Programming ELLPACK Examples in PDEQSOL

Yukio Umetani

Report Number:
92-080
PROGRAMMING ELLPACK EXAMPLES IN PDEQSOL

Yukio Umetani

CSD-TR-92-080
October 15, 1992
Programming ELLPACK Examples in PDEQSOL

Yukio Umetani
Hitachi Central Research Laboratory
Computer Science Department
Purdue University
October 21, 1992

Contents

1 Introduction 3
2 Language Concept and Usage 3
3 Results 7
   3.1 Programmability ........................................... 7
   3.2 Numerical Accuracy ......................................... 11
   3.3 Performance ................................................ 13
4 Conclusion 15

A Recoding of the ELLPACK examples into PDEQSOL along with a few cases of output displays. 17

Abstract

Twenty ELLPACK examples covering various aspects of PDE problems are recoded into PDEQSOL, and comparisons of programmability, numerical accuracy, and performance are made between the two solver systems. As a result, it becomes clear that ELLPACK is more flexible in treating boundary shapes, mesh sizes, discretization methods, and solver algorithms, while PDEQSOL has advantages in flexible description of numerical schemes, and straightforward treatment of time dependent problems and multiple equation problems. Numerical accuracy depends on the discretization method applied and ELLPACK has a rich repertoire. But PDEQSOL shows better accuracy in tightly coupled multiple equation problems. ELLPACK's preprocessing time is extremely short compared to PDEQSOL reflecting the modular structure of ELLPACK language. The execution time varies depending on the problems.
1 Introduction

ELLPACK ([1], [3]) is a high level PDE solver for elliptic problems developed by a cooperative research project of Purdue University, the University of Texas at Austin, Yale University, and others. PDEQSOL ([1], [4], [2]) is also a high level PDE solver developed at Hitachi Ltd, Japan. Both have common bases as PDE solvers, but differ in many ways reflecting the differences of their original purposes, namely ELLPACK as the test bed for numerical software, while PDEQSOL as the high level language for supercomputer applications.

The purpose of this report is to make the resemblances and the differences of both softwares clear through the test coding of the same benchmark problems.

The author chose as the benchmark problems all the problems listed in the book [3], these are ELLPACK examples with source code and computation results, and tried code them into PDEQSOL. Because these problems represent many aspects of PDE problems in good balance, the author believes this choice is reasonable for the first trial. But, more complex problems should be added for the second trial to reflect the original intentions of PDEQSOL.

Though both solvers are equipped with advanced features like a visual interface, parallel computation, and numerical expertise, only the basic core functions are used in this trial. Also, only the finite element methods, FEM, discretization is used for PDEQSOL, because the finite difference method (FDM) is not currently operational at Purdue.

First we compare the programming concepts of PDEQSOL with those of ELLPACK in the next chapter, then compare the programmability, numerical accuracy, and the performance for each problem in Chapter 3. Whole PDEQSOL source codes and a part of computation results are included in the appendix. Readers can refer to this information for the detailed comparison with the equivalent information in the book [3].

2 Language Concept and Usage

An ELLPACK program consists of several segments like EQUATION., BOUNDARY., GRID., DISCRETIZATION., SOLUTION., OUTPUT., etc. ([3]), a PDEQSOL program consists of statements such that DOMAIN., REGION., BCOND., MESH., SOLVE., WRITE ([4]). The functional correspondences of segment and statement are summarized in Table 1. We can see a clear correspondence between them.

To investigate the correspondence in more detail, let us pick one example. Figure 1 shows the ELLPACK program and corresponding PDEQSOL program for Example 1.E1 in the book [3].

ELLPACK’s OPTIONS. to monitor the execution time and memory requirement has no counterpart in PDEQSOL. ELLPACK’s EQUATION. and SOLUTION. correspond to PDEQSOL’s three lines beginning from “SOLVE U OF” and “BY ‘GAUSS’ ”. ELLPACK uses reserved names U, UXX, etc. for unknown quantities and a fixed form of the equation, but PDEQSOL allows more freedom at this point because any names and equation forms are permitted under certain restrictions. Also several equations can be treated in a single program. Moreover, PDEQSOL allows the programmer to write down any procedure to solve the equation in the scheme block between SCHEME and END SCHEME. ELLPACK’s BOUNDARY. corresponds to two items of PDEQSOL, namely REGION and BCOND. ELLPACK has a very elegant way to specify boundary shape and boundary condition in one line. Also ELLPACK can define many kinds of boundary curves in a simple way using curve formulas. PDEQSOL is awkward in this point because it relies on spline approximation except for lines and arcs(parts of a circle). ELLPACK’s GRID. corresponds to MESH in PDEQSOL. But, their meshing is different,
Table 1: Correspondence of the language constructs between ELLPACK and PDEQSOL.

