1988


Wayne R. Dyksen

Carl R. Gritter

Report Number: 88-840
ELLiptic EXPERT: AN EXPERT SYSTEM FOR ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

Wayne R. Dyksen
Carl R. Gruter

CSD-TR-840
December 1988
Elliptic Expert:

An Expert System for Elliptic Partial Differential Equations

Wayne R. Dyksen
Carl R. Gritter

Purdue University
Department of Computer Sciences

December 1988
Elliptic Expert: 
An Expert System for Elliptic Partial Differential Equations

Wayne R. Dyksen†
Carl R. Gritter‡

1. Overview

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

We believe that problem oriented, very high level languages represent a first step toward the modernization of scientific software. An example of such a system is XELLPACK, an X Window System based problem solving environment for solving elliptic partial differential equations (PDEs). We believe that it represents a significant step in the development of scientific problem solving environments since it makes significant use of interactive color graphics output and input, as well as use of the X client-server paradigm. XELLPACK provides a natural interface to bridge the gap from the world of the scientist or engineer to the world of the numerical analyst.

Although XELLPACK contains vast raw PDE solving power, it takes an “elliptic expert” to make full use of its capabilities. For a given elliptic problem, XELLPACK provides 1147 distinct solution paths. For the nonexpert user, choosing a valid path is difficult while choosing the “best” path is nearly impossible. The need for confidence in results thus dictates selecting inferior familiar algorithms over superior unfamiliar ones. Most similar sophisticated scientific systems share analogous drawbacks.

As a step toward the solution of this problem, we are investigating the use of artificial intelligence techniques to make powerful scientific computing techniques usable by nonexperts. We are building Elliptic Expert, an expert system for solving elliptic PDEs. Elliptic Expert incorporates enough expertise to make the extensive problem solving capabilities of XELLPACK completely accessible to the nonexpert; i.e., the average design engineer. Elliptic Expert not only advises the user in the selection of the “best” solution method, but also aids in the analysis of the accuracy of the computed solution.

2. ELLPACK

ELLPACK is a very high level language for solving elliptic partial differential equations (PDEs) developed at Purdue University (Rice & Boisvert, 1984). It provides the elliptic problem solving machinery for Elliptic Expert. The basic building blocks in an ELLPACK program are segments which perform various tasks necessary to define and solve an elliptic problem. The equation, boundary and grid segments are used to define the PDE, the domain and the grid. ELLPACK contains four basic types of elliptic problem solving segments. Discretization modules discretize the continuous problem by generating a system of linear equations. Indexing modules are used to order the linear system which is then solved by a solution module. Triple modules incorporate all three of

Supported in part by National Science Foundation grants CCR-8612676 and DCR-8602355.
the above steps into one module. Other segments allow the user to specify output modules and to incorporate FORTRAN code and subroutines into an ELLPACK program. ELLPACK currently includes 9 discretization modules, 7 indexing modules, 18 solution modules and 13 triple modules.

Though ELLPACK was initially developed as an environment for evaluating the performance of algorithms and software for elliptic PDEs, it is also recognized as a very powerful tool for solving a large class of problems. Besides its basic function of solving second order, linear elliptic PDEs with Dirichlet, Neumann, mixed or periodic boundary conditions, ELLPACK has been used to solve nonlinear problems, time dependent problems and coupled systems of elliptic equations. Moreover, extensions of ELLPACK have been built to handle multiple domains, automatic domain mapping, Schwarz splitting and automatic grid adaption.

3. XELLPACK

The original ELLPACK system is strictly "batch" oriented and produces only simple, monochrome graphics output. With the advent of modern scientific workstations comes the ability to do nontrivial interactive computing and sophisticated interactive color graphics. In order to take advantage of these powerful problem solving environments, Wayne Dyksen and Calvin Ribbens have developed Interactive ELLPACK [Dyksen and Ribbens, 1986]. Interactive ELLPACK is an extension of ELLPACK which improves upon it several important ways.

1. A new menu segment was added which allows the user to build menus of traditional ELLPACK statements, and choose interactively from them at run time.
2. ELLPACK's grid specification scheme was extended to include a grid module interactive which allows the user to specify and change grids via graphical input devices throughout the execution. In particular, the grid construction can be overlaid on a plot of any function such as the residual or an estimate of the error.
3. New three-dimensional color graphics output modules were incorporated [Bonomo and Dyksen, 1987].

