1992

XELLPACK User Guide

The Parallel ELLPack Group

Report Number:
92-053

https://docs.lib.purdue.edu/cstech/974

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
XELLPACK User Guide

The Parallel ELLPACK Group

August 28, 1992
Abstract

This document describes how to solve Partial Differential Equations (PDEs) using the XELLPACK programming environment. XELLPACK is an interactive environment for defining and solving PDE problems, and a programming system for implementing new PDE solvers. The interactive environment provides comprehensive tools for specifying the PDE, the domain, the domain discretization and the solution method. XELLPACK also displays and analyzes solution and performance data. The XELLPACK library of solvers contains a large collection of solution modules, and the library can be extended to include user-built modules. In this way, new modules can be developed, implemented, and analyzed within XELLPACK.

XELLPACK is an extension of the well-known ELLPACK language. XELLPACK has extended the ELLPACK language and has enhanced the ELLPACK solution modules library. XELLPACK can solve PDEs using finite element methods in addition to finite difference methods. Finally, XELLPACK provides a user interface within which to build and solve PDE problems.
## Contents

1 Introduction
   1.1 The XELLPACK Approach ........................................... 5
   1.2 The XELLPACK System Structure .................................. 5

2 XELLPACK Software Architecture ............................... 7
   2.1 XELLPACK Session Control ...................................... 7
   2.2 Equation Specification ........................................... 8
   2.3 Domain Specification and Discretization ....................... 8
   2.4 Solution Scheme Specification .................................. 8
   2.5 PDE Program Processing ......................................... 8
   2.6 PDE Program Post-Processing ................................... 8
   2.7 XELLPACK Language and Language Processing ................. 9

3 XELLPACK : The Problem Specification ................... 12
   3.1 The XELLPACK Session ............................................ 12
      3.1.1 Session Control Window .................................. 12
      3.1.2 Session Template .......................................... 13
   3.2 Equation Specification .......................................... 15
      3.2.1 Symbolic Manipulation Tool ................................ 15
      3.2.2 Program Generation ......................................... 18
   3.3 Domain Specification and Discretization ..................... 21
      3.3.1 Boundary Tool ............................................... 22
      3.3.2 Grid Tool .................................................... 25
      3.3.3 Mesh Tool .................................................... 29
   3.4 Solution Scheme Specification .................................. 31

4 XELLPACK: Processing and Post-Processing ............... 34
   4.1 Problem Processing .............................................. 34
      4.1.1 Performance Data Collection ................................ 34
      4.1.2 Program Execution .......................................... 34
   4.2 Problem Post-Processing ......................................... 37
      4.2.1 Execution Tracing ........................................... 37
      4.2.2 3-D Solution and Data Visualization ....................... 39
      4.2.3 Performance Visualization Tool ............................... 41
      4.2.4 Performance Evaluation Tool ................................ 45

5 Closing Remarks .................................................. 50

6 APPENDIX ............................................................ 53
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Software Architecture of XELLPACK</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Sample XELLPACK Program</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>XELLPACK Session Control</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>Empty XELLPACK Session</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>Symbolic Manipulation Tool</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>Symbolic Tool With Information Windows</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>2-D Boundary Tool</td>
<td>23</td>
</tr>
<tr>
<td>8</td>
<td>The Completed Boundary</td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td>2-D Grid Tool With Information Windows</td>
<td>28</td>
</tr>
<tr>
<td>10</td>
<td>The Completed Mesh</td>
<td>30</td>
</tr>
<tr>
<td>11</td>
<td>Configuration Window</td>
<td>35</td>
</tr>
<tr>
<td>12</td>
<td>Execution Trace Window</td>
<td>36</td>
</tr>
<tr>
<td>13</td>
<td>Processing Performance Window</td>
<td>37</td>
</tr>
<tr>
<td>14</td>
<td>3-D Visualization Tool</td>
<td>41</td>
</tr>
<tr>
<td>15</td>
<td>Graph of Solution Function</td>
<td>42</td>
</tr>
<tr>
<td>16</td>
<td>Multiple Solution Graphs</td>
<td>42</td>
</tr>
<tr>
<td>17</td>
<td>Performance Analysis Tool</td>
<td>44</td>
</tr>
<tr>
<td>18</td>
<td>Graph of Performance Data</td>
<td>46</td>
</tr>
<tr>
<td>19</td>
<td>Performance Evaluation Tool</td>
<td>47</td>
</tr>
</tbody>
</table>
1 Introduction

XELLPACK is a programming environment for defining and solving partial differential equations (PDEs) defined on two or three dimensional domains. XELLPACK can solve a large population of PDE problems including linear and non-linear elliptic problems on arbitrary two-dimensional domains, linear elliptic problems on rectangular three-dimensional domains, systems of elliptic problems, and time dependent problems.

The XELLPACK system is based on the ELLPACK language and data structures [20]. XELLPACK provides a complete environment for building, solving and analyzing two-dimensional PDE problems. Three dimensional problems which are defined using the ELLPACK language can be loaded into XELLPACK for solution and analysis. XELLPACK solves PDE problems sequentially. Other versions of the ELLPACK-based systems allow the interactive specification of 3-dimensional problems, and the execution of ELLPACK language programs on parallel machines.

1.1 The XELLPACK Approach

The XELLPACK design is based on two principles. The first is to create a user-friendly environment for specifying and solving PDE problems, and the second is to provide a method for integrating new PDE solvers into the system for testing and analysis. Among the many areas of research that XELLPACK addresses are:

- The creation, analysis and implementation of new PDE solvers.
- The development of a high level problem specification language which is easy to use and allows efficient integration of numeric and symbolic processing.
- The creation of a uniform environment for obtaining software engineering measurements.
- The development of a data visualization system to view and analyze computational results.

1.2 The XELLPACK System Structure

XELLPACK consists of three major components: the XELLPACK user interface, the XELLPACK language and language processor, and the XELLPACK module library.

The XELLPACK interface provides a user-friendly way of defining the PDE problem. It contains tools for describing PDE equations, geometric domains, domain discretizations, solution requirements and solution paths. When the problem definition is complete, XELLPACK generates an XELLPACK language program using specific constructs that represent the PDE problem and its solution method.

The XELLPACK language processor converts an XELLPACK program into a Fortran program. The program is compiled and linked with the XELLPACK library modules specified in the solution path of the XELLPACK program. It is then executed on the target
machine. When program execution is complete, the solution and performance data can be visualized graphically.

The XELLPACK library consists of modules which are classified in a hierarchical structure, with fixed interfaces among adjacent levels. Modules from different levels can be mixed to form different solution paths based on the user's objective, precision requirements, and available resources.

The XELLPACK user interface is designed to provide a textual and graphical environment for the high level, interactive composition of programs which solve PDE problems using the ELLPACK system.

The XELLPACK user interface and the various tools it provides are the topic of this document. The XELLPACK source code, language and language processor are discussed in "The XELLPACK Programmer Guide" and the XELLPACK library is discussed in "The XELLPACK Library Guide."

The remainder of this document is organized as follows: In section 2, we discuss the software architecture of the XELLPACK system. In section 3, we give a detailed description of the tools within the XELLPACK system which are used to interactively define a PDE problem. The PDE and solution method specification make up the pre-processing system of XELLPACK. In section 4, the use of the processing and post-processing systems are described. Facilities for execution control, performance data collection, data analysis, and solution visualization are discussed.

The XELLPACK programming environment is implemented on a hardware facility consisting of graphics workstations supporting the X Window System.
2 XELLPACK Software Architecture

The software architecture of XELLPACK is based on a six step approach to defining and solving PDE problems. Each step in the process corresponds to a tool within the XELLPACK environment. The steps in this process are

1. Specify the PDE equation(s).
2. Define the domain and the boundary conditions.
3. Discretize the domain with a grid or mesh.
4. Discretize the PDE equations using finite difference or finite element methods.
5. Solve the resulting system of linear equations.
6. Display the solution and performance data.

The PDE problem specification and solution occur within an XELLPACK session. The session contains the tools which allow users to describe the problem, specify the solution path, generate and execute an XELLPACK program, and view the results. Based on the above steps, the software architecture of XELLPACK consists of

- XELLPACK Session control.
- Equation specification.
- Domain specification and discretization.
- Solution scheme specification.
- PDE problem processing.
- PDE problem post-processing.
- XELLPACK language and language processing.

2.1 XELLPACK Session Control

The session control is the top-most level of the XELLPACK system. Session control functions include creating new XELLPACK programs, opening and modifying existing XELLPACK programs, specifying program execution options, and selecting target machines. The first two functions initiate XELLPACK program sessions; the other functions control program processing. Multiple sessions are allowed, and sessions are independent of each other.

The session provides a text window called the session template and an emacs-like text editor. You can specify the problem textually within the session template using the XELLPACK language. The session also provides tools for defining equations, performing symbolic manipulation, drawing domains, discretizing domains, specifying solution methods, and executing programs. The results of these tools are saved as text into the session template using the XELLPACK language format. The tools can also "read" the contents of the session template, and graphically display the textual representation of the problem.
2.2 Equation Specification

PDE equation are specified within XELLPACK's symbolic manipulation tool. The symbolic manipulation tool can be used to derive or linearize PDE equations based on the mathematical specification of the problem.

2.3 Domain Specification and Discretization.

The two-dimensional geometry specification tool allows you to build two-dimensional domains using the workstation mouse. You draw the boundary in pieces by specifying control points, which are then used by the tool to parametrically represent each piece. You can modify previously defined control points and boundaries. Domains with holes and arcs are supported. Boundary conditions can be assigned interactively to each piece of the boundary.

You can discretize the domain either with a grid or with a mesh. Grids can be uniform or non-uniform, and are specified using the Grid Tool. The Mesh Tool generates orthogonal and triangular meshes automatically.

2.4 Solution Scheme Specification.

In order to specify a solution path, you must choose a sequence of solution algorithms from a set of menus. The menus list the modules names for the solution modules contained in the XELLPACK library. There are separate menus for the operator discretization modules, equation indexing modules and linear system solvers.

Several foreign systems have been integrated into XELLPACK. The foreign system interface allows you to access the foreign system solvers as XELLPACK solution modules. These modules are also listed in the solution path menus.