<table>
<thead>
<tr>
<th>Category</th>
<th>ELLPACK</th>
<th>PDEQSOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape</td>
<td>BOUNDARY.</td>
<td>DOMAIN, TIME</td>
</tr>
<tr>
<td></td>
<td>HOLE.</td>
<td>REGION, FREGION</td>
</tr>
<tr>
<td></td>
<td>ARC.</td>
<td>BOUND</td>
</tr>
<tr>
<td>Mesh</td>
<td>GRID.</td>
<td>MESH, TSTEP</td>
</tr>
<tr>
<td>Discretization</td>
<td>DISCRETIZATION.</td>
<td>METHOD, ELMTYPE</td>
</tr>
<tr>
<td></td>
<td>INDEXING.</td>
<td></td>
</tr>
<tr>
<td>Variables &amp;</td>
<td>Reserved Names</td>
<td>VAR, SVAR, VEC, TENS</td>
</tr>
<tr>
<td>Constants</td>
<td>(U,UX,UY,UXX,...)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SUBPROGRAMS.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(function)</td>
<td></td>
</tr>
<tr>
<td>Equation &amp;</td>
<td>EQUATION.</td>
<td>EQU</td>
</tr>
<tr>
<td>Boundary</td>
<td>BOUNDARY.</td>
<td>BCOND</td>
</tr>
<tr>
<td>Condition</td>
<td>SOLUTION. PROCEDURE. SET U</td>
<td>SOLVE, Assignment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ICOND</td>
</tr>
<tr>
<td>Control</td>
<td>FORTRAN.</td>
<td>SCHEME, END SCHEME</td>
</tr>
<tr>
<td></td>
<td>ITER, END ITER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IF THEN, ELSE, ENDIF</td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td>FORTRAN.</td>
<td>CALL(arguments),</td>
</tr>
<tr>
<td>with FORTRAN</td>
<td>(reserved name,</td>
<td>Function call( arguments),</td>
</tr>
<tr>
<td></td>
<td>interface table)</td>
<td>EPROC, EFUNC</td>
</tr>
<tr>
<td></td>
<td>SUBPROGRAMS.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DECLARATIONS.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GLOBAL.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OPTIONS.</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>OUTPUT.</td>
<td>PRINT, WRITE</td>
</tr>
<tr>
<td>Others</td>
<td>OPTIONS.</td>
<td>*OPTION</td>
</tr>
<tr>
<td></td>
<td>END.</td>
<td>END</td>
</tr>
</tbody>
</table>
ELLPACK

OPTIONS. TIME & 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.

PDEQSOL

PROG EX1E1;
METHOD FEM;
DOMAIN X = [0.0:1.0], Y = [-1.0:2.0];
POINT A = (0,-1), B = (1,-1), C = (1, 2), D = (0, 2);
REGION AB = LN(A,B), BC = LN(B,C),
CD = LN(C,D), DA = LN(D,A),
ABCD = QUAD(AB,BC,CD,DA);
REGION ABCD;
BOUND AB + BC + CD + DA;
MESH AB = D(A,5), BC = D(B,5),
CD = D(C,5), DA = D(D,5),
INSIDE = AUTOSIM;
VAR U;
CONST PI = 3.14159265;
BCOND U = 0 AT DA,
U = SIN(PI*X)-X/2.0 AT AB,
U = Y/2.0 AT BC,
U = X AT CD;
SCHEME;
SOLVE U OF
LAPL(U) + 3.0*DX(U) - 4.0*U = EXP(X+Y)*SIN(PI*X)
BY 'GAUSS';
PRINT U;
WRITE U TO FILE11;
END SCHEME;
END;

Figure 1: Source Program for Example 1.E1.
especially for nonrectangular regions. ELLPACK always generates a rectangular grid. So, the node points do not always lie on the boundary curve. The boundary condition is treated by a sophisticated interpolation technique [(3)]. But, PDEQSOL generates nonrectangular grids along the boundary curve, and a boundary node is guaranteed to be on the boundary curve. Also, PDEQSOL allows finer control of the grid spacing by the programmer based on considering the nature of the problem. DISCRETIZATION. in ELLPACK corresponds to METHOD in PDEQSOL, but PDEQSOL has fewer choices of discretization than ELLPACK. Instead, PDEQSOL has a function to control interpolation orders of the basis functions and an up-wind discretization facility for each term to achieve numerical stability. The PDEQSOL strategy is to compensate for the fewer choices of discretization by using high order basis functions and fine meshing.

PDEQSOL’s MESH and METHOD is fixed for the entire execution and lacks the dynamic treatment of ELLPACK. This is the opposite of the treatment of EQUATION. This nature comes from the difference of language architecture mentioned later. ELLPACK’s OUTPUT. TABLE corresponds to PRINT in PDEQSOL, and PLOT corresponds to WRITE. The known quantities like PI must be defined in CONST statement of PDEQSOL. They usually correspond to the function definitions of the SUBPROGRAM. segment of ELLPACK. But complicated functions need to be defined as external functions in the same way as ELLPACK. The role of reserved names X, Y, T are just the same for both languages.

A fundamental difference which is not clear from this example is the relation to the FORTRAN language. ELLPACK is a language embedded in FORTRAN, while PDEQSOL is an independent language.

Each segment of ELLPACK is just like an island in a FORTRAN sea. Each has its own modular function and communicate with FORTRAN through reserved names and the information and data structures called the “interface”. This structure allows the extendability of the language and enables the flexible control of algorithms (once users become familiar with this interface, which is not a very easy task). Segments like DECLARATIONS., GLOBAL. are included in ELLPACK to assist the modification of the interface structure. This way, the dynamic change of shape parameters, mesh sizes and discretization methods becomes possible in ELLPACK, satisfying its requirements to be a test bed. This structure is also partly the reason for the restricted equation form in ELLPACK and for the uniform treatment of boundary shapes by interpolation techniques.

The ELLPACK preprocessor replaces each occurrence of an ELLPACK segment by a few FORTRAN statements and subroutine calls. Therefore, the preprocessing load is very light. The performance at execution time relies on the performance of each library module.

On the contrary, PDEQSOL is an independent language in itself. Communication with FORTRAN is done through the arguments of external functions or subroutine calls, or external datafiles. Thus, PDEQSOL has a lot of control statements like ITER, END ITER, IF THEN/ELSE, CALL in its own language. The PDEQSOL translator generates optimized FORTRAN code from the combination of statements. Parameterization is only possible for quantities, not for shapes, mesh sizes or the discretization method. PDEQSOL aims at better execution performance while sacrificing flexibility.