Most recently, Interactive ELLPACK has been supplanted by XELLPACK, an X Window System based version of ELLPACK [Bonomo and Dyksen, 1988]. XELLPACK provides graphics input for constructing grids, pop-up menus for selecting solution techniques, and color graphics output for analyzing solutions. Using the X paradigm, a user can interface with XELLPACK from any X workstation while an XELLPACK client solves an elliptic problem on any machines or machines on the network.

4. Elliptic Expert

Elliptic Expert is an extension of Interactive ELLPACK which guides the user to the solution of an elliptic problem. Elliptic Expert advises the user in three main areas, namely

1. in selecting the "best" ELLPACK elliptic problem solver
   (i.e., the "best" discretization/indexing/solution module sequence or the "best" triple module),
2. in selecting the "best" grid, and
3. in analyzing the accuracy of the computed solution.

A sample Elliptic Expert program to solve the following problem is shown in Figure 4.1:

\[-\nabla^2 u - 20\pi^2 u = 0 \quad (x, y) \in (0, 1) \times (0, 1)\]
\[ u = 0 \quad x = 0, 1, \quad y = 0\]
\[ n_x = 4\pi\sin(2\pi x) \quad y = 1.\]

The Elliptic Expert program in Figure 4.1 uses two menu segments. The first gives the user choices in solving the elliptic problem, and the second allows the user to graph any of several functions. The resulting X windows are managed by a built-in pop-up menu. There is no limit to the number of menus that can be included in an Elliptic Expert program. The objectives, knowledge and advice segments are new and unique to Elliptic Expert. A sample display from this program is shown in Figures 4.2.
**Elliptic Expert**

**Options:**
- Max x points = 33
- Max y points = 33
- Interpolation = splines

**Objectives:**
- Accuracy = 0.5

**Knowledge:**
- Solution is periodic

**Equation:**
- $u_{xx} + u_{yy} + (20\pi^2u^2) = 0$

**Boundary:**
- $u = 0$ on $x = 0$
- $u = 0$ on $x = 1$
- $u_y = 4\pi \sin(2\pi x)$ on $y = 0$

**Menu - 'Solution Methods'**
- 'ga: grid advice'
- 'ig: interactive grid'
- 'ma: method advice'
- 'fd: finite differences'
- 'fe: finite elements'
- 'fft: high order fft'

**Menu - 'Output Menu'**
- 'ct: contour true'
- 'cu: contour u'
- 'ce: contour error'
- 'pt: panel true'
- 'pu: panel graph u'
- 'pe: panel error'
- 'gt: graph true'
- 'gu: graph u'
- 'ge: graph error'

**Subprograms**

```fortran
function true(x,y)
    common / clrvgl / zlepsg, zlepsm, pi
    ttrue = sin(2*pi*x) * sin(4*pi*y)
    return
end
```

**Figure 4.1. Sample Elliptic Expert Program.** The program uses two menu segments; the first gives the user choices in solving the elliptic problem, and the second allows the user to graph any of several functions. The resulting X windows are managed by a built-in pop-up menu. The objectives, knowledge and advice segments are new and unique to Elliptic Expert. A sample display of Elliptic Expert is given in Figure 4.2.
Figure 4.2. Sample Elliptic Expert Session. Elliptic Expert advice is given in the upper left window. Graphs of the absolute value of the error for two different grids is shown in the two lower windows. (The actual Elliptic Expert display is in color.)
The architecture of Elliptic Expert is given in Figure 4.3. The expert system portion of Elliptic Expert is being implemented in OPS5 which is a member of the family of production-system languages based on the production-system paradigm [Brownston et al., 1986]. The basic compound data structure definable in OPS5 is called an element class which is similar to a record in Pascal or a structure in C. Components of an element class are called attributes. An OPS5 program consists of a declaration section which describes the element classes used in the program, followed by a production section which contains the rules. A generic element class, Class, is declared by

\[
\text{(literalize Class attribute1 attribute2 attributeK)}
\]

A generic rule is given by

\[
\text{(p RuleName condition1 condition2 ... conditionM \rightarrow action1 action2 ... actionN)}
\]

The basic architecture of OPS5 is included in Figure 4.3. During execution, instances of element classes are kept in working memory while rules are kept in production memory. Working memory is usually initialized after the declarations and rules have been loaded; instances of element classes are created via the OPS5 make action. The conditions in the rules involve element classes. The actions may include making, removing or modifying instances of element classes in working memory, making or removing rules in production memory, writing output to the user, and reading input from the user.