The XELLPACK foreign system interface acts as a bridge between the XELLPACK system and other numerical software systems. This allows other systems to take advantage of the programming environment of XELLPACK, and allows XELLPACK users to access the specialized problem solving techniques that are supported by the foreign systems.

2.5 PDE Program Processing

The XELLPACK processing includes program execution and performance data collection. Performance data is collected when specified as an option in the XELLPACK program. Each module is monitored for time spent during initialization, setup and computation.

2.6 PDE Program Post-Processing

The post-processing system consists of a performance analysis tool and a solution visualization tool.
The performance analysis tool provides various statistical methods for viewing and analyzing performance data. This tool is very flexible, and allows you to interactively design your analysis methods.

There is also a Performance Evaluation System (PES) that allows you to compare the performance of solution modules under varying conditions, domains or parameters.

The solution visualization tool displays the solution data and its derivatives for two-dimensional domains, and slices of the solution and its derivatives for three-dimensional domains. When a finite difference method is used, the visualization tool displays the solution on any user-selected grid, in addition to the original grid on which the problem was solved. The solution is interpolated for grid points that vary from the original. When a finite element method is used, the visualization tool displays the solution either on the original mesh, or on any user-selected grid.

2.7 XELLPACK Language and Language Processing

The XELLPACK language allows you to specify PDE equations and boundary conditions using mathematical notation, to define the boundary pieces of the PDE domain by a piecewise parametric representation, to determine the solution path and its parameters using reserved keywords, and to specify the output format for the PDE solution and its derivatives. The XELLPACK language is an extension of the ELLPACK language [20], and Appendix D contains a syntax definition of the XELLPACK extensions for reference. A sample XELLPACK program is listed in figure 2.

The remainder of this document describes each tool within the XELLPACK programming environment.
Figure 1: Software Architecture of XELLPACK
FLOW AROUND A CYLINDER

** FLOW AROUND A CYLINDER

* u - stream function
* uy - velocity in y direction
* -ux - velocity in x direction

OPTIONS.

time $ memory $ clockwised = .true. $ level = 1 $ xplot3d

* BOUNDARY DEFINITION DATA GENERATED BY XELLPACK

** BOUNDARY DEFINITION DATA GENERATED BY XELLPACK

DECLARATIONS + both + boundary.

common / plotx / xc(6,5)
common / ploty / yc(6,5)

FORTRAN + both + boundary.

data (xc(l,j),j=l,5) / 0.00,0.00,0.0,0.0,0.0 /
data (xc(2,j),j=l,5) / 0.00,8.00,0.0,0.0,0.0 /
data (xc(3,j),j=l,5) / 8.00,8.00,0.0,0.0,0.0 /
data (xc(4,j),j=l,5) / 8.00,6.43,0.0,0.0,0.0 /
data (xc(5,j),j=l,5) / 6.43,5.63,4.00,2.41,1.61 /
data (xc(6,j),j=l,5) / 8.00,6.43,0.0,0.0,0.0 /
data (yc(l,j),j=l,5) / 0.00,2.00,0.0,0.0,0.0 /
data (yc(2,j),j=l,5) / 2.00,2.00,0.0,0.0,0.0 /
data (yc(3,j),j=l,5) / 2.00,0.00,0.0,0.0,0.0 /
data (yc(4,j),j=l,5) / 0.00,0.00,0.0,0.0,0.0 /
data (yc(5,j),j=l,5) / 0.00,0.80,1.40,0.80,0.00 /
data (yc(6,j),j=l,5) / 0.00,0.00,0.0,0.0,0.0 /

SPECIFICATION OF MATHEMATICAL MODEL

EQUATION.

uxx + uyy = 0.

SPECIFICATION OF GEOMETRIC MODEL

BOUNDARY +par.

U = true(x,y) ON X = paramx(1,t,2,6), Y = paramy(1,t,2,6) &
FOR t = 0.00 TO 1.00
U = true(x,y) ON X = paramx(2,t,2,6), Y = paramy(2,t,2,6) &
FOR t = 0.00 TO 1.00
U = true(x,y) ON X = paramx(3,t,2,6), Y = paramy(3,t,2,6) &
FOR t = 0.00 TO 1.00
U = true(x,y) ON X = paramx(4,t,2,6), Y = paramy(4,t,2,6) &
FOR t = 0.00 TO 1.00
U = true(x,y) ON X = paramx(5,t,5,6), Y = paramy(5,t,5,6) &
FOR t = 0.00 TO 1.00
U = true(x,y) ON X = paramx(6,t,2,6), Y = paramy(6,t,2,6) &
FOR t = 0.00 TO 1.00

SPECIFICATION OF GEOMETRY DISCRETIZATION

GRID.

81 x points 0.00 to 8.00
21 y points 0.00 to 2.00

SPECIFICATION OF SOLUTION METHODS

DISCRETIZATION.

5 point star

SOLUTION.

jacobi SI (zeta=0.005,itmax=1000)

END.

Figure 2: Sample XELLPACK Program
3 XELLPACK: The Problem Specification

3.1 The XELLPACK Session

In this section we will describe how to use the XELLPACK pre-processing system to define a 2-dimensional PDE problem and specify the solution methods. PDE problems are defined by using a sequence of tools within an XELLPACK session. We will describe the function and operation of each tool and then illustrate the use of the tool by a sample problem. If you follow the sample session presented in this section, you will be able to build a complete XELLPACK program.

The sample problem is as follows:

Operator:

\[ u_{xx} + u^2 + u_{yy} = e^u + f(x, y) \]

where \( f(x, y) \) is chosen to make this PDE satisfy \( u(x, y) = x^2 + y^2 \)

Domain: Shown in Figure 8.

Boundary Conditions:

\[ u(x, y) = \text{true}(x, y) \equiv x^2 + y^2. \]

Before running XELLPACK, you must set the UNIX environment variable `PELLPACK` to the root directory that contains the XELLPACK system. In most situations, this is `/usr/pellpack`. If your site has a different setup, please contact your system programmer for the proper directory. As with all X Window System applications, it is necessary to set the UNIX environment variable `DISPLAY`, and load in the proper X resources in order for XELLPACK to work properly. See the `elltool` man-page for more details.

3.1.1 Session Control Window

XELLPACK is invoked by typing

`elltool`

This command displays the top level window, which is shown in Figure 3.

The top level of XELLPACK initiates sessions. A session provides all the functionality you need to specify and solve a PDE problem. The XELLPACK top level allows you to run multiple sessions concurrently, and the sessions are independent of one another.

The first button of the top level is the Dimension button. You can choose a 2-dimensional session or a 3-dimensional session. Currently, XELLPACK supports the interactive definition of a 2-dimensional PDE problem. If you choose to invoke a 3-dimensional session,
you can only load in and execute an XELLPACK language program. Future versions of XELLPACK will fully support the interactive definition of 3-dimensional problems.

The next two buttons of the top level window start up sessions. The *Load* button starts the session with an existing file which can then be modified or run. The *New File* button starts a new session. The last button is the *Configuration* button. It is used to select the target machine for XELLPACK program execution, and to set execution options. The *Configuration* button is discussed in Section 4.1.

We will specify and solve a two-dimensional PDE problem. The two-dimensional session is the XELLPACK default, so it is not necessary to select it. Since we will build a complete program, click the left mouse button in *New File* to bring up a new session window.

### 3.1.2 Session Template

At the top of the session window is a command panel. This panel contains buttons which invoke the tools used to build an XELLPACK program. The session window below the panel is the session *template*. The template displays the current textual version of the XELLPACK program which has been generated by the tools from the problem specification. You can edit this text directly using an *emacs*-like text editor.

Here is a synopsis of the the buttons in the two-dimensional session panel from left to right:

- *Quit*: Quit the session.
- *Save*: Save the XELLPACK program in a file.
- *Save As*: Save the XELLPACK program in the specified file.
- *Symbolic Tool*: Invoke the symbolic manipulation tool.
- *Boundary Tool*: Invoke the boundary definition tool.
Figure 4: Empty XELLPACK Session
3 XELLPACK: THE PROBLEM SPECIFICATION

- **Grid Tool**: Invoke the grid definition tool.
- **Mesh Tool**: Invoke the mesh generation tool.
- **Discretization Menu**: Display the operator discretization modules.
- **Indexing Menu**: Display the linear system indexing modules.
- **Solution Menu**: Display the linear system solution modules.
- **Run**: Execute the XELLPACK program in the session template.

3.2 Equation Specification

The XELLPACK problem specification begins with the specification of the PDE operator. You can type the equation in the session template directly using the editor, or you can invoke the symbolic tool to manipulate the equations symbolically.

3.2.1 Symbolic Manipulation Tool

The symbolic manipulation tool allows you to enter PDEs explicitly, or to manipulate PDE equations analytically. Its tasks include linearizing nonlinear PDE equations, discretizing time dependent PDEs, computing expressions for the right hand sides, and performing other symbolic operations. It is implemented as a front-end to MAXIMA. It possesses the full power of MAXIMA plus some special operations tailored for XELLPACK problem specification.

To invoke the symbolic tool, click the left mouse button on the third button from the left in the session command panel. The XELLPACK-Maxima Interface window is displayed. This is the tool that is used to specify the PDE operator. The tool allows you to specify linear, nonlinear and time dependent problems without any restrictions.

At the top of the symbolic tool window is a command panel. This panel contains the buttons which control the operation of the tool. Below the panel are two windows. The top window allows you to enter information about the PDE into the entry fields and pop-up windows associated with the buttons. The bottom window is connected to MAXIMA, and can be used by experienced MAXIMA users to manipulate equations. The information that MAXIMA produces can be cut and pasted into the entry fields in the top window.

Here is a short synopsis of the buttons in the command panel of the symbolic tool:

- **File**: Display the File pull-down menu. The menu contains these options:
  - **View**: View the current version of the XELLPACK program which has been generated from the problem specification. Close the view window by clicking the left mouse button on Close.
  - **Save As...**: Save the current XELLPACK program in the specified file.
  - **Quit**: Close the symbolic tool.
Figure 5: Symbolic Manipulation Tool
• **Maxima:** Display the Maxima pull-down menu. The menu contains these options:
  - **Start:** Start Maxima as a subprocess. Note that Maxima is automatically invoked when you enter the symbolic tool.
  - **Stop:** Quit the Maxima sub-process.