Other differences in design are as follows. The HOLE. and ARC. in ELLPACK are treated in a more generic manner in PDEQSOL. But, ARC. has no counterpart. INDEXING. in ELLPACK has no counterpart in PDEQSOL. Also, SET U ... used for the initialization of values is programmed in the scheme block of PDEQSOL. But the BLENDING function of ELLPACK has no counterpart in PDEQSOL. PROCEDURE(DISPLAY MATRIX PATTERN, EIGENVALUES) has no counterpart in PDEQSOL.
3 Results

In this chapter, the results of reading the examples into PDEQSOL are reported and compared with these ELLPACK programs. The problem examples chosen are listed in Table 2.

These examples were originally prepared to show the capability of ELLPACK. Examples 1.E1 and 3.A1 are simple boundary value problems with rectangular and non-rectangular boundary shapes. Examples 3.B1 and 3.C1 treat similar problems, but the focus is on the dynamic change of the discretization method or the solver stopping criterion. Example 4.D1 treats the parameterization of boundary shapes and the mathematical model. Example 4.D2 shows the zero patterns and eigenvalues of the discretization matrices. Example 4.D3 treats a nonlinear equation by Picard iteration. Examples 4.D4, 5.A1, and 5.A2 are boundary value problems with an elliptic domain. The treatment of holes and arcs are the main concerns. Examples 5.B1 and 5.B2 involve a two-phase diffusion problem. The way to change the equation in different regions is the main concern. Examples 5.C1 to 5.C4 treat a nonlinear equation again but by Newton iteration this time. Also the change of the grid size is treated in 5.C4. Example 5.D1 is the time dependent problem. Examples 5.E1, 5.E2, and 5.E6 involve two simultaneous equations.

3.1 Programmability

The results of the test recoding by PDEQSOL are summarized in Table 2. In general, all the discretization methods were changed to Galerkin Linear or Quadric due to the current capability of PDEQSOL at Purdue. Allowing for this change, the original intention of the problems was completely maintained in 13 examples out of 20, and in 5 examples partially maintained. The main reason of partial fulfillment is the lack of the facility to change the grid size or discretization method dynamically. Therefore, separate programs were coded for each grid size or discretization method. Two examples (4.D2, 5.C2) could not be coded by PDEQSOL due to the lack of corresponding functionality.

Let us examine the individual coding in more detail. Complete documentation is contained in the Appendix. Example 1.E1 is as listed in Figure 1.

In Example 3.A1, the bottom line constituting a part of the boundary is expressed as DA = LN(D,A) in PDEQSOL in place of X = 1.-S,Y = -S FOR S = 0. TO 1. in ELLPACK.

In Example 3.B1, the FORTRAN functions TRUE and F in ELLPACK are defined as CONST in PDEQSOL. Also, instead of UXX + (1.0+Y**2)*UYY in ELLPACK, the expression DIV(CT ... GRAD(U)) with CT=(1.0,0,1.0+Y**2) is used in PDEQSOL because UXX,UYY, and UXY are not allowed in the FEM version of PDEQSOL. Neumann boundary conditions need to be given in the form N(CT ... GRAD(U))= ... where N(CT ... GRAD(U)) means the outward normal vector component of CT ... GRAD(U) at the boundary. Also we must be careful to adjust the coefficient of UY from -(1.0+Y**2) to -(1.0+2.*Y +Y**2), because DIV(CT ... GRAD(U)) differs from UXX+(1.0+Y**2)*UYY by 2.0*Y*UY. This is an error prone pitfall of FEM PDEQSOL. But, the physical meaning is rather clear in this form. Also, separate programs were coded for each discretization method, as noted above.

In Example 3.C1, the FORTRAN functions W and TRUE are defined as CONST, but F is defined as the external function EFUNC because it is too lengthy for CONST. External functions and their arguments need to be declared in double precision, as PDEQSOL treats all data in double precision.
Table 2: Summary of the benchmark problems and the results of reprogramming them in PDEQSL.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Rate of Function Realization</th>
<th>Main Changes (* Serious changes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.E1</td>
<td>Simple Poisson equation. Rectangular domain.</td>
<td>Complete 5 point star → Galerkin(Lin).</td>
<td></td>
</tr>
<tr>
<td>4.D1</td>
<td>Parameter study for alloy solidification. Dynamic change of shape and model parameters.</td>
<td>Partial 5 point star → Galerkin(Lin). *Shape parameters are fixed.</td>
<td></td>
</tr>
<tr>
<td>4.D2</td>
<td>Zero pattern and Eigen values of Matrices</td>
<td>Non</td>
<td></td>
</tr>
</tbody>
</table>
Table 2: Summary of the benchmark problems and the results of reprogramming by PDEQSOL (continued).

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Rate of Function Realization</th>
<th>Main Changes (* Serious changes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.C2</td>
<td>MACSYMA program to derive linearized equation.</td>
<td>Non</td>
<td></td>
</tr>
</tbody>
</table>
Table 2: Summary of the benchmark problems and the results of reprogramming by PDEQSOL (continued).

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Rate of Function Realization</th>
<th>Main Changes (* Serious changes)</th>
</tr>
</thead>
</table>

The expression \(( W(X,Y)\cdot UX )_X + ( W(X,Y)\cdot UY )_Y \) precisely corresponds to \( \text{DIV}(W \cdot \text{GRAD}U) \) in PDEQSOL. The FORTRAN loop \DO 100 in ELLPACK is expressed by \ITER NZETA ..., END ITER loop in PDEQSOL, but a trivial constraint in PDEQSOL that a variable name cannot be passed to the convergence criterion of the solver prevents the complete coding of this loop. This constraint is to be removed in December, 1992. ELLPACK's JACOBI CG does not exist in the PDEQSOL repertoire, so the PCG(Preconditioned Conjugate Gradient) is used instead of it.