When run, the OPS5 inference engine executes the following so-called recognize-act cycle.

```
repeat
  do unification pattern match with bindings
  perform conflict resolution
  fire the selected rule
until (the conflict set is empty) or
(a halt is performed) or
(the cycle count is reached) or
(a breakpoint is reached)
```

To do the pattern matching, the inference engine compares the contents of working memory with the conditions in each rule. If there exist instances of element classes in working memory which satisfy the conditions of a rule, then an instance of the rule with the particular bindings is added to the conflict set. If, after the pattern matching, the conflict set contains more than one rule instantiation, a series of tests collectively constituting conflict resolution is performed to select one instantiation to be fired (i.e., the actions associated with the rule are executed). Each test produces a partial ordering of the conflict set; instantiations that are dominated by others (i.e., deemed less important) are discarded. One selection strategy places precedence on the first condition of each rule; this is called the means ends analysis (MEA) selection strategy.

Elliptic Expert is goal oriented, using the MEA selection strategy in OPS5. Examples of types of goals are select_applicable_discretization, select_best_discretization and print_discretizations. Elliptic Expert has two types of goal strategies, symbolic and algorithmic. The symbolic strategies are based on \textit{a priori} knowledge; that is, the goal achieving process involves only symbolic analysis of the elliptic problem,
Elliptic Expert

Figure 4.3. Architecture of Elliptic Expert. ELLPACK provides the problem solving power, XELLPACK provides a high level graphics and network interface, and Elliptic Expert provides guidance in problem solving and analyzing computed results.
knowledge from the user, theoretical performance knowledge of modules and actual performance knowledge of modules applied to other problems. The algorithmic strategies are dynamic and based on both \textit{a priori} and \textit{a posteriori} knowledge; that is, the goal achieving process may involve any of the following: calculations to study the behavior of the coefficients, forcing function and boundary data; trial, low accuracy, cheap solutions; trial solutions using the two or three "most promising" methods; or, additional input from the user after presentation of the initial results.

The knowledge-base of Elliptic Expert contains facts and rules which it uses to achieve goals in the process of solving an elliptic problem. The knowledge-base facts describe the following:

1. The elliptic problem by defining
   - the domain,
   - the partial differential equation,
   - the boundary conditions, and
   - the grid;
2. User objectives by giving
   - accuracy requirements, and
   - resource constraints;
3. And metaknowledge such as
   - elliptic problem properties, and
   - solution properties.

For example, OPS5 declarations for the element classes describing an elliptic problem in Elliptic Expert are given in Figures 4.4 and 4.5; the naming conventions used in Elliptic Expert are inherited from ELLPACK. The information needed to create instances of these element classes is gleaned from the Elliptic Expert program by the preprocessor. Instances are created via the OPS5 \textit{make} action; these makes are written to a file from a FORTRAN environment before OPS5 is invoked. In turn, OPS5 reads this file of makes and creates instances of these element classes in its working memory.

The knowledge-base contains goal achieving rules pertaining to the following:

1. Discretization or triple modules by determining
   - the applicability to the elliptic problem,
   - the effect of grid on the discrete problem,
   - convergence properties,
   - the discrete problem properties,
   - resource requirements, and
   - relative ranking from the past performance data;
2. Solution modules by determining
   - the effect of and/or need for indexing,
   - the applicability to discretization,
   - resource requirements, and
   - relative ranking from the past performance data;
3. Goal strategies for
   - selecting applicable solution paths,
   - selecting the "best" solution paths,
   - selecting the "best" grid, and
   - evaluating the solution.

As an example, an OPS5 declaration for the element class describing discretization modules is given in Figure 4.6. Example rules for the goal \textit{select_applicable_discretization} are given in Figure 4.7 for the module 5-
Each coefficient of the PDE can have the value

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero</td>
<td>0</td>
</tr>
<tr>
<td>one</td>
<td>1</td>
</tr>
<tr>
<td>constant</td>
<td>constant, but not 0 or 1</td>
</tr>
</tbody>
</table>

[x][y][z] is nonconstant and depends on all of the independent variables given (e.g., "xy" means it depends on x and y)

```plaintext
(literalize PDE

cuxx ; coefficient of uxx
cuxy ; coefficient of uxy
cuxy ; coefficient of uyy
cux  ; coefficient of ux
cuy  ; coefficient of uy
cu   ; coefficient of u

c uz ; coefficient of uzz
cuzx ; coefficient of uzx
cuzy ; coefficient of uz y
cuz  ; coefficient of uz

liself ; t if in self-adjoint form

llcrst ; t if no cross derivative terms are present

llcstc ; t if all the coefficients are constants

llpois ; t if Poisson equation

lllapl ; t if Laplace's equation

llhmeq ; t if homogeneous

llhmbz ; t if helmholtz

)

; Domain

(literalize Domain

lltwod ; t if two dimensional

llrect ; t if rectangular

llckw ; t if boundary is parameterized clockwise

llhole ; t if holes

llarcx ; t if internal arcs

)
```

Figure 4.4. Elliptic Expert OPS5 Declarations. These declarations define the element classes describing the partial differential equation and the domain.
Boundary Conditions

The boundary conditions can be one of "dirichlet", "neumann", "mixed", or "periodic".

