types of events generated by the X server can be organized into several categories; for example keyboard events (`KeyPress` and `KeyRelease` events), pointer motion events (`ButtonPress, ButtonRelease, MotionNotify`), window crossing events (`EnterNotify, LeaveNotify`), and exposure events (`Expose`).

Client applications interface with the X server via routines from a C routine library called *Xlib*. Xlib routines allow the client to connect to the X server via a stream connection and send and receive information from it. In order to minimize the amount of network overhead, events are queued until there is an explicit request for them. Output requests generated by Xlib routines are also queued in order to keep network requests to a minimum. While this queue can be explicitly flushed with the `XFlush` routine such action is in practice rarely needed since this action is a side effect whenever the client attempts to read any event or reply from the server.

The X server is started on a machine using the *xinit* command. A client application must also be started up along with the X server; once this client terminates, *xinit* kills the server and then terminates itself. By default, the initial client application is *xterm*, a terminal emulator which starts up a shell on the local machine. *Xterm* emulates a DEC VT102 terminal and provides labeling of the window, scrolling, and cut and paste abilities. Once the server is running, more *xterms* (or other clients) can be started, either locally or remotely. Also, an *xterm* can be instructed to execute a command other than a shell; once this command is completed, the *xterm* terminates.

5 XELLPACK

5.1 Overview

XELLPACK is the result of applying the facilities of the X Window System to the methodology of Interactive ELLPACK. Figure 3 shows an XELLPACK program to solve the elliptic problem given in (1). XELLPACK is started by invoking an *xterm* with a compiled XELLPACK program as its argument. The resulting window is called the *XELLPACK dialog window* and is used for all text input and output throughout the XELLPACK run. Keyboard input is *focused* on the dialog window so that the cursor does not have to be located in the window when typing. XELLPACK is event driven and does nothing until the user presses the left mouse button to get the first menu. Figure 4 shows an example display of the XELLPACK program in Figure 3.

5.2 Menus

There are two types of menus in any XELLPACK program: user-defined menus and built-in menus. The syntax for an XELLPACK used-defined menu segment is:

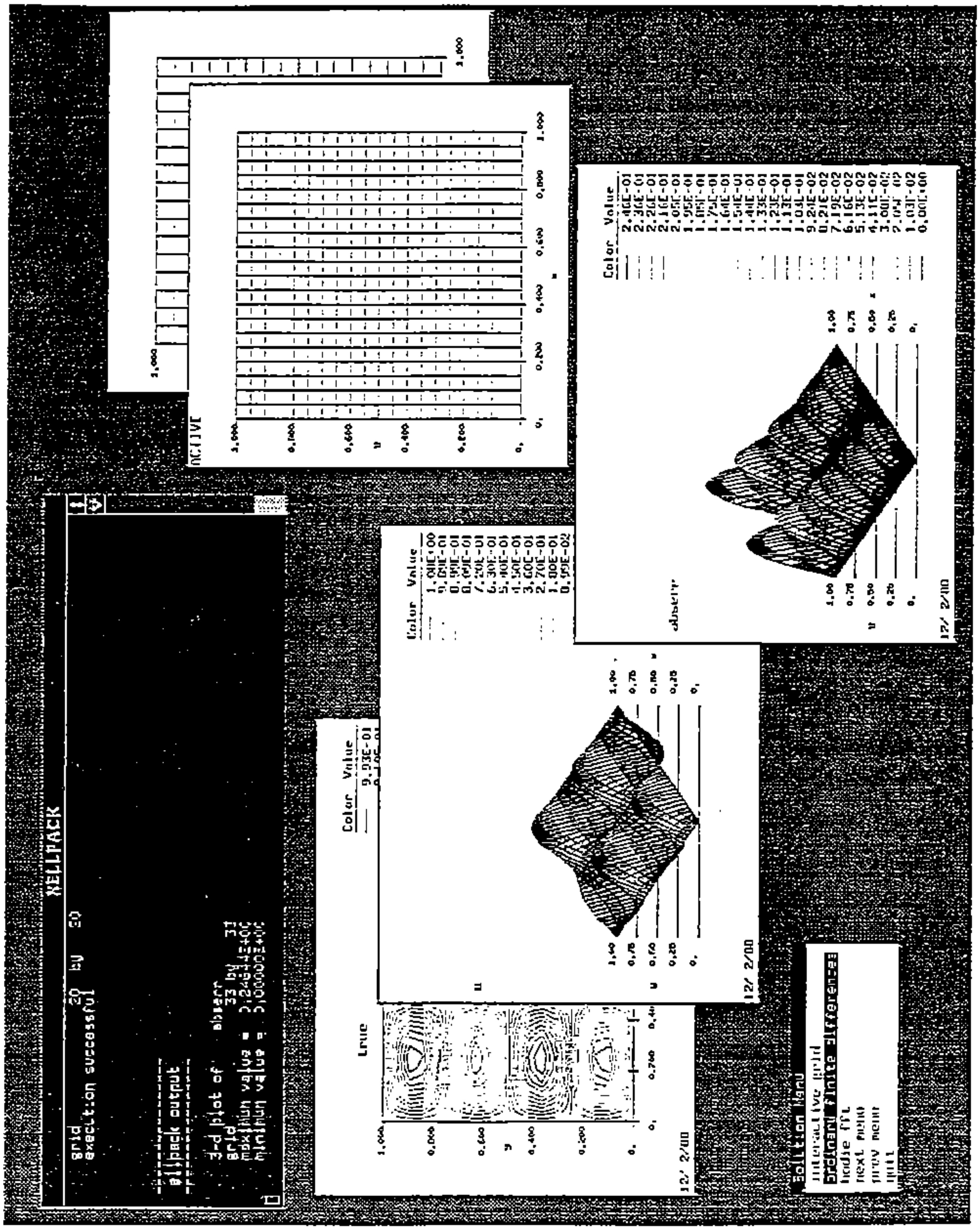
```
menu.   '<menu name>'
         <menu item>
         <menu item>
             .
             .
             .
         <menu item>
```