In Example 4.D1, PROC EX4D1(BETA) is used to parameterize BETA. PROC makes this procedure a subroutine. This subroutine is called from the FORTRAN main program EX4D1F. The variables YINF and DELTA are not parameterized by PDEQSOL because they are related to the boundary shape. The curve representing a solidification front is approximated by a spline curve in PDEQSOL. It takes a lengthy specification.

Example 4.D2 can not be programmed into PDEQSOL's normal framework. One must prepare a special external procedure accessing the matrix table of generated FORTRAN code to realize the equivalent functionality.

Example 4.D3 can be smoothly translated to PDEQSOL. Picard iteration is expressed by \ITER, END ITER loop in PDEQSOL. One needs to distinguish the input variable \( UOLD \) from the output variable \( U \) and update \( UOLD \) for each iteration. The built-in procedure NORMM calculates the maximum norm of the second argument. SET U BY BLENDING has no equivalent in PDEQSOL, thus \( UOLD \) is initialized to zero at the beginning of the iteration.

In example 4.D4, the boundary line is approximated by a spline curve in PDEQSOL because it is an elliptic curve. Also the meshing pattern is different from ELLPACK (see Appendix). PDEQSOL discretizes only the interior of the target region. Grid points at the boundary are always placed on the boundary.

In Examples 5.A1 and 5.A2, ARC in ELLPACK is approximated by a thin diamond hole by PDE-
QSOL. The boundary line is approximated by spline curve. The meshing is different from ELLPACK.

Examples 5.A2 and 5.B1 were easily translated to PDEQSOL except for the boundary curve and meshing pattern.

Example 5.B2 is coded by adopting different schemes from ELLPACK. The author devised an iterative scheme to alternatingly solve the different equations on the two neighboring subregions while keeping the boundary conditions between them (see Appendix). This scheme is easier to understand and code than the methods adopted by ELLPACK which directly modifies the interface table.

Example 5.C1 is coded by PDEQSOL using the ITER and END ITER loops.

Example 5.C2 (MACSYMA program to derive the linearized equation by applying Newton’s method) is not coded by PDEQSOL because the corresponding functionality is lacking.

Example 5.C3 is coded without using the FORTRAN function. These are defined by the CONST definitions or the assignment statements within the scheme block.

Example 5.C4 is also coded in a straightforward manner, but the grid is fixed at 12*8.

In Example 5.D1 (the time dependent program), the Crank-Nicolson scheme is expressed as

\[
\text{SOLVE U OF} \\
2./DLT*U = 2./UO + \text{DIV(CT..GRAD(U))} + DN*DY(U) + U + F(X,Y,T) \\
+ \text{DIV(CT..GRAD(UO))} + DNO*DY(UO) + UO + F(X,Y,T-DLT) \\
\text{BY 'GAUSS'}
\]

by PDEQSOL, preserving the original form of this scheme. This example takes advantage of the PDEQSOL ability to accept an arbitrary form of the equation.

In Example 5.E1 and 5.E2 (two equation problems), PDEQSOL can write a Gauss-Seidel like iterative procedure in the original form without switching the parameter KEQN as is necessary for ELLPACK. This is also an advantage of the PDEQSOL.

In Example 5.E6, PDEQSOL adopts the alternate approach described on p. 121 of the book [3]. Better results were obtained by comparing the results of Gauss-Seidel like iterations, see Table 3.

3.2 Numerical Accuracy

Numerical accuracy for these benchmark problems is summarized in Table 3 in those cases where comparisons are possible. In Table 3, ERMAX indicates the maximum absolute value of the difference between the true solution and the computed solution for all grid points unless otherwise stated.

Differences in the conditions related to the accuracy are:

1. Precision: ELLPACK is single, PDEQSOL double.

2. Discretization method: ELLPACK uses 5 Point Star, Hermite Collocation, or Hodie Helmholtz and PDEQSOL uses Galerkin Linear or Quadruple. 5 Point Star and Galerkin Linear both have first order accuracy in the grid spacing, while Galerkin Quadruple, Hermite Collocation, and Hodie Helmholtz, have second, third, and fourth order of accuracies.

3. The mesh decomposition for nonrectangular domain and the treatment of boundary conditions are usually different.

As is shown in Table 3, ELLPACK and PDEQSOL have the same order of accuracy except for a few problems. ELLPACK shows better accuracy in examples 4.D3, 5.C1, 5.C3, and 5.E1. This can
Table 3: Comparison of numerical accuracy achieved by the two PDE problem solvers.