• **Boundary:** Specify the boundary and boundary conditions for two- and three- dimensional rectangular domains. This option is used only for XELLPACK/FIDISOL problems.

• **Help:** Display Help Text.

We first select the **Mode of Operation** which governs how the PDE operators will be written to the session template. XELLPACK has two options: **PELLPACK** and **FIDISOL**. The default mode of operation is **PELLPACK**, and we will restrict our discussion in this section to the pellpack mode. FIDISOL [22] is a foreign system which has been integrated into XELLPACK, and a discussion of FIDISOL and its XELLPACK interface can be found in Appendix B.

We must specify the number of operators in the entry field labeled **Number of Operators**. The sample problem has only one operator, and this is the default value. For systems of equations, you must enter the number of equations in your system in this field first, and then enter each equation in turn in the fields provided by the **PDE Operator Editor**. Each field in the operator editor is numbered at the left to identify which equation is expected. See Appendix B for an example of entering multiple equations.

Click the left mouse button in the entry field labeled 1 in the **PDE Operator Editor** to enter our single PDE. Type the equation as follows:

\[ u_{xx} + u^2 u_{yy} = \exp(u) \]

We intend to force a solution, so click the left mouse button in the button labeled **Forcing Solution Information**. A window is displayed which requests the forcing solution for Operator 1. Click the left mouse button in the entry field and type the solution as follows:

\[ x^2 + y^2 \]

and then click on the **OK** button to proceed.

The PDE is non-linear since the coefficient of \( u_{yy} \) is a function of \( u \). Therefore, we will specify parameters which control the linearization of the PDE. Click the left mouse button on the button labeled **Linearization Information**. You can specify **Tolerance**, **Norm**
to Use, Maximum Number of Iterations, and Initial Guess in the entry fields provided. For a complete description of the function of each of these fields, see [21].

For our problem, we modify the Tolerance value by clicking the left mouse in the entry field labeled Tolerance and typing 0.05. Similarly, click the left mouse button in the Initial Guess entry field and set it to 0. Accept the modifications and continue by clicking the left mouse button in the OK button at the bottom of the window.

If the problem were time-dependent, we could specify the parameters which control the time discretization of the PDE. To view these parameters, click the left mouse button in the button labeled Time Discretization Information. For a complete description of each of these fields, see [21]. Click in the Cancel button to dismiss the window.

3.2.2 Program Generation

After the PDE equation is specified, the symbolic tool can be used to generate a complete XELLPACK program, with default values for the domain and the solution scheme. This program will be placed in the session template, and the XELLPACK processor can then be used to solve the default problem.

Generate a program by clicking the left mouse button in the Generate Program button. The tool responds by filling the session template with the default program. Note that the program is separated into segments which define the pieces of the problem definition, and correspond to the tools which help you specify them.

In the EQUATION segment is the PDE you just defined, along with the force term computed by the Symbolic Tool. Look at the SUBPROGRAM segment and find the FORCE function which produced the forcing term required by the solution you specified. The FORTRAN segments generated by the Symbolic Tool are required for solving non-linear problems like the one in our example.

The rest of the template is filled with default specifications for the remaining segments required by an XELLPACK program. Updates to the segments will result from using the other tools in the session command panel. The template can also be edited by typing directly into the session template.

It is possible to solve the problem now, since all segments either have been specified or have default information. However, we will also specify a domain, mesh and solution path before solving the PDE. The solution path can be specified within the symbolic tool using the Solution Parameters pull-down menus for Discretization, Indexing and Linear Solver, but we will defer the discussion of the solution path until section 3.4.

Close the symbolic tool by using the left mouse button to select Quit from the File pull-down menu. The program generated by the above problem is given below.
Figure 6: Symbolic Tool With Information Windows
time = .true.
clockwise = .true.
xplot3d

DEclarations.
real tol
integer niters

Equation.
UXX+U(X,Y)**2+UYY+(2*U(X,Y)+UYY(X,Y)-E**U(X,Y)+LOG(E)*U = 2*U(X,Y & 
)***2+UYY(X,Y) - E**U(X,Y) + LOG(E)*U(X,Y) + FORCE(X,Y) + E**U(X,Y)

Boundary.
u_true(x,y) on x=-1.0, y=t for t=-1.0 to 1.0
u_true(x,y) on x=t, y=1.0 for t=-1.0 to 1.0
u_true(x,y) on x=1.0, y=1.0-t for t=0.0 to 2.0
u_true(x,y) on x=1.0-t, y=-1.0 for t=0.0 to 2.0

Grid.
20 x points -1.0 to 1.0
20 y points -1.0 to 1.0

Triple.
set (u = zero)

Fortran.
niters = 10
tol = 0.050000000000000003
do 300 i = 1, niters

Discretization.
5-Point Star

Indexing.
As Is

Solution.
Band GE

Output.
max (error)

Fortran.
illevl = 0

Fortran.
* test for convergence
if (R1R1 .lt. tol) then
  go to 301
endif
300 continue
print *, 'failed to converge!'
go to 302
301 continue
print *, 'converged in ', i, ' iterations.'
302 continue

Subprograms. *both.

function FORCE(x,y)
real tol
integer niters
FORCE = 2*Y**4+4*Y**2*2+2*Y**4-E**Y*(Y**2*2+2)**2+2
return
The above XELLPACK program contains all of the segments required by the XELLPACK program processor. Standard XELLPACK program segments and their definitions are listed below:

- **EQUATION**: Defines the PDE operator.
- **BOUNDARY**: Defines the domain and boundary conditions.
- **GRID**: Defines how to discretize the continuous domain for finite difference methods.
- **MESH**: Defines how to discretize the continuous domain for finite element methods.
- **DISCRETIZATION**: Specifies how to discretize the operator.
- **INDEXING**: Specifies how to reorder the algebraic equations prior to running the solver.
- **SOLUTION**: Specifies how to solve the system of linear equations.
- **OUTPUT**: Specifies what quantities are to be output.
- **SUBPROGRAMS**: Adds FORTRAN user code to the generated control program as functions or subroutines.
- **FORTRAN**: Adds FORTRAN user code to the generated control program between XELLPACK segments.

The rest of our problem specification process involves changing the default segment values which have been generated by the Symbolic Tool for the BOUNDARY, GRID, DISCRETIZATION, INDEXING, SOLUTION, and OUTPUT segments, as well as adding a MESH segment.

### 3.3 Domain Specification and Discretization

The XELLPACK environment provides a two-dimensional domain specification tool for drawing the boundary graphically and specifying boundary conditions. The domain discretization tools include the Grid Tool and the Mesh Tool. These are tools which allow you to discretize the domain graphically. We will demonstrate the use of the grid tool, but we will use the mesh tool to discretize the domain in our example.
3.3.1 Boundary Tool

Click the left mouse button in the fourth icon from the left to invoke the Boundary Tool. Since the Symbolic Tool has parameterized the rectangle using functions other than the Boundary Tool's internally defined parametric functions, the Boundary Tool must do some extra work to understand the boundary definition. A special Log Window is displayed, and it contains a trace of XELLPACK execution steps required for domains which are built outside of the Boundary Tool. Since ELLPACK allows you to define your domain using any function, whether intrinsically fortran or user-built, XELLPACK will dynamically compile and link those routines. Thus, your domain can be displayed with the Boundary Tool even if it is not built within the tool. After the boundary definition is read into the Boundary Tool, the boundary window is displayed.

At the top of the window is a command panel. This panel contains buttons which control the operation of the boundary tool. At the bottom right of the command panel are two fields, XPos and YPos. They display the current X and Y coordinates of the mouse pointer. These values can be used to position the pointer for precision drawing. The window below the command panel is used to construct the domain for our PDE.

Here is a short synopsis of the buttons in the command panel of the boundary tool.

- **Quit**: Close the boundary tool.
- **Save**: Save the current boundary information into the template.
- **Clear**: Clear the entire work area.
- **Delete**: Delete the last object drawn from the work area.
- **Set Conditions**: Set the boundary conditions for each piece of the boundary.
- **New Boundary**: Start a new boundary.
- **New Hole**: Add a hole to an existing boundary. Any number of holes can be added to the boundary. The holes must be added in the opposite direction of the outer domain boundary. Thus, if the domain is built in the clockwise direction, each hole must be built in the counterclockwise direction.
- **Add CP**: Add a control point to modify a boundary piece.
- **Delete CP**: Delete a control point from a boundary piece.
- **Set Range**: Set the \((x, y)\) range for drawing the boundary.
- **Set Lines**: Reset the \((x, y)\) lines as an aid to drawing the domain.
- **Show Control Points**: Turn the display of control points ON/OFF.
- **Bernstein**: Approximate the boundary pieces by Bernstein polynomials, using the control points as knots.
Figure 7: 2-D Boundary Tool
• **Interpolation:** Interpolate to approximate the boundary pieces between control points.

Curves are generated by specifying control points. Two curve generation algorithms are supported, *Bernstein* and *Interpolation*. Both algorithms use the control points to construct a curve. We will use the default *Bernstein* algorithm to build the domain.

The boundary of the domain is drawn piece by piece using the mouse buttons. The three buttons perform the following tasks when drawing:

- Click the left mouse button in the work buffer to place a control point.
- Click the middle mouse button to end a boundary piece and display it.
- Click the right mouse button to add the last control point in the boundary. It will end the boundary definition by connecting the last boundary piece to the first control point on the first boundary piece. Thus the first control point on the first piece will also be the last defined control point for the last boundary piece.

For our example, we will draw a domain that has eight boundary pieces forming an octagon, with two boundary pieces in the interior forming a hole. First, click the left mouse button in the *Clear* button to clear the work area. Then click in the *New Boundary* button to initiate drawing. This de-activates the top row of buttons and prepares for the specification of a new boundary. The *XPos* and *YPos* position counters in the command panel indicate the current position of the sprite within the drawing area.

To begin construction of the example domain, move the sprite until XPos=-1.00 and YPos=0.00 (approximately) appear in the counter, then click the left mouse button. This places a control point, indicated by a black dot, at that position in the graph. Move the sprite to XPos=-0.65 and YPos=0.65. Click the middle mouse button to complete the first boundary piece. Now move the sprite to XPos=0.00 and YPos=1.00 and click the middle mouse button to complete the second boundary piece. Similarly, place the next five sides of the octagon to roughly correspond with the domain in Figure 8. After the seventh side has been placed, click the right mouse button to attach the seventh side to the first boundary point. This closes the domain and completes the final side of the octagon.