where each `<menu item>` is of the form:

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

```
options.
                max x points = 33 $ max y points = 33
                interpolation = splines

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
        'fd:ordinary finite differences'        dis.    5 point star
                                                sol.    band ge
        'hf:hodie fft'                          tri.    hodie fft

menu.   'Output Menu'
        'ct:contour true'                   out.    plot(true)
        'cu:contour u'                      out.    plot(u)
        'ce:contour error'                  out.    plot(error)
        'ca:contour abserr = abs(error)'    out.    plot(abserr)
        'gt:graph true'                     out.    plot3d(true)
        'gu:graph u'                        out.    plot3d(u)
        'ge:graph error'                    out.    plot3d(error)
        'ga:graph abserr = abs(error)'      out.    plot3d(abserr)
end.
```

Figure 3: An XELLPACK program to explore the use of different methods (5 point star and hodie fft) and grids in solving an Helmholtz problem.



```
grid                  20 by  20
execution successful

-------------
ellpack output
-------------

3-d plot of      absern
grid                33 by  33
maximum value =  0.24644E+00
minimum value =  0.00000E+00
```

Figure 4: XELLPACK display showing dialog window (the black window), two interactive grid windows, three output windows and one menu.

The string `<menu name>` is used as a label for the menu. Each menu selection is identified by the `<label>` field in the item specification (the optional `<key>` field is for compatibility with Interactive ELLPACK and is not used by XELLPACK). Once the user selects a menu item, XELLPACK executes the ELLPACK commands listed in `<item definition>`. This list can be of arbitrary length and contain any (viable) combination of discretization, indexing, solution, triple, fortran, procedure and output segments. The program in Figure 3 includes two user-defined menus. The first allows selection of the interactive grid module and two solution procedures. Note that the first solution method "`ordinary finite differences`" is associated with two ELLPACK modules: a 5-point star discretization module and a band Gauss elimination solution module. XELLPACK appends three extra menu selections to the end of every user- defined menu: "`next menu`", "`prev menu`" and "`quit`". There is no limit to the number of menus allowed in an XELLPACK program. An example of the two menus defined in the XELLPACK program in Figure 3 are shown in Figure 5.

Several built-in menus are incorporated into XELLPACK. The graphics utility menu (Figure 6) allows selection of routines which process XELLPACK graphics output window. Currently the only selections are "`put window`", "`get window`" and "`delete window`". The first two selections allow the user to save and restore window graphics. Future versions will allow the user to resize windows as well. Two other built-in menus are produce by the interactive grid module (Figure 7). They are discussed further in Section 5.4.