<table>
<thead>
<tr>
<th>Problem #</th>
<th>Accuracy (ERMAX)</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ELLPACK</td>
<td>PDEQSOL</td>
</tr>
<tr>
<td>1.E1</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>3.A1</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>4.262 E-2</td>
<td>2.718 E-2</td>
</tr>
<tr>
<td></td>
<td>Hermite colloc.</td>
<td>Galerkin(Quad)</td>
</tr>
<tr>
<td></td>
<td>4.096 E-4</td>
<td>4.523 E-4</td>
</tr>
<tr>
<td>3.C1</td>
<td>5 point star</td>
<td>Galerkin(Lin)</td>
</tr>
<tr>
<td></td>
<td>1.135 E-1</td>
<td>3.990 E-3</td>
</tr>
<tr>
<td></td>
<td>1.135 E-1</td>
<td>3.948 E-3</td>
</tr>
<tr>
<td></td>
<td>1.134 E-1</td>
<td>3.949 E-3</td>
</tr>
<tr>
<td>4.D1</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>4.D2</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>4.D3</td>
<td>Hodie Helmholtz</td>
<td>Galerkin(Quad)</td>
</tr>
<tr>
<td></td>
<td>0.596 E-5</td>
<td>0.939 E-4</td>
</tr>
<tr>
<td>4.D4</td>
<td>5 point star</td>
<td>Galerkin(Lin)</td>
</tr>
<tr>
<td></td>
<td>1.476 E-3</td>
<td>4.451 E-3</td>
</tr>
<tr>
<td>5.A1</td>
<td>5 point star</td>
<td>Galerkin(Lin)</td>
</tr>
<tr>
<td></td>
<td>1.598 E-1</td>
<td>1.275 E-1</td>
</tr>
<tr>
<td>5.A2</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>5.B1</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>5.B2</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>5.C1</td>
<td>Hermite colloc.</td>
<td>Galerkin(Quad)</td>
</tr>
<tr>
<td></td>
<td>3.576 E-6</td>
<td>2.903 E-4</td>
</tr>
<tr>
<td></td>
<td>6.188 E-5</td>
<td></td>
</tr>
<tr>
<td>5.C2</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
Table 3: Comparison of numerical accuracy achieved by the two PDE problem solvers (continued).

<table>
<thead>
<tr>
<th>Problem #</th>
<th>Accuracy (ERMAX)</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.C3</td>
<td>Hermite colloc.</td>
<td>Galerkin (Lin)</td>
</tr>
<tr>
<td></td>
<td>2.808 E-3</td>
<td>5.972 E-2</td>
</tr>
<tr>
<td></td>
<td>DFMAX (7.153 E-7)</td>
<td>(3.297 E-7)</td>
</tr>
<tr>
<td>5.D1</td>
<td>Interior colloc.</td>
<td>Galerkin (Quad)</td>
</tr>
<tr>
<td></td>
<td>1.561 E-3</td>
<td>2.678 E-3</td>
</tr>
<tr>
<td></td>
<td>1.701 E-3</td>
<td>1.813 E-3</td>
</tr>
<tr>
<td>5.E1</td>
<td>Hermite colloc.</td>
<td>Galerkin (Quad)</td>
</tr>
<tr>
<td></td>
<td>V 0.109 E-3</td>
<td>1.174 E-3</td>
</tr>
<tr>
<td></td>
<td>W 0.227 E-4</td>
<td>5.215 E-4</td>
</tr>
<tr>
<td>5.E2</td>
<td>5 point star</td>
<td>Galerkin (Lin)</td>
</tr>
<tr>
<td></td>
<td>V 0.463 E-2</td>
<td>0.154 E-2</td>
</tr>
<tr>
<td></td>
<td>W 0.575 E-3</td>
<td>1.392 E-3</td>
</tr>
<tr>
<td>5.E6</td>
<td>Hermite colloc.</td>
<td>Galerkin (Quad)</td>
</tr>
<tr>
<td></td>
<td>Alternating solve</td>
<td>Simultaneous solve</td>
</tr>
<tr>
<td></td>
<td>U 15.77</td>
<td>5.879 E-2</td>
</tr>
<tr>
<td></td>
<td>V 0.511</td>
<td>3.096 E-1</td>
</tr>
</tbody>
</table>

be attributed to the difference in accuracy of the discretization methods used. On the other hand, PDEQSOL shows better accuracy in Example 3.C1 and 5.E6. The former shows the effect of the Galerkin method for self adjoint operators, and the latter is attributed to the advanced simultaneous solving facility of PDEQSOL.

3.3 Performance

The timing data for translation, compilation & link, and execution is summarized in Table 4. Here the data for the PDEQSOL translation is measured on the HP9000/370. All other data are measured on a Sun/SPARC workstation. An internal monitor is used for the measurement of the ELLPACK translation and execution time. Other data are measured by the UNIX TIME command.

As shown in Table 4, translation by ELLPACK is extremely fast compared to PDEQSOL, even taking into account the difference of machine speeds of the Sun/SPARC and HP9000/370 (in nonnumeric processes, the Sun/SPARC is observed to be 1.5 to 2.0 times faster than the HP9000/370). This is due to the difference of the language structure as explained in Chapter 2. Compile & link time is
Table 4: Performance comparison.

<table>
<thead>
<tr>
<th>Problem #</th>
<th>Processing Time (seconds)</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Translation</td>
<td>Compile &amp; Link</td>
</tr>
<tr>
<td>1.E1</td>
<td>E</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>(9.4)</td>
</tr>
<tr>
<td>3.A1</td>
<td>E</td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>(9.5)</td>
</tr>
<tr>
<td>3.B1</td>
<td>E</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>(13.2)</td>
</tr>
<tr>
<td>3.C1</td>
<td>E</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>(19.7)</td>
</tr>
<tr>
<td>4.D1</td>
<td>E</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>(26.7)</td>
</tr>
<tr>
<td>4.D2</td>
<td>E</td>
<td>0.83</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>4.D3</td>
<td>E</td>
<td>0.83</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>(16.4)</td>
</tr>
<tr>
<td>4.D4</td>
<td>E</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>(66.9)</td>
</tr>
<tr>
<td>5.A1</td>
<td>E</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>(39.9)</td>
</tr>
<tr>
<td>5.A2</td>
<td>E</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>(47.4)</td>
</tr>
<tr>
<td>5.B1</td>
<td>E</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>(10.6)</td>
</tr>
<tr>
<td>5.B2</td>
<td>E</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>(28.6)</td>
</tr>
</tbody>
</table>