(vector-attribute nbnd)
(literalize BC)

; Rectangular domain
  right  ; x = rlxgr
  bottom ; y = rlygr
  left   ; x = rlxgr
  top    ; x = rlygr
  front  ; z = ribzgr
  back   ; z = rlxgr

; Non-Rectangular domain

nbnd  ; boundary info for ilnbnd boundary segments

; Logical boundary variables

llcstb ; t if coefficients of the boundary conds are all constants
lldrch ; t if all are dirichlet
llneum ; t if all are neumann
lldnmm ; t if all are either dirichlet or neumann
lldcpp ; t if all are either dirichlet, neumann, or periodic
lltrv  ; t if all are trivially uncoupled
llprdx ; t if periodic in x
llprdy ; t if periodic in y
llprdz ; t if periodic in z
llprdc ; t if all are periodic
llmixd ; t if not (lldrch or llneum or llprdc)
llhmc  ; t if all are homogeneous
lltang ; t if tangential derivative components in BC

; Grid

(literalize Grid)

rlxgr ; initial x point
rlygr ; initial y point
rlzgr ; initial z point
rlbxgr ; final x point
rlbygr ; final y point
rlbzgr ; final z point
rlhxgr ; average x spacing
rlhygr ; average y spacing
rlhzgr ; average z spacing
llngrx ; number of x points
llngry ; number of y points
llngrz ; number of z points
llpxx2 ; t if llngrx equal (2**k + 1) for k>0
llpxy2 ; t if llngry equal (2**k + 1) for k>0
llunfg ; uniform
llunfx ; uniform in x
llunfy ; uniform in y
llunfz ; uniform in z
status ; either just_created, new, refined

Figure 4.5. Elliptic Expert OPS5 Declarations. These declarations define the element classes describing the boundary conditions and the grid for a problem with a rectangular domain.
point-star. Note that if one of these rules fires, it merely creates an instance of the element class discretization for the module 5-point-star in OPS5 working memory. Other rules use these instances to achieve other goals; e.g., to print_discretizations or to select_best_discretization.

(vector-attribute options)

; Discretization

(literalize Discretization
  name ; ELLPACK discretization module name
  status ; either just created, new, changed, possible
  sort ; used for ordering Discretizations
  ilmno ; maximum number of coefficients per equation
  ilmneq ; maximum number of equations
  ilkban ; bandwidth of the resulting linear system
  ilwork ; amount of workspace needed
  time ; time = ilkban * ilkban * ilmnoq
  order ; order of accuracy
  llsymm ; t if generated linear system is symmetric
  options ; options to be used
)

Figure 4.6. Elliptic Expert OPS5 Declarations. This declaration defines the element classes describing discretization modules.

A typical scenario for the use of the expert system part of Elliptic Expert is the following:
1. The user asks for advice (in a FORTRAN environment).
2. Elliptic expert writes a file containing the OPS5 "makes" describing the elliptic problem.
3. Elliptic Expert invokes ("execs") the expert system which is Franz LISP with OPS5, the Elliptic Expert top level and the Elliptic Expert rules preloaded.
4. The user asks for applicable discretizations.
5. Elliptic Expert makes print_discretizations and select_applicable_discretization goals (now in a Franz LISP environment).
6. Elliptic Expert runs the OPS5 inference engine which prints the desired information.
7. The user requests to exit.
8. Elliptic Expert exits Franz LISP environment back to FORTRAN environment, and the user continues with the session.

5. Conclusions

When completed, Elliptic Expert will serve in a number of roles. Obviously, it will function as a powerful tool for engineers and scientists who solve elliptic partial differential equations. Since the rules are compartmentalized by goals (e.g., "select_best_discretization"), Elliptic Expert will provide elliptic researchers with an environment for experimenting with writing rules for production systems. Most importantly, since the computing done is typical of scientific computing in general, Elliptic Expert will serve as a model for other expert systems in scientific computing.

6. References

Figure 4.7. Elliptic Expert OPS5 Rules. These are a subset of the rules for achieving the goal select_applicable_discretization for the module 5-point-star. Note that individual attributes of an element class are accessed by preceding the class name with a "\"."