5.3 XELLPACK Output

XELLPACK contains several output modules which produce graphic plots of any user specified function. If *function* is a FORTRAN function, then *plot(function)* produces a two-dimensional contour plot of level curves, *plot2d(function)* produces a two-dimensional panel plot, and *plot3d(function)* produces a three-dimensional rendering of *function* (see Bonomo and Dyksen [2] for a description of the *plot3d* module for an earlier version of Interactive ELLPACK). The plotted functions can either be user-defined (using the subroutine segment) or built-in. Examples of built-in functions are: u, the computed solution, ux, the x derivative of the computed solution, and *residu*, the residual of the calculated solution. One other output module, *plot domain*, is used to output the boundary of the domain (this is typically only of interest with non-rectangular domains).

Each output module open a new window and draws onto it. The dimensions of the window are scaled to the dimensions of the graphed function, but once they are set they cannot be changed. Future versions of XELLPACK will allow rescaling of output windows. The initial placement of the window and subsequent movement is controlled by the local window manager. After a module finishes drawing onto the window, the pixel image in the window is stored by the X server as a

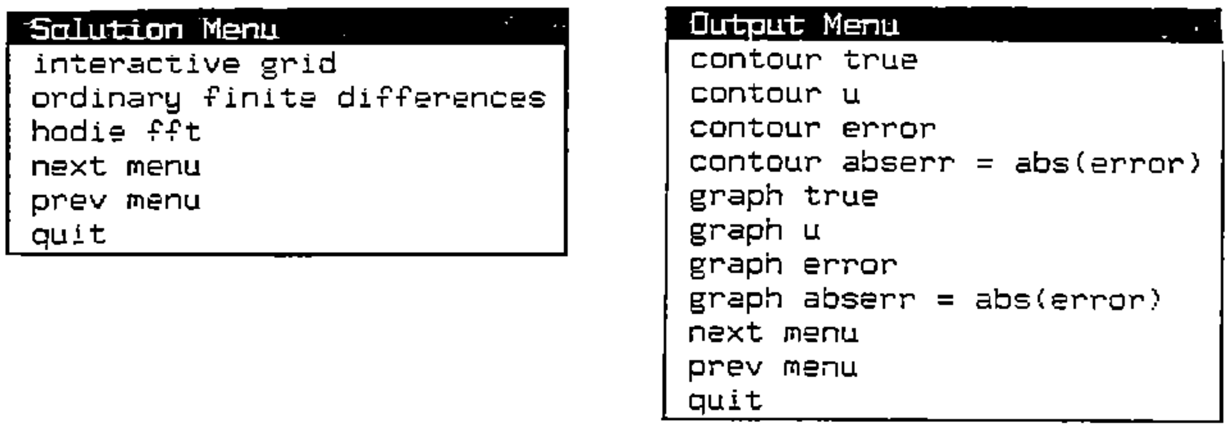


Figure 5: XELLPACK user-defined menus created from the XELLPACK program in Figure 3. Note that three extra selections are appended to the end of each menu.

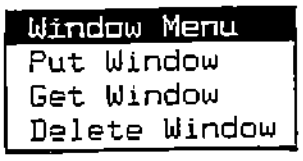


Figure 6: Built-in XELLPACK window utility menu.

pixmap which is later used to restore window contents when all or part of the window is hidden and then re-exposed. Since pixmaps are a limited resource in X, once a window is deleted, its pixmap is also deleted. Entire windows can be saved and viewed later using the X client *xwd* (X window dump) and *xwud* (X window undump).

Several ELLPACK output and procedure modules produce purely textual output; e.g., the output module *max(function)*. All textual output appears in the XELLPACK dialog window.

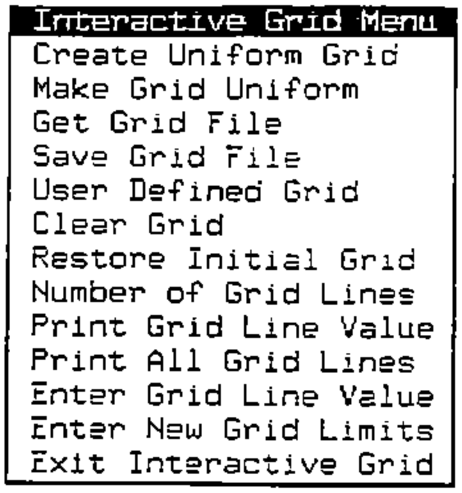
5.4 XELLPACK Input

There are two type of XELLPACK input: keyboard input in response to either a built-in or user-defined menu selection, and interactive grid input. All keyboard input events are constrained to be recognized by the XELLPACK dialog window (a process called *focusing*). This facilitates keyboard input as otherwise the user would have to move the screen cursor onto the dialog window prior to each keystroke.