14
Table 4: Performance comparison (continued).

<table>
<thead>
<tr>
<th>Problem #</th>
<th>Processing Time (seconds)</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Translation</td>
<td>Compile &amp; Link</td>
</tr>
<tr>
<td>5.C1 E</td>
<td>0.85</td>
<td>8.8</td>
</tr>
<tr>
<td></td>
<td>D (18.2)</td>
<td>21.7</td>
</tr>
<tr>
<td>5.C2 E</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>D —</td>
<td>—</td>
</tr>
<tr>
<td>5.C3 E</td>
<td>0.95</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td>D (38.8)</td>
<td>41.7</td>
</tr>
<tr>
<td>5.C4 E</td>
<td>1.02</td>
<td>11.1</td>
</tr>
<tr>
<td></td>
<td>D (22.5)</td>
<td>23.9</td>
</tr>
<tr>
<td>5.D1 E</td>
<td>0.95</td>
<td>10.3</td>
</tr>
<tr>
<td></td>
<td>D (11.6)</td>
<td>13.4</td>
</tr>
<tr>
<td>5.E1 E</td>
<td>1.00</td>
<td>8.8</td>
</tr>
<tr>
<td></td>
<td>D (25.2)</td>
<td>27.7</td>
</tr>
<tr>
<td>5.E2 E</td>
<td>1.00</td>
<td>9.1</td>
</tr>
<tr>
<td></td>
<td>D (30.4)</td>
<td>27.8</td>
</tr>
<tr>
<td>5.E6 E</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>D (19.8)</td>
<td>24.7</td>
</tr>
</tbody>
</table>

of the same order for both systems. The execution time varies according to the programs. In general, ELLPACK is faster than PDEQSOL in examples which use high accuracy discretization like 4.D3, 5.C1, and 5.D1, involving specialized discretization modules. Hordie Helmholtz in Example 4.D3 is very fast. Example 5.C3 is the exception. Examples 5.B2 and 5.E2 show less efficiency in PDEQSOL along with the improved programmability.

4 Conclusion

As is shown in the above analysis, the current PDEQSOL lacks in the flexibility to alter boundary shapes, mesh sizes and discretization methods as ELLPACK can do. The first two are important functions for a practical solver like PDEQSOL. Also, the specification of boundaries using a curve function is a convenient facility which the current PDEQSOL does not have.

On the other hand, PDEQSOL has the advantage in flexible descriptions of numerical schemes and
high flexibility in the SOLVE statement. As a result, the straightforward treatment of time-dependent problems or two equations become possible. PDEQSOL has advantages in controlling the discretization accuracy including changing the interpolation order of basis functions, up-wind techniques, and fineness control of the mesh: (although these facilities are not used in the examples in this report).

The numerical accuracy of both solvers are of the same order except for a few examples. In some cases, ELLPACK shows better accuracy from the rich repertoire of the discretization method. But PDEQSOL shows better accuracy in the tightly coupled multi-equation problem from the advanced solve facility of PDEQSOL.

The preprocessing by ELLPACK is quite fast reflecting the modular structure of the ELLPACK language. Compiling time is of the same order for both ELLPACK and PDEQSOL. The execution time differs according to the functions used in the program. In general, ELLPACK is fast when sophisticated high accuracy discretization is used, or when the programmer carefully applies it to the advanced problems.

Though there exist differences as stated above, it becomes clear that both solvers have many features in common, and are applicable to a broad range of problems.

Acknowledgments

The author expresses his gratitude to Dr. John R. Rice of the Computer Science Department, Purdue University for providing him with the opportunity for this work, as well as for discussions on the results of this work. The author also thanks Mr. Chisato Konno at Hitachi Central Research Laboratory for various comments to this report.

References


APPENDIX A: Recoding the ELLPACK examples into PDEQSOL.

The PDEQSOL code for the examples is given along with a few cases of output displays.
PROG EX3B1;
/* ELLPACK Example 3.B1 */
/* Mixed Boundary Condition */
/* Iteration by Changing Discretization */
/* Linear Element */
/* Coded by Y. Umetani June 29, 1992 */
METHOD FEM;