We must now add a hole to the interior. Click the left mouse button on the *New Hole* button in the command panel. Holes are added in the counterclockwise direction. Place control points at (-0.50,0.00), (-0.50,-0.60) and (0.50,-0.60) using the left mouse button. Then place the sprite at (0.50,0.00) and click the middle mouse button to place the boundary piece for one half of the interior circle. Repeat the process to place the other half.

The next task is to specify the boundary conditions. Boundary conditions are specified on each boundary piece. Click the left mouse button on the *Set Conditions* button to pop up the boundary conditions dialog. The boundary pieces are numbered clockwise
starting at the first piece (defined by the first control point). The default conditions are \( u(x, y) = \text{true}(x, y) \) on the boundary. Different boundary conditions can be set by editing the expressions in the entry fields for each piece of the boundary.

The default conditions are those required by our example, so it is not necessary to change them. Click the left mouse button in the Continue button of the dialog to dismiss the dialog. Save the boundary in the template of the session window by clicking the left mouse button in the Save button of the boundary tool command panel. *Do not close the Boundary Tool.* It must remain open to access the domain discretization tools.

You can also use the Boundary Tool to display domains that you have parameterized on your own. For example, if you have entered the following parameterization in the template's BOUNDARY and HOLE segments:

\[
\text{BOUNDARY.} \\
\quad u = 0. \quad \text{on} \quad x = \cosh(3.0) \sin(t), \quad y = \sinh(3.0) \cos(\pi t) \quad \& \\
\quad \text{for} \quad t = 0.0 \text{ to } 2
\]

\[
\text{HOLE.} \\
\quad u = 1. \quad \text{on} \quad x = \cosh(2.3) \sin(t), \quad y = -\sinh(2.3) \cos(\pi t) \quad \& \\
\quad \text{for} \quad t = 0.0 \text{ to } 2
\]

two confocal ellipses will be displayed by the Boundary Tool. The constant \( \pi \) is a defined keyword in the ELLPACK language.

The Boundary Tool window must remain open to access the Grid and Mesh discretization tools; however, you can iconify the Boundary window to avoid overcrowding on your workstation screen. Note also that when the Boundary window is open, you cannot edit the session template. This is to ensure that the Boundary, Grid and Mesh Tools correspond to each other and to the contents of the session template.

### 3.3.2 Grid Tool

You use the Grid Tool to discretize a domain when the solution involves finite difference methods. When no grid segment is present in the session template, the default grid is \( 10 \times 10 \). The maximum grid size is set in the Configuration window. You must change the default maximum if you want to specify a grid larger than \( 50 \times 50 \).

The Symbolic Tool has generated the following grid segment for us:

\[
\text{GRID.} \\
\quad 20 \ x \ points \ -1.0 \ to \ 1.0 \\
\quad 20 \ y \ points \ -1.0 \ to \ 1.0
\]
Figure 8: The Completed Boundary
Assume we want to change the grid to a 25 x 30 grid on the rectangular domain from -1.0 to 1.0. The new grid segment would appear as follows:

\[
\text{GRID.}
\]

25 x points -1.0 to 1.0
30 y points -1.0 to 1.0

This change could easily be made by editing the template directly (Remember that you need to close the Boundary window to edit the template). However, we will demonstrate the operation of the grid tool by modifying the grid using the tool. Invoke the grid tool by clicking the left mouse button in the fifth icon from the left. The Grid Tool window is displayed.

At the top of the grid tool window is a command panel. This panel contains buttons which control the operation of the tool. The window below the command panel is used to construct and display the grid.

The domain and the grid which are defined in their respective segments, are displayed upon entry to the tool. Note that the grid segment contents override the domain definition when you are in the Grid Tool. Only the part of the domain within the grid range is displayed in the grid window. So if, for example, the grid range defined in the grid segment contains none of the domain as defined in the boundary segment, the grid range is displayed without the domain.

Here is a short synopsis of the buttons in the command panel of the grid tool.

- **Quit**: Close the grid editor.
- **Set Range**: Set the \((x, y)\) grid range.
- **Save**: Save the grid in the template of the editing session window.
- **Mode: X/Y**: Toggle between editing in the vertical (Y) and horizontal (X) directions.
- **Uniform Grid**: Set a uniform grid in the direction specified by the current Mode.

You can use this tool to set a non-uniform grid. Add grid lines by moving the sprite to the proper position and clicking the left mouse button; delete lines by clicking the middle mouse button while the sprite is positioned on the grid line to be deleted; move grid lines by holding down the right mouse button to grab a line and then moving the mouse to the new grid line location.

You can also request a uniform grid. Click the left mouse button in the Uniform Grid button to pop up the dialog. Since the X mode is set upon entry, the dialog applies to the
Figure 9: 2-D Grid Tool With Information Windows
horizontal direction. To change the number of grid lines to 25, type 25 in the entry field and either press RETURN or click the left mouse button in the Continue button.

To modify the number of grid lines in the vertical direction, click the left mouse button in the Mode: X button. Then follow the above procedure to change the number of grid lines for the Y mode. After all modifications have been made, click the left mouse button in the Save button to save the domain discretization into the grid segment of the session template. Click in the Quit button to close the grid tool.

We have modified the original grid, but our grid discretization will be ignored when the mesh segment is added to the session template. We now define the mesh discretization.

3.3.3 Mesh Tool

We have specified the PDE operator along with its domain and boundary conditions. The next step is to define the discretization of the domain using the mesh editing tool.

Click the left mouse button on the fifth button from the left to invoke the Mesh Tool. A dialog box is displayed which requests information concerning the type of mesh to be used. The dialog box is shown below.

You can choose either an Orthogonal mesh discretization or a Triangular mesh discretization. The default is orthogonal, and if the user selects orthogonal, no further information is required to proceed. We will select triangular, and must provide two additional pieces of information, Mesh Type and Mesh Length.

The mesh type sets the type of triangle which will make up the mesh. There are two choices, \(d < 1 < 1.5d\) and \(d < 1 < 2d\). If the first choice is specified, the mesh consists of triangles which are nearly equilateral. The second choice places triangles in which the angles are not required to be close to 60 degrees. The default is the first choice, and we retain the default. The mesh length sets the length, in inches, of the edge of a triangle. Again, we retain the default value of 0.10.

Click the left mouse button in the Continue button, and the mesh editing tool window is displayed. The graphical representation of the tool is given in Figure 10.
Figure 10: The Completed Mesh
There are two buttons in the command panel of the tool. Here is a short synopsis of each:

- **Quit**: Close the mesh editor.
- **Save**: The data describing the mesh discretization is saved in a file, and the filename is written to the template, along with other mesh parameters. This button brings up a filename selection window so that you can choose a name for the mesh information file.

Click the left mouse button in the *Save* button of the command panel. A dialog requesting a filename for the mesh data is displayed. Type *data* into the filename entry field and press *RETURN*. Since no directory path has been specified, the file is saved into the current directory. Click the left mouse button in the *Quit* button to close the mesh tool. Since the domain and its discretization are both defined, you can now close the Boundary Tool. Click in the *Quit* button of the Boundary Tool command panel.

### 3.4 Solution Scheme Specification

We have completely defined the PDE problem: the PDE operator, the domain, the boundary conditions, and the domain discretization scheme. We must now specify the solution method to use in solving the problem.

You choose the solution path for PDE problems by selecting a sequence of modules that discretize the operator, reorder the resulting algebraic equations or unknowns, and solve the resulting linear system. With XELLPACK, determining the solution path means selecting the solver modules that best fit the specified problem.

You specify the solution path by selecting modules from the drop-down menus that correspond to the three command panel buttons: *DISC MENU*, *INDX MENU* and *SOLN MENU* in the session window.

To select the discretization module, press and hold the left mouse button on the *DISC MENU* button. A menu containing the list of available modules is displayed. We will choose the *Bi-Linear FEM* method for solving our PDE on the triangular mesh. You select this module from the menu by moving the mouse while holding down the left mouse button to the "Bilinear FEM" entry, then release the left mouse button. The new discretization method will replace the current one in the discretization segment of the session template.

You select the indexing and solution methods in the same way. In our example, we choose "As is" for indexing and "Jacobi SI" for the algebraic equations solver. After doing so, a complete XELLPACK program has been generated according to our specifications. The program is shown below.
OPTIONS.
  litime = .true.
  clockwise = .true.
  xplot3d

DECLARATIONS. +both. +boundary.
  common / plotx / xc(10, 4)
  common / ploty / yc(10, 4)

FORTRAN. +both. +boundary.
  data (xc(1,j),j=1,4) / -0.99,-0.47,0.0,0.0 /
  data (xc(2,j),j=1,4) / -0.47,0.00,0.0,0.0 /
  data (xc(3,j),j=1,4) / 0.00,0.71,0.0,0.0 /
  data (xc(4,j),j=1,4) / 0.71,1.00,0.0,0.0 /
  data (xc(5,j),j=1,4) / 1.00,0.69,0.0,0.0 /
  data (xc(6,j),j=1,4) / 0.69,0.00,0.0,0.0 /
  data (xc(7,j),j=1,4) / 0.00,-0.71,0.0,0.0 /
  data (xc(8,j),j=1,4) / -0.71,-0.99,0.0,0.0 /
  data (xc(9,j),j=1,4) / -0.50,-0.50,0.51,0.50 /
  data (xc(10,j),j=1,4) / 0.50,0.49,-0.45,-0.50 /
  data (yc(1,j),j=1,4) / 0.00,0.71,0.0,0.0 /
  data (yc(2,j),j=1,4) / 0.71,1.00,0.0,0.0 /
  data (yc(3,j),j=1,4) / 1.00,0.71,0.0,0.0 /
  data (yc(4,j),j=1,4) / 0.71,0.00,0.0,0.0 /
  data (yc(5,j),j=1,4) / 0.00,-0.70,0.0,0.0 /
  data (yc(6,j),j=1,4) / -0.70,-1.00,0.0,0.0 /
  data (yc(7,j),j=1,4) / -1.00,-0.70,0.0,0.0 /
  data (yc(8,j),j=1,4) / -0.70,0.00,0.0,0.0 /
  data (yc(9,j),j=1,4) / 0.00,-0.60,-0.60,0.01 /
  data (yc(10,j),j=1,4) / 0.01,0.61,0.61,0.00 /