The *interactive grid* module requires both keyboard input (handled in the manner described above) and mouse input. XELLPACK allows one or more grids to be defined at any one time; the current grid that is being used in solving the PDE is called the *active* grid. Interactive grid can be used to create a new grid, or update an existing grid. When creating a new grid, the module first opens a window scaled to the size of the domain, copies the currently active grid onto the new window and makes this new grid the active grid. When an XELLPACK program starts up, the default active grid is a 2×2 grid corresponding to the limits of the smallest rectangle which encloses the domain. When updating a grid, the module raises the grid window and makes it the active grid. The active grid is identified by the word `ACTIVE` printed in the upper left hand corner of the grid window (see Figure 4).

Once a grid has been selected it can then be modified. Figure 7 shows the two built-in interactive grid menus. The larger of the two is the main menu and allows selection of various routines to modify or query the grid. The smaller menu is used to specify which dimension various routines are to be applied. It pops up automatically when the user specifies interactive grid selections "`Create Uniform Gird`", "`Make Grid Uniform`", "`Clear Grid`", "`Print All Grid Lines`" and "`Enter Grid Line Value`".

Adding and deleting grid lines is performed by the mouse. When the cursor enters the grid domain, it turns into a cross hair. At this point, clicking the left mouse button will insert an x grid line at the current cross hair location; clicking the right mouse button inserts a y grid line; and clicking the center button adds both and x and y grid lines. Deleting lines is done in an analogous manner, but with the shift key pressed simultaneously with the mouse buttons. Grid lines may also be added numerically using the "`Enter Grid Line Values`" selection, with grid line values



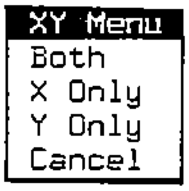


Figure 7: Interactive grid built-in menus. The smaller menu is used to specify the dimension(s) in which interactive grid routines are applied.

specified using the keyboard.

All the interactive grid selections apply to both rectangular and non-rectangular domains except the "`Enter New Grid Limits`" selection which applies only to non-rectangular domains. This menu selection allows the user to specify and modify an enclosing rectangle over the domain. For rectangular domains, the first and last grid lines are constrained to lie along the x and y limits of the domain.

6 Future Plans for XELLPACK

While no major changes to XELLPACK are planned in the near future, several modifications and improvements are necessary to make XELLPACK more usable in a problem solving environment. The ability to resize windows will be implemented, though it must be decided whether or not to constrain the resized window to have the same aspect ratio as the original window. In order to interactively resize windows, XELLPACK must build and store *graphic display lists* for every graphed function. Graphic display lists contain the contents of an output window in an intermediate representation (such as POSTSCRIPT[4]); once the size of a window is determined, the intermediate description of the window is translated into the proper X Window System graphic commands.

In the current implementation of XELLPACK, the interactive grid windows are treated very differently from the output windows. In future version, we hope to reconcile some of these dif-

[4]POSTSCRIPT is a trademark of Adobe Systems Incorporated

ferences. In particular, we hope to be able to query output windows in the same manner that one queries interactive grid windows. For example, given an output window, the user will be able to find out the grid/solution combination that resulted in the function being plotted. Also under investigation is the concept of "updating" an output window; for example, for a three dimensional output window this might mean allowing rotation of a function.

At the time XELLPACK was written, X10 was the latest version of the X Window System available. However with the recent introduction of X11 and its claims to being the new X standard, XELLPACK will be rewritten to conform to X11 standards. It should be noted that in this paper we have often used X11 terminology and concepts in anticipation of this planned update.

References

[1] R. F. Boisvert, E. N. Houstis, and J. R. Rice. A system for performing evaluation of partial differential equations software. *IEEE Trans. Softw. Eng.*, 5:418–425, 1979.

[2] J. P. Bonomo and W. R. Dyksen. *Three Dimensional Graphics Software in Interactive ELL-PACK.* Technical Report CSD-TR 674, Purdue University, 1987.

[3] J. P. Bonomo, W. R. Dyksen, and J. R. Rice. *The ELLPACK performance evaluation system.* Technical Report CSD-TR 569, Purdue University, 1986.

[4] W. R. Dyksen and C. J. Ribbens. Interactive ELLPACK: an interactive problem-solving environment for elliptic partial differential equations. *ACM Transactions on Mathematical Software*, 13(2):113–132, June 1987.

[5] J. R. Rice and R. F. Boisvert. *Solving Elliptic Problems Using ELLPACK.* Springer-Verlag, New York, 1985.

[6] 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.*, 36:475–484, 1981.