DOMAIN X=[0:1.0], Y=[0:1.0];
POINT A=(0,0), B=(1.0,0), C=(1.0,1),
D=(0,1);
REGION AB=LN(A,B), BC=LN(B,C),
CD=LN(C,D), DA=LN(D,A),
ABCD=QUAD(AB,BC,CD,DA);
FREGION ABCD;
BOUND AB+BC+CD+DA;
MESH AB=D(A,3), BC=D(B,4),
CD=D(C,3), DA=D(D,4),
INSIDE=AUTOSIM;
VAR U,ERROR;
SVAR TN2,TNM,EN2,ENM;
CONST TRUE=EXP(X+Y)+((X*(X-1.0))**2)*LOG(1.0+Y**2),
F=LOG(1.0+Y**2)*((2.0+X+(14.0+X**2)*(15.0-4.0*X)));
C1=1.0+Y**2; C2=1.0+2.0*Y+Y**2;
CTENS CT=(1,0,0,C1);
CVEC V=(1,C2);
BOUND N(CTOGRAD(U))=U+20.0*EXP(Y) AT DA,
U=TRUE AT AB,
N(CTOGRAD(U))=U AT BC,
U=TRUE AT CD;
SCHEME;
CALL NORM2(TN2,TRUE); CALL NORMM(TNM,TRUE);
TN2=TN2/SQRT(20.0);
PRINT TN2,TNM;
SOLVE U OF
DIV(CTOGRAD(U))=VTOGRAD(U) = F
BY 'GAUSS';
PRINT U;
WRITE U to FILE11;
WRITE TRUE to FILE12;
ERROR=TRUE;U;
CALL NORM2(EN2,ERROR); CALL NORMM(ENM,ERROR);
EN2=EN2/SQRT(20.0);
WRITE ERROR to FILE13;
PRINT EN2,ENM;
END SCHEME;
END;
PROG EX3Bl;
/* ELLPACK Example 3Bl */
/* Mixed Boundary Condition */
/* Iteration Changing Discretization */
/* Quadric Element */
/* Coded by Y. Umelani July 14,1992 */
METHOD FEM;
DOMAIN X=[0:1.0], Y=[0:1.0];
POINT A=(0.0), B=(1.0,0), C=(1.0,1),
D=(0.1);
REGION AB=LN(A,B), BC=LN(B,C),
CD=LN(C,D), DA=LN(D,A),
ABCD=QUAD(AB,BC,CD,DA);
REGION ABCD;
BOUND AB+BC+CD+DA;
MESH AB=D(A,3), BC=D(B,4),
CD=D(C,3), DA=D(D,4),
INSIDE=AUTOSIM;
VAR U,ERROR;
SVAR TN2,TNM,EN2,ENM;
CONST TRUE=EXP(X+Y)+((X*(X-1.0))*(y**2)*LOG(1.0+Y**2),
+2.0*(X*(X-1.0))*(1.0-Y-2.0*Y**2)/(1.0+Y**2)),
C1=1.0+Y**4;
C2=1.0+2.*Y+Y**2;
CTENS CT=(I,O,O,C1);
CVEC V=(I,C2);
ELMTYPE QUADRIC for ALLNAMES;
BCOND N(CT..GRAD(U)=U.2.0*EXP(Y) AT DA,
U=TRUE AT AB,
N(CT..GRAD(U)=U AT BC,
U=TRUE AT CD;
SCHEME;
CALL NORM2(TN2,TRUE); CALL NORMM(TNM,TRUE);
TN2=TN2/SQRT(20.0);
PRINT TN2,TNM;
SOLVE U OF
DIV(CT..GRAD(U))= V ..GRAD(U) = F
BY 'GAUSS';
PRINT U;
WRITE U to FILE11;
WRITE TRUE to FILE12;
ERROR=TRUE-U;
CALL NORM2(EN2,ERROR); CALL NORMM(ENM,ERROR);
EN2=EN2/SQRT(20.0);
WRITE ERROR to FILE13;
PRINT EN2,ENM;
END SCHEME;
END;
PROG EX3Cl;
/* ELLPACK Example 3.Cl(Zeta=10D-4) */
/* Iteration Testing the Effect of Changing */
/* Stop Criterion Zeta. */
/* Coded by Y. Umemoto June 30, 1992 */
METHOD FEM;
DOMAIN X=[0.5:1.0], Y=[0.5:1.0];
SFUNC F;
POINT A=(0.5,0.5), B=(1.0,0.5), C=(1.0,1.0),
D=(0.5,1.0);
REGION AB=LN(A,B), BC=LN(B,C),
CD=LN(C,D), DA=LN(D,A),
ABCD=QUAD(AB,BC,CD,DA);
FRREGION ABCD;
BOUND AB+BC+CD+DA;
MESH AB=D(A,16), BC=D(B,16),
CD=D(C,16), DA=D(D,16),
INSIDE=AUTOSIM;
VAR U,ERROR,FV;
SVAR TN2,TNM,EN2,ENM,ZETA;
CONST PI=3.14159265358979,
W=pi**2*pi2*cos(pi*x)*sin(pi*y)**2 +
(pi*sin(pi*x)**2*cos(pi*y)**2)**0.15,
TRUE=pi*sin(pi*x)**2*sin(pi*y);*
RCOND New(WGRAD(U))=0 AT DA,
N(WGRAD(U))=0 AT AB,
U=0 AT BC,
U=0 AT CD;
COUNT NZETA;
SCHEME;
CALL NORMM(TNM,TRUE);
PRINT TNM;
/* FV=F(X,Y); PRINT FV; */
ITER NZETA UNTIL NZETA GE 3;
WRITE TRUE 10 FILE11;
ZETA=1.0*10.0**(NZETA+2);
PRINT ZETA;
/* Parameterization of EPS is not allowed right now. */
SOLVE U GP
DIV(WGRAD(U)) = F(X,Y)
BY 'PCG' WITH(EPS(1.0D-4));
WRITE U to FILE12;
ERROR=TRUE;
CALL NORMM(ENM,ERROR);
PRINT ENM;
WRITE ERROR 10 FILE13;
END ITER;
END SCHEME;
END;

function w(x,y)
double precision x,y
pi = 3.14159265358979
w = (pi**2*pi2*cos(pi*x)*sin(pi*y))**2 +
a * (pi*sin(pi*x)**2*cos(pi*y)**2)**0.15
return
end

function f(x,y)
double precision x,y
pi = 3.14159265358979
i2 = pi**2 * pi;
sinpix = sin(pi*x)
sinpy = sin(pi*y)
cospix = cos(pi*x)
cospix = cos(pi*y)
sin = sinpix*sinpy
tu = pi2*cospix*sinpy
tux = pi2*csypix
tuxx = -pi2*tu
tuy = pi2*csypix*cospy
tuyy = -pi2*tu
f = w(x,y)*(tuxx + tuyy) + cdxu(x,y)*tux + cdyu(x,y)*tuy
return

d function cdxu(x,y)
double precision x,y
pi = 3.14159265358979
rlepsm = 5.96046E-08
sinpix = sin(pi*x)
sinpy = sin(pi*y)
cospix = cos(pi*x)
cospix = cos(pi*y)
d = ((pi2*cospix*sinpy)**2 +
a * (pi2*csypix*cospy)**2)**0.5
return