DECLARATIONS.
  real tol
  integer niters

EQUATION.
  \( U(X,Y) = 2 \times U(Y) \times (U(X,Y) + \exp(U(X,Y))) \times \frac{1}{(1-U(X,Y)) + \exp(U(X,Y)) + \text{FORCE}(X,Y)} \)

BOUNDARY. +parm.
  \( U = \text{true}(x,y) \quad \text{ON} \quad X = \text{paramx}(1,t,2,10), \quad Y = \text{paramy}(1,t,2,10) \quad \& \quad \text{FOR} \quad t = 0.00 \quad \text{TO} \quad 1.00 \)
  \( U = \text{true}(x,y) \quad \text{ON} \quad X = \text{paramx}(2,t,2,10), \quad Y = \text{paramy}(2,t,2,10) \quad \& \quad \text{FOR} \quad t = 0.00 \quad \text{TO} \quad 1.00 \)
  \( U = \text{true}(x,y) \quad \text{ON} \quad X = \text{paramx}(3,t,2,10), \quad Y = \text{paramy}(3,t,2,10) \quad \& \quad \text{FOR} \quad t = 0.00 \quad \text{TO} \quad 1.00 \)
  \( U = \text{true}(x,y) \quad \text{ON} \quad X = \text{paramx}(4,t,2,10), \quad Y = \text{paramy}(4,t,2,10) \quad \& \quad \text{FOR} \quad t = 0.00 \quad \text{TO} \quad 1.00 \)
  \( U = \text{true}(x,y) \quad \text{ON} \quad X = \text{paramx}(5,t,2,10), \quad Y = \text{paramy}(5,t,2,10) \quad \& \quad \text{FOR} \quad t = 0.00 \quad \text{TO} \quad 1.00 \)
  \( U = \text{true}(x,y) \quad \text{ON} \quad X = \text{paramx}(6,t,2,10), \quad Y = \text{paramy}(6,t,2,10) \quad \& \quad \text{FOR} \quad t = 0.00 \quad \text{TO} \quad 1.00 \)
  \( U = \text{true}(x,y) \quad \text{ON} \quad X = \text{paramx}(7,t,2,10), \quad Y = \text{paramy}(7,t,2,10) \quad \& \quad \text{FOR} \quad t = 0.00 \quad \text{TO} \quad 1.00 \)
  \( U = \text{true}(x,y) \quad \text{ON} \quad X = \text{paramx}(8,t,2,10), \quad Y = \text{paramy}(8,t,2,10) \quad \& \quad \text{FOR} \quad t = 0.00 \quad \text{TO} \quad 1.00 \)

HOLE. +parm.
  \( U = \text{true}(x,y) \quad \text{ON} \quad X = \text{paramx}(9,t,4,10), \quad Y = \text{paramy}(9,t,4,10) \quad \& \quad \text{FOR} \quad t = 0.00 \quad \text{TO} \quad 1.00 \)
  \( U = \text{true}(x,y) \quad \text{ON} \quad X = \text{paramx}(10,t,4,10), \quad Y = \text{paramy}(10,t,4,10) \quad \& \quad \text{FOR} \quad t = 0.00 \quad \text{TO} \quad 1.00 \)
mesh.

read fem from file &
  (filename="/arthur/u1/acc/problem2.mesh", &
   iimnp=166, iimeln=258, iimtyp=0, iimnen=3)

TRIPLE.
set (u = zero)

FORTRAN.
  niter = 10
  tol = 0.050000000000000003
  do 300 i = 1, niter

  DISCRETIZATION.
  Bi-linear FEM

  INDEXING.
  As is

  SOLUTION.
  Jacobi SI

  OUTPUT.
  max (error)

FORTRAN.
  illevl = 0

FORTRAN.
  * test for convergence
  if (R1RM1 .lt. tol) then
    go to 301
  endif

  300 continue
  print *, 'failed to converge!'
  go to 302

  301 continue
  print *, 'converged in ', i, ' iterations.'
  302 continue

SUBPROGRAMS. *both.

function FORCE(x,y)
  real tol
  integer niter
  FORCE = -EXP(Y**2+X**2)+2*Y**4+4*X**2*Y**2+2*X**4+2
  return
end

function true(x,y)
  real tol
  integer niter
  TRUE = Y**2*X**2
  return
end

END.
4 XELLPACK: Processing and Post-Processing

4.1 Problem Processing

XELLPACK program processing provides performance data collection, error tracing, and program monitoring and convergence information.

4.1.1 Performance Data Collection

The XELLPACK facility for performance data collection provides performance measurements for a single solution algorithm, and performance comparisons for multiple solution algorithms. The timing facility follows ELLPACK conventions, but provides utilities for collecting computation data. Performance data can be obtained at different granularity levels ranging from the whole program to individual modules to arbitrary program blocks.

The code for collecting performance data is generated by the XELLPACK language processor and is done automatically when the XELLPACK option `iltime` is set to a non-zero value. This option can be placed anywhere in the XELLPACK program, and performance data collection will be started after the option is set.

```plaintext
option.
    itime = 1
```

The above option is equivalent to defining the option `time` as shown in the example in Figure 2. The timing facility can be turned off by setting the variable `iltime` to 0.

```plaintext
option.
    itime = 0
```

For more details about performance data collection, refer to XELLPACK Programmer's Guide.

The performance data collection facility not only records the timing of each of the modules, it also records other important information about the modules such as names and types of the modules, and the order in which the modules appear in the XELLPACK program. This information can be stored in a database so that systematic performance evaluation of numerical algorithms is possible.

4.1.2 Program Execution

Click the left mouse button in the last button in the session command panel. This starts the process of compiling and executing the XELLPACK program in the session template. The execution proceeds by first running the template contents through the XELLPACK language processor to generate a FORTRAN control program. The control program is compiled and linked with the XELLPACK libraries, and then executed. This process is controlled by a `shell script` which is displayed in a trace window during execution.
Recall that the *Configuration* button in the top-level XELLPACK control window is used to set the target machine for compiling and running the program. These settings are used by the shell script to determine the destination machine for the control program.

The Configuration settings are also used to determine which options you have chosen for XELLPACK program execution.

Here is a synopsis of the the entry fields in the Configuration window:

- **Max Grid Lines**: The default maximum setting for grid lines in the Grid Tool. You can increase (or decrease) the first two values. They represent the x and y grid lines.

- **pellpack Command**: The command used to invoke the XELLPACK language processor. Your PELLPACK environment variable determines where the *pellpack* shell script resides.

- **pellpack Options**: The default options are listed in the figure above. For a list of all available options, see Appendix A.

- **pellpack Server**: The target machine for compiling and executing the XELLPACK control program.
Pellpack Binary Type: The type of machine you used as the XELLPACK server. You must have compiled XELLPACK libraries which are appropriate for the machine on which you execute the XELLPACK programs.

When the Execute button is pressed, an execution trace window is opened to show the status of the execution. Figure 12 shows sample contents of the trace window with the -t option specified as a Pellpack Option. The trace information and the execution output can be found in the trace files generated by the XELLPACK processor.

A graph window describing the performance related to program parsing, generating the control programs, and other preprocessing tasks is displayed immediately before program execution begins. The command panel of the graph window contains three buttons which

![Figure 12: Execution Trace Window](image)
control the operation of the window. Here is a short synopsis of the panel buttons:

- **Close:** Close the graph window.
- **Hardcopy:** Send a copy of the graph window to a printer.
- **About:** Display a message box describing the graph window.

Observe the timing information contained in this window, then click the left mouse button in the Close button to dismiss it.

### 4.2 Problem Post-Processing

After the program is executed, XELLPACK post-processing is invoked. Post-processing includes execution error reports, solution visualization tools and performance analysis and visualization tools.

#### 4.2.1 Execution Tracing

If your program aborts, you should check the output files which have been generated by the XELLPACK processor. These files reside in the current directory unless you have specified
otherwise (see the pellpack option -u pathname in Appendix A). The execution trace files are:

- **prep_output**: trace file for language processing. This file lists syntax errors and errors in solution module names or parameters which have been found in your XELLPACK program.

- **cmpl_output**: trace file for compiler output.

- **run_output**: trace file for runtime output.

- **abt_output**: message indicating where program processing failed.

- **output**: trace file for overall processing. You should always check this file first.

Below is the trace information in the output file for our example.

```
Execution Output:
1
---------------------
domain processor
---------------------
boundary points found 80
boundary pieces found 8
grid size 20 by 20
execution successful

---------------------
hole processor
---------------------
boundary points found 38
boundary pieces found 2
grid size 20 by 20
execution successful

---------------------
mesh generator
---------------------
read triangular mesh from a file
number of nodal points 170
number of elements 268

---------------------
set
---------------------
execution successful

---------------------
discretization module
---------------------
```
bi-linear finite element

execution successful

-------------------------
indexing module
-------------------------
a is i

equations indexed 98
unknowns indexed 98

execution successful

-------------------------
solution module
-------------------------

itpack jacobis i

-------------------------
elpack output
-------------------------

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+  + max(abs(error)) on 170 nodal points = 6.1411262E-03  +
+  + 11 norm(error) on 170 nodal points = 1.2647379E-03  +
+  + 12 norm(error) on 170 nodal points = 1.9731126E-03  +
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

****************************************************************************
+  + max(abs(error)) on 170 nodal points = 6.1411262E-03  +
+  + 11 norm(error) on 170 nodal points = 1.2647379E-03  +
+  + 12 norm(error) on 170 nodal points = 1.9731126E-03  +
****************************************************************************

******************************************************************************
HOST ******************************************************************************
Total elapse time = 2.7900000
Total communication = 0.0000000
Node program load time = 0.0000000
Node program init+load = 0.0400000

Timing for each module:
Module Num  Setup  Elapse  Communication
701  0.000000  0.1900000  0.0000000
702  0.000000  0.1400000  0.0000000
284  0.000000  0.7500001  0.0000000
603  0.000000  0.0100000  0.0000000
313  0.000000  0.9699999  0.0000000
401  0.000000  0.0000000  0.0000000
515  0.000000  0.6500001  0.0000000
801  0.000000  0.0100000  0.0000000

4.2.2 3-D Solution and Data Visualization

For 2-dimensional domains, the 3-dimensional data visualization tool displays the com-
computed solutions and other user-defined functions in 3-dimensional graphics. The 3-dimensional
graphic solution visualization tool is invoked when the XELLPACK option xplot3d is set in
the template.

\begin{verbatim}
option.
  xplot3d.
\end{verbatim}

By default, the functions \( u, u_x, u_y, u_{xx}, u_{yy} \) and \( u_{xy} \) are added to the list of function
to be viewed. Other functions, including user-defined functions, can be added to the list by
defining the visualization modules in the template:

\begin{verbatim}
visualization.
  xplot3d (true)
visualization.
  xplot3d (error)
visualization.
  xplot3d (myfunction)
\end{verbatim}

The above segments cause the functions \textit{true} and \textit{error} to be included in the list of
functions to be viewed. The last example, \textit{myfunction} is a real-valued user-defined function
of two real variables, \( x \) and \( y \), which has been defined in the SUBPROGRAMS segment of
the XELLPACK program. This function might be used to define some part of the PDE, the
boundary conditions or the solution processing. The solution visualization tool is invoked
if either the option xplot3d is set or the visualization module xplot3d is used within the
template.

In our example, \textit{xplot3d} has automatically been specified for us in the template as a
result of the Symbolic Tool's program generation. The output window is displayed immedi­
ately after program execution terminates, unless execution is aborted as a result of improper
problem specification. The plotting selection window is shown in Figure 14.

The command panel contains two buttons. Here is a short synopsis of each:

- \textit{Function Selection}: Display the menu with the functions to view.
- \textit{Grid}: Modify the grid on which the function is graphed. Interpolation is used to
determine the function values on finer grid settings.
- \textit{Quit}: Close the 3-D visualization tool.

In this example, we can view \( u, u_x, u_y \), and \textit{true}. Click the left mouse button in the
\textit{Function Selection} button in the command panel, and move the mouse while holding down
the left mouse button. Select the function \( u \) and release the left mouse button. A window
containing the graph of $u$ on the problem domain is displayed. Note that multiple plots can be displayed by selecting functions from the control window without closing the previously selected plots.

The window containing the graph contains a control panel which allows you to manipulate the view of your solution. Here is a short synopsis of the panel buttons:

- **Quit:** Close the visualization tool.
- **Graph Type:** The type of graph which is used to display the plot. Users can select *Filled*, *No Panels*, *Color Wireframe*, or *Wireframe*.
- **Phi:** Rotate the domain on which the function is plotted. The rotation angle ranges from -180 degrees to 180 degrees. The current amount of rotation from the original position is displayed to the right of the Phi button.
- **Theta:** Rotate the view of the function. The initial view is from above. The rotation angle ranges from -90 degrees to 90 degrees. The current amount of rotation from the original view is displayed to the right of the Theta button.

Use the Phi and Theta scroll bars to rotate the domain and view. First place the left mouse button on the scroll position indicator and move the mouse while holding down the left mouse button. Release the button when the scroll position indicator has reached the appropriate location.

Close the window containing the plot of the solution $u$ by clicking on the Quit button in the command panel. Close the visualization tool by clicking on the Quit button in the control window.

### 4.2.3 Performance Visualization Tool

After the solution visualization tool is closed, the performance analysis and visualization tool will be invoked if the time option in the template has been set. This tool can be used for general performance evaluation or for algorithm-specific performance fine tuning. It
Figure 15: Graph of Solution Function

Figure 16: Multiple Solution Graphs
computes statistics based on performance data generated by the XELLPACK performance data collection facility, and displays the results in a graphical format. The performance visualization tool is shown in Figure 17.

Users can select any combination of categories of performance data and visualize them in different formats. Currently, only the bar chart and text formats are implemented; other formats will be added in future releases. The statistic analysis methods for nodes are relevant only for parallel execution.

The performance visualization tool consists of three panels: a command panel, an information panel and a message panel.

The top panel is the command panel which controls the operation of the window. Here is a short synopsis of the buttons in the command panel:

- **Exit**: Close the performance visualization tool.
- **Hardcopy**: Send a copy of the graph window to a printer.
- **Draw**: Visualize the performance data.

The middle panel is the information panel which is used to select analysis and display methods. Here is a short synopsis of the buttons in the information panel:

- **Processor(s)**: Select the processor or set of processors for which performance data will be generated. The choices are:
  - **Host Only**: Show the performance data of the host processor only.
  - **One node**: Show the performance data of a processing element.
  - **All nodes**: Show the performance data of all processing elements.
  - **Subset of nodes**: Show the performance data of a subset of the processing elements.

- **Analysis method**: Select the statistical methods which should be used to analyze the data.
  - **Mean value**: Compute the mean values of each category.
  - **Maximum**: Find the maximum value of each category.
  - **Minimum**: Find the minimum value of each category.
  - **Variance**: Find the variance value of each category.
  - **One By One**: Display the performance data of each node in the selected processor group one by one.
  - **Standard deviation**: Find the standard deviation of each category.
Figure 17: Performance Analysis Tool
Mean deviation: Find the mean deviation of each category.

- **Display method:** Select the format to display the performance data. Users can choose *Bar Chart* or *Text* in the current version of XELLPACK.

The area titled *Select the performance data to display* is used to compose the performance data. The categories include all modules which appear in the XELLPACK program along with a total. Each category has three entries: setup time, elapsed time, and communication time. The system allows you to select any or all of the above entries. Express buttons are also provided to set or clear the entire column with one click.

The bottom panel is the message panel. Messages which describe the information selection process and associated error messages are displayed in this area.

For our example, retain the default settings for the analysis and display selections. Click the left mouse button on the *Draw* button in the command panel. The pop-up performance window is displayed. This window is shown in Figure 18.

The performance graph window is similar in appearance and operation to the preprocessor performance window. The command panel contains three buttons which control its operation: Here is a short synopsis of the panel buttons:

- **Close:** Close the graph window.
- **Hardcopy:** Send a copy of the graph window to a printer.
- **About:** Display a message box describing the graph window.

Observe the timing information contained in this window, then click the left mouse button in the *Close* button to dismiss it. To close the performance visualization control window, click the left mouse button in the *Exit* button of the command panel.

### 4.2.4 Performance Evaluation Tool

The performance analysis and visualization tool is designed to evaluate the performance of the execution of a single XELLPACK program. The performance evaluation system (PES), is a system that is designed for systematic comparison of the performances of the modules for different domains and setups.

This tool can be used to help you to:

- visualize performance curves resulting from the analysis of performance data for given PDE solution method or methods, and
- generate text and bar charts for statistical analysis of the performance data.
Figure 18: Graph of Performance Data
Figure 19: Performance Evaluation Tool
The tool uses the performance data generated by the XELLPACK performance data collection facility. A sample PES session is shown in Figure 19.

The PES tool window consists of the command panel, the session information area and a message area. The PES command panel contains three buttons. Here is a short synopsis of the panel buttons:

- **Hardcopy**: Send a copy of the graph window to a printer.
- **Draw**: Generate the graph according to the specifications in the information area.
- **Help**: Access context-sensitive help.

The session information area consists of five panels. Each panel requires you to enter specifications which control the content of the performance graph.

In the first panel, **Select PDE Problem/Parameter sets**, you specify the number of problem/parameter sets and enter the problem and parameter number.

In the second panel, **Select Discretization(D)/Indexing(I)/Solution(S) module sets**, you enter the number of DIS sets and to select the Discretization, Indexing, and Solution modules from the menu buttons. After the selection of each, press the OK button to continue or the Cancel button to cancel the last selection and reenter it.

In the third panel, **Select performance data (as x and y coordinates)**, you select the performance data to be analyzed through the menu buttons.

In the fourth panel, you select a machine for the solution of the PDE problem(s). You also enter the number of processors (for parallel execution only).

The fifth panel is a selection box from which you select a statistical method according to which the selected performance data will be analyzed.

After entering the above information, you can select **Continue** to proceed with the analysis, or **Cancel** to cancel the session and begin again. If you proceed, the data will be analyzed and a message in the message area will indicate when the analysis is complete.

You can now request that the tool display the data as a **Curve**, as **Text**, or as a **Bar Chart**. This selection is made in the **Display method** panel. After choosing the display method, click the left mouse button on the **Draw** button in the command panel. The graph will be generated.

The command panel of the graph window contains three buttons which control the operation of the window. Here is a short synopsis of the panel buttons:
• Close: Close the graph window.

• Hardcopy: Send a copy of the graph window to a printer.

• About: Display a message box describing the graph window.

Observe the performance information contained in this window, then click the left mouse button in the Close button to dismiss it. Close the PES system by clicking the left mouse button in the QUIT button immediately above the message area.
5 Closing Remarks

XELLPACK is a high level programming environment for defining and solving PDE problems. In this document we discussed the XELLPACK user interface and the tools which are provided to support

- the interactive definition of PDE problems,
- the specification of their solution methods,
- the generation and execution of the XELLPACK program and
- the visualization of the solution and performance data.

For users who want more details about the XELLPACK library of solution modules or the XELLPACK system design, please refer to the following documents:

- XELLPACK Library Guide
- XELLPACK Programmer’s Guide

Bug reports and suggestions can be sent to the following address:

The //ELLPACK Group
Department Of Computer Sciences
Purdue University
West Lafayette, IN 47906
U. S. A.

E-Mail Address: pellpack@cs.purdue.edu

Your comments on this Guide and on any component of the system are welcome.

Acknowledgements

The XELLPACK project was partially supported by the following grants: National Science Foundation CCR-8619817, AFOSR 88-0243, Strategic Defense Initiative (through the ARO) DAAL03-90-0107, ESPRIT project GENESIS.

XELLPACK has been created by the following current or past members of the PELLPACK group: William Bouma, Ann Christine Catlin, Nicholas Chrisochoides, Athanasios Gaitatzes, Apostolos Hadjidimos, Catherine Houstis, Elias Houstis, Sang Bae Kim, Stavros Kortesis, Tunghsing Ku, Yue-Jinning Lian, Mo Mu, Panos Papachiou, Winniw Ng, John Rice, Meletis Samartzis, George Vanecek, Pelayia Varodoglu, Manolis Vavalis, Ko Yang Wang, Po-Ting Wu, and Sanjiva Weerawarana.

The following persons contributed to the writing of this User Guide: Ann Christine Catlin, Ko-Yang Wang, Sanjiva Weerawarana, Pelayia Varodoglu, Xingkang Fu, and Elias Houstis.
REFERENCES

References


REFERENCES


Appendix A: XELLPACK Execution Options

The following options and binary file types can be specified in the Pel/pack Options entry field of the Configuration window:

```
# [options] [binfiles]

Options can be any .o, .a, .qm and .lib files to link with the executable.

Options are listed below

-1 => run XELLPACK program sequentially
-2 => use double precision XELLPACK
-7 => use a standard FORTRAN 77 preprocessor
-c Template => use Template as the control template
-d => compile with debug option
-o => save the XELLPACK control program
-e => input file is to be run through genpgm
-i => enable interaction
-h => help
-k => keep the temp directory
-l file => include lfile in link edit
-m mfile => specify the makefile to use
-n NODE => name of node OS to use
-o ofile => put XELLPACK output on ofile
-p => preprocessor only (implies -e)
-s Savefile => save pes data in file Savefile
-t => trace this shell script
-u udir => use udir for XELLPACK files
-C => compile only, save object file in PELLPACK
-D => generate a domain decomposition server
-E => don't run preprocessor and compiler, execut only

must use -u to tell where to find the executable

-H machinename => name of the machine to run preprocessor
-N machinename => name of the target machine
-o => save object code
-S plfile => save XELLPACK Plot file on plfile
-P printer => send XELLPACK Plot file to printer
-X DISPLAY => specify the X window display to use for graphic output
-U DISPLAY => use DISPLAY to open monitoring window
```
Appendix B: Solving PDE Problems with XELLPACK/FIDISOL
Appendix C: Sample XELLPACK Sessions
The first session uses *elltool* to load and run an existing XELLPACK program. The problem is described in the ELLPACK book [20] on page 19 and the description is repeated here for your convenience.

Find the elliptic function $u(x, y)$ which satisfies the partial differential equation

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

in the rectangle $0 < x < 1$, $-1 < y < 2$, and satisfies the Dirichlet boundary conditions

$$u = 0 \text{ for } x = 0, -1 < y < 2$$
$$u = \sin(\pi x) - \frac{x}{2} \text{ for } y = -1, 0 < x < 1$$
$$u = \frac{1}{2} \text{ for } x = 1, -1 < y <$$
$$u = x \text{ for } y = 2, 0 < x < 1$$

The ordinary finite difference approximation is used to discretize the problem at the points of a $6 \times 6$ grid, and the resulting linear system is solved with ordinary Gauss elimination for band matrices.
Type `elltool`. The `elltool` window is displayed:

![elltool window](image)

Click the left mouse button in LOAD. The Select File window is displayed:

![selFile window](image)

The top entry field contains the file which is currently selected. It is initially blank. The display below the entry field shows the directory structure. The top bar of the right column shows the name of the current directory, and the remaining entries are the names of the files under the current directory. The top bar of the middle column shows the parent directory of the current directory, and the remaining entries
are the names of the files or directories under that parent directory. The top bar of the left column shows
the parent directory of the directory whose name appears in the top bar of the middle column. Similarly,
the remaining entries are the names of the files or directories under that directory.

We want to load the file "ex.1.el", so move the cursor to the line ex.1.el and click the left mouse button.
The filename ex.1.el appears in the entry field of the selFile window. Click the left mouse button in OPEN.
The XELLPACK 2D Session window for solving XELLPACK problems in two dimensions appears:

The XELLPACK program in the window is discussed in detail in the ELLPACK book[20] as example
1.e1. We do not need to change anything before running the program, so move the cursor to the rightmost
button, run, and click the left mouse button. The XELLPACK execution trace window appears. After the
XELLPACK pre-processor has finished processing the program, you will see the xgraph window showing
XELLPACK Preprocessor Timing information.

A portion of the text which appears in the trace window during program execution is shown below.
This is the trace listing produced during XELLPACK execution of ex.1.e1. The description of the text
output is in the ELLPACK book on page 19.
discretization module

5-point star

domain rectangle
discretization uniform
number of equations 16
max no. of unknowns per eq. 5
matrix is non-symmetric

execution successful

solution module

linpack band

number of equations 16
lower bandwidth 4
upper bandwidth 4
required workspace 224

execution successful

ellpack output

x-abscissae are

- 0.000000E+00 2.000000E-01 4.000000E-01 6.000000E-01
  8.000000E-01 1.000000E+00

y = 2.000000E+00

- 0.000000E+00 2.000000E-01 4.000000E-01 6.000000E-01
  8.000000E-01 1.000000E+00

y = 1.400000E+00
ellpack output

contour plot of $u$
grid 20 by 20
execution successful

The //ELLPACK Preprocessor Timing window is shown below:
The first (red) bar shows the time used by initialization process. The next (green) bar represents the time used to parse the XELLPACK program. The third (blue) bar shows the time used to create the main control program and the last (cyan) bar shows the total time used by the //ELLPACK preprocessor.

Close the //ELLPACK Preprocessor Timing window by clicking the left mouse button in CLOSE. The //ELLPACK Performance Analysis & Visualization Tool window appears:
The first line, **Processors(s)**, lets you select information on **Host only**, **One node** or **All nodes** by clicking the left mouse button in the corresponding block to darken it. For sequential execution, only the Host information is relevant. The second line, **Analysis method**, lets you choose the method to measure the performance by clicking the left mouse button in the corresponding block. The third line, **Display method**, lets you choose the display method. Data can be displayed textually or using a bar chart. You choose a method by clicking the left mouse button in the corresponding block.

In the area titled **Select the performance data to display**, the first line, **total**, represents the total time used. The second line, **di** (5 point star), represents the time used by the discretization module. In this case, 5 point star. The third line, **so** (linpack band), represents the time used by the solution module, in this case, linpack band. The fourth line, **ou** (table), represents the time used by the output module to output the **table(u)**. The fifth line, **ou** (plot), represents the time used by the output module to output the **plot(u)**. The last line is a set of express set/clear buttons to change all items in the corresponding column.

The left column represents the time used by the setup process for each module. The middle column represents the execution time used by each module, and the right column represents the communication time used by each module. If you want to see certain information, darken the corresponding block by clicking the left mouse button in the block. If you click the left button in a black block, it will become...
white and the corresponding information will not be shown. If you click the left button in the blocks on the bottom line, all information on corresponding column is selected or de-selected.

Click the left mouse button in HARDCOPY, and the information will be printed by your default printer. Click the left mouse button in DRAW, and the information will be shown in the X-window.

For example, if we want to see the information for the host processor only, and we want to use the Maximum analysis method, click the left mouse button in Maximum on the second row, and then click the left button in DRAW:

![Graph Image]

The right side of the graph window contains a legend which describes the color code for the information displayed to the left. After you close this window, you can click the left mouse button in EXIT in the //ELLPACK Performance Analysis & Visualization Tool window. Then hit return in the XELLPACK execution trace window.

You can review the execution results of the XELLPACK program by checking the output file. This file resides in the current directory, unless you have specified an alternate directory using the -u pathname option in the Configuration window.
The second session demonstrates a special XELLPACK feature for bypassing the file selection window. This allows you to initiate elltool and load your file in at the same time.

If you know the name of the file you want to work on, you can specify the directory path and filename of that file when you call elltool. For example, if your file resides in the directory /usr/pellpack/users/acc/e and the name of the file is sample.e, then you can specify

elltool /usr/pellpack/users/acc/e/sample.e

The XELLPACK session will appear with the file already loaded in.
If the file ex_1.el resides in the current directory, and you type in `elltool ex_1.el`, the following two windows will appear:

![Example XELLPACK 2D Session window](image1.png)

If you just want to run this program, move the cursor to the rightmost block, `run`, and click the left mouse button. For the XELLPACK program and its text output, consult the ELLPACK book [20] on page 19. The rest of the processing is identical to that in the previous session.

```plaintext
# Example ellpack program 1.el

options.
  time 5 memory

equation.
  uxx + uyy + 3.0*ux - 4.0*u = exp(x+y)*sin(pi*x)

boundary.
  u = 0.0  on  x = 0.0
  u = sin(pi*x) - x/2.0  on  y = 1.0
  u = y/2.0  on  x = 1.0
  u = x  on  y = 2.0

grid.
  6 x points
  6 y points

discretization.
  5 point star

solution.
  linpack band

output.
  table (u)
  plot (u)

end.
```
The third session uses elltool's Symbolic Tool to specify the partial differential equation. The problem is described in the ELLPACK book [20] on page 52, and the description is repeated here for your convenience.

This example solves a completely general equation with mixed boundary conditions on a rectangular domain. The domain is discretized using a 6x6 grid. A function, \( f(x, y) \), will be constructed to force a solution for \( u \) called the true solution. That is, we want \( u(x, y) = \text{true}(x, y) \). The equation is

\[
\frac{\partial^2 u}{\partial x^2} + (1 + y^2) \frac{\partial^2 u}{\partial y^2} - \frac{\partial u}{\partial x} - (1 + y^2) \frac{\partial u}{\partial y} = f(x, y)
\]

where \( f(x, y) \) is chosen to make this PDE satisfy

\[
u(x, y) = e^{x+y} + (x(x - 1))^2 \log(1 + y^2)
\]

in the rectangle \( 0 < x < 1, \ 0 < y < 1 \) with mixed boundary conditions

\[-u + u_{x} = 0 \text{ for } x = 1 \\
u = \text{true}(x, y) \text{ for } y = 0 \\
u + u_{x} = 2e^y \text{ for } x = 0 \\
u = \text{true}(x, y) \text{ for } y = 1
\]

Two solution schemes are used to solve this problem. The solution given by the last method is used for visualizing the solution.
Type in `elltool`; the `elltool` window appears:

Click the left mouse button in NEW FILE. The XELLPACK 2D Session window appears, and it is initially empty.

Click the left mouse button in the //ELLPACK-Maxima Interface (third from left). The //ELLPACK-Maxima Interface window appears:
This tool is used for defining the system of PDEs you want to solve. The default value for Mode of Operation is PELLPACK. It can be changed to FIDISOL by pressing the left mouse button in the block for PELLPACK and releasing the left button on FIDISOL. The default value for Number of Operators is 1. It can be changed by erasing 1 and typing in another value. The default discretization method is 5-Point Star. You can choose other methods by pressing the left mouse button in the block for 5-Point Star and releasing the left button on the method you prefer. The default Indexing method is As Is and the default Linear Solver is Band GE. They can both be changed as described above.

In the PDE Operator Editor entry field, type:

\[ uxx + (1.0+y^2)uyy - ux - (1.0+y^2)uy \]

Then click the left mouse button in Forcing Solution Information. You will see:
\[ \exp(x+y) + ((x*(x-1.0))**2)*\log(1.0+y**2) \]

Click the left mouse button in OK to accept the Forcing Solution Information. Then click the left mouse button in Generate Program in the //ELLPACK-Maxima Interface window. To exit the //ELLPACK-Maxima Interface window, click the left mouse button in File on the top line, and then click the left button in Quit.

Now the XELLPACK 2D Session window appears as below:

This program is automatically generated by \textit{elltool}. You may notice that the TRUE function is what you specified in the Forcing Solution Information window. \textit{elltool} automatically calculates the righthand
side of the equation, \( \text{FORCE}(X,Y) \), based on the TRUE solution and on the equation you specified in the PDE Operator Editor window. The DISCRETIZATION, INDEXING and SOLUTION segments are the defaults from the XELLPACK-Maxima Interface window. The OPTIONS, BOUNDARY, GRID, and OUTPUT segments are also defaults which were generated by the tool.

Now we can change some of the default values. Replace the BOUNDARY segment with:

\[
\begin{align*}
-u + ux &= 0. & \text{on } x=1. \\
    u &= \text{true}(x,y) & \text{on } y=0. \\
   u + ux &= 2.0*\exp(y) & \text{on } x=0. \\
     u &= \text{true}(x,y) & \text{on } y=1.
\end{align*}
\]

by removing the default and typing into the session template.

In the same way, change the GRID segment to:

\[
20 \text{ x points } 20 \text{ y points}
\]

and add an additional output segment:

\[
\text{OUT. } \text{max(true)}
\]

Remove the INDEXING segment and change the default OUTPUT segment to:

\[
\text{OUTPUT.} \\
    \text{table(u) } \text{max(error,7,9)}
\]

Add a second solution path to your program by typing:

\[
\text{dis. } \text{hermite collocation} \\
    \text{sol. } \text{band ge} \\
    \text{out. } \text{table(u) } \text{max(error,7,9)}
\]

after the OUTPUT segment. You can add some comments at the beginning of the XELLPACK program. The XELLPACK 2D Session window now looks like:
To run the program, move the cursor to the rightmost block, run, and click the left mouse button. For the XELLPACK program and its text output, consult the ELLPACK book[20] on page 52 example 3.b1.

Note that the FORCE function generated by elltool looks different from the F function in example 3.b1, but they are the same function. Also, the grid segment 20 x points $\times$ 20 y points is different from the grid segment 4 x points $\times$ 5 y points in example 3.b1, so the text output will be different from that of example 3.b1.

The remainder of the session is similar to the previous sessions, except that since we have xplot3d in the program, the following window appears:
Press the left mouse button in Function Selection and move the cursor to $U$. Release the left button and you will see:

![Graph of U]

Solution resulting from Hermite Collocation discretization

To view the functions $U_x, U_y, U_{xx}, U_{yy}$ and $U_{xy}$, proceed as above. Before you leave the XELLPACK 2D Session window, do not forget to save your XELLPACK program.
The fourth session uses elltool's Boundary Tool to create a domain with a hole and to specify the boundary conditions. We will also use the Grid Tool to specify the domain discretization. The problem is similar to the one described in the ELLPACK book [20] on page 93.

ELLPACK can handle additional boundary conditions on arc or holes placed inside the domain. We will solve Laplace's equation

\[ u_{xx} + u_{yy} = 0 \]

and specify Dirichlet boundary conditions on the domain parameterized by the XELLPACK Boundary Tool. The domain is discretized using a 21x21 grid. The operator is discretized by 5 point star and the resulting linear system is solved using Band Gauss Elimination.
Type in `elltool`, the elltool window appears:

![elltool window](image)

Click the left mouse button in `NEW FILE`. The XELLPACK 2D Session window appears:

![XELLPACK 2D Session window](image)

Click the left button in the XELLPACK-Maxima Interface (third from left). The XELLPACK-Maxima Interface window appears:
In the PDE Operator Editor, type:

$$u_{xx} + u_{yy} = 0.0$$

Then click the left mouse button in Generate Program. Press the left button in File on top line and release the left button in Quit to quit the XELLPACK-Maxima Interface window.

Now the XELLPACK 2D Session window appears as follows:
Click the left mouse button in the Boundary Specification Editor (fourth from left). The Boundary Specification Editor window appears:
Click the left button in Clear, and then in Set Range. You will see:

Change the default range to: -3.76 3.76 -3.63 3.63. Then click in Continue. Now the range in the Boundary Specification Editor window will be from -3.76 to 3.76 in the X direction and from -3.63 to 3.63 in the Y direction. Click the left button in New Boundary, and you are now ready to set up the new boundary. Start from (0.0, 3.63), and click the left mouse button at the following points: (0.49, 3.60) (0.97, 3.50) (1.44, 3.35) (1.88, 3.14) (2.29, 2.88) (2.66, 2.56) (2.98, 2.20) (3.26, 1.81) (3.48, 1.39) (3.63, 0.94) (3.73, 0.47). Then click the middle button at (3.76, 0.00). This will make the first piece of the boundary from (0.00, 3.63) to (3.76, 0.00).

Now click the left button at the following points: (3.73, -0.47) (3.63, -0.94) (3.48, -1.39) (3.26, -1.81) (2.98, -2.20) (2.66, -2.56) (2.29, -2.88) (1.88, -3.14) (1.44, -3.35) (0.97, -3.50) (0.49, -3.60). Then click the middle button at (0.00, -3.63). This will make the second piece of the boundary from (3.76, 0.00) to (0.00,
Click the left button at the following points: (-0.49, -3.60) (-0.97, -3.50) (-1.44, -3.35) (-1.88, -3.14) (-2.29, -2.88) (-2.66, -2.56) (-2.98, -2.20) (-3.26, -1.81) (-3.48, -1.39) (-3.63, -0.94) (-3.73, -0.47). Then click the middle button at (-3.76, 0.00). This will make the third piece of the boundary from (0.00, -3.63) to (-3.76, 0.00). Click the left button at the following points: (-3.73, 0.47) (-3.63, 0.94) (-3.48, 1.39) (-3.26, 1.81) (-2.98, 2.20) (-2.66, 2.56) (-2.29, 2.88) (-1.88, 3.14) (-1.44, 3.35) (-0.97, 3.50) (-0.49, 3.60). Then click the right button at (0.0, 3.63). This will make the fourth piece and also close the boundary.

The Boundary Specification Editor window appears as follows:

Click the left button in Set Conditions, and you will see a small window.
Type 0.0 in each entry field, and then click the left button in Continue. This establishes the boundary conditions for the first four boundary pieces.

Now click the left mouse button in New Hole. Click the left mouse button at (-1.00, 0.00), click the middle button at (0.00, -0.50) (1.00, 0.00) (0.00, 0.50) and click the right button at (-1.00, 0.00) to close the hole. The Boundary Specification Editor window now looks like:

![Boundary Specification Editor](image)

Again, click the left mouse button in Set Conditions. In the first two entry fields type in 1, and in the last two entry fields type in 2.0 - x**2. Then click the left mouse button in Continue. Now click the left button in Save to save the domain, the hole and their conditions to the session template.

Click the left mouse button in the Grid Specification Editor (fifth from left) in the XELLPACK 2D Session window. The Grid Specification Editor window appears:
Click the left mouse button in Set Range, and the following window appears:

Enter range \( (x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}) \): \((-1.00, 1.00, -1.00, 1.00)\)

Press <Return> or [Continue] to accept.

Change the default range to \(-3.76, 3.76, -3.63, 3.63\), then click in the Continue block. Now the range in the Grid Specification Editor window will be from -3.76 to 3.76 in the X direction and from -3.63 to 3.63 in the Y direction. Then click the left mouse button in Uniform Grid, and you will see:
Change the default grid line to 21 and click in the Continue block. This changes the X direction grid lines to 21 because the Mode is X. Click the left mouse button in the Mode: X block; it becomes Mode: Y. Then click the left button in Uniform Grid and change the default Y direction grid line to 21. Now the Grid Specification Editor window looks like:

Click the left mouse button in Save, and then click the left button in Quit to quit the Grid Specification.
Editor window. You can now quit the Boundary Specification Editor by clicking the left button in Quit. At this time, part of the XELLPACK 2D Session window looks like:

Notice that the BOUNDARY and GRID sections have been changed to our specifications, and a HOLE section has been added to the XELLPACK program.

To run the program, move the cursor to the rightmost block, run, and click the left mouse button. For the XELLPACK program and its text output, consult the ELLPACK book[20] on page 93, example 5.a2. The solution $U$ looks like:
You can view the solution $U$ from different angles by moving the cursor left and right to the desired angles in the Phi and Theta scroll bars. The following is a view of function $U$ with Phi = 180 and Theta = -59:
Note that the actual boundary and hole conditions are:

bound. \( u = 0 \) on \( x = \cosh(2.0) \cdot \sin(t), y = \sinh(2.0) \cdot \cos(t) \) &
for \( t = 0.0 \) to \( 2\pi \)

hole. \( u = 1 \) on line -1.0, 0.0 to 0.0, -0.50 to 1.0, 0.0
\( u = 2 \cdot -x^2 \) on line 1.0, 0.0 to 0.0, 0.50 to -1.0, 0.0

grid. 21 x points \(-\cosh(2.0)\) to \(\cosh(2.0)\)
21 y points \(-\sinh(2.0)\) to \(\sinh(2.0)\)

Since the boundaries, the hole and their conditions are generated by the tools, they are slightly different from that of example 5.a2 in the ELLPACK book on page 93. Also, in order to show the hole more clearly, the hole in this example is different from that in the example. As a result, the solution and the text output are also different from that of example 5.a2.