END.;
20
C
end funclion cdyu(x,y)
double precision x,y
pi = 3.14159265358979
rlepam = 5.060465-08
sinpix = sin(pi*x)
sinpy = sin(pi*y)
cospix = cos(pi*x)
cospix = cos(pi*y)
d = ((pi*sinpix*sinpy)**2 + (pi*cospix*cospix)**2)**.85
return
end

PROC EX'DI(BETA);
/*
 * Ellpack Example 4.D1
 * Parameter Study for Alloy Solidification.
 * BETA is paraemetrized, but YINF and DELTA can't be */
 * right now, YINF=0.5, DELTA=0.2 assumed.
 * Coded by Y. Umezumi June 30, 1992 */
METHOD FEM;
DOMAIN X=[0:0.1>, Y=[-0.2:0.1>);
POINT A=(0,0.2), B=(0.1>,-0.2), C=(0.1>,0.1>,
D=(0,0.1>, AB1=(0.021>,0.1911>, AB2=(0.010,0,0.1902), AB3=(0.011>,0.182),
AB4=(0.100,0.1618), AB5=(0.121>,0.1182), AB6=(0.110,0.1116),
AB7=(0.111>,0.0908), AB8=(0.200,0.0618), AB9=(0.221>,0.0313),
AB10=(0.245,0.1975);
REGION AB=SPL(A,AB1,AB2,AB3,AB4,AB5,AB6,AB7,AB8,AB9,AB10,AB11,AB12,
AB13,AB14,AB15,AB16,AB17,AB18,AB19,B),
BC=LN(B,C), CD=LN(C,D), DA=LN(D,A),
ABCD=QUAD(AB,BC,CD,DA);
FREGION ABCD;
BOUND AB+BC+CD+DA;
MESH AB=D(A,10), BC=D(B,15),
CD=D(C,10), DA=D(D,15),
INSIDE=AUTOSIM;
VAR U,CN;
CONST pi=3.14159265;
SVAR BETA,BOV2,BOV2Q;
BCOND NGRAD(U)=0 AT DA,
NGRAD(U)=BETA*SINH(BOV2*Y) AT AB,
NGRAD(U)=0 AT BC,
U=0 AT CD;
SCHNE;E;
BOV2=0.5*BETA; BOV2Q=BOV2*BOV2;
SOLVE U OF
LAPL(U)-BOV2Q*U = 0.0
BY 'GAUSS';
CN=1.0+EXP(-BETA*Y)+U*EXP(-BOV2*Y);
WRITE U to FILE11;
WRITE CN to FILE12;
PRINT CN at AB;
END SCHNE;
TEND;
PROGRAM EX4D1P
DOUBLE PRECISION YINF,BETA,DELTA
YINF=0.5
BETA=2.0
DELTA=0.2
WRITE(6,2000) YINF,BETA,DELTA
2000 FORMAT('1 PARAMETERS FOR THIS RUN ARE' ,
A '/
B 'YINF = ',1PE12.5 ' /X,'BETA = ',1PE12.5 /)
C 'DELTA = ',1PE12.5 /
CALL EX4D1(BETA)
STOP
END
**PROG EX4D4**

/* ELLPACK Example 4.D4 */
/* Nonrectangular Domain with a Hole */
/* Coded by Y. Umetsu July 1, 1992 */

**METHOD FEM**

**DOMAIN**

\[ x = [0.15, 15.0], y = [-15.0, 15.0] \]

**EFUNC TRUEF**

**POINT**

\[ A = (0.1, 0), B = (0, 0.1), C = (10.0, 0), D = (0, 10.0), 
E = (5.0, 0), F = (0, -5.0), 
G = (5.0, 0), H = (0, 5.0), 
AB1 = (-9.9, -1.6), AB2 = (9.9, 1.6), AB3 = (8.9, -4.6), 
AB4 = (8.1, -5.9), AB5 = (7.1, 0.8), AB6 = (5.9, 8.1), 
AB7 = (4.6, -8.9), AB8 = (3.1, -9.5), AB9 = (1.6, -9.9), 
AD1 = (9.9, 1.6), AD2 = (9.9, 3.1), AD3 = (8.9, 4.6), 
AD4 = (8.1, 5.9), AD5 = (7.1, -0.8), AD6 = (5.9, -8.1), 
AD7 = (4.6, 8.9), AD8 = (3.1, 9.5), AD9 = (1.6, 9.9), 
CB1 = (9.9, 1.6), CB2 = (9.9, 3.1), CB3 = (8.9, 4.6), 
CB4 = (8.1, 5.9), CB5 = (7.1, -0.8), CB6 = (5.9, -8.1), 
CB7 = (4.6, 8.9), CB8 = (3.1, 9.5), CB9 = (1.6, 9.9), 
CD1 = (9.9, 1.6), CD2 = (9.9, 3.1), CD3 = (8.9, 4.6), 
CD4 = (8.1, 5.9), CD5 = (7.1, -0.8), CD6 = (5.9, -8.1), 
CD7 = (4.6, 8.9), CD8 = (3.1, 9.5), CD9 = (1.6, 9.9), 
EF1 = (-4.9, 0.8), EF2 = (4.9, 0.8), EF3 = (4.4, 2.2), 
EF4 = (4.1, -2.9), EF5 = (-3.6, -3.4), EF6 = (2.9, -3.9), 
EF7 = (2.5, 3.2), EF8 = (-2.1, 3.7), EF9 = (0.7, -4.8), 
EH1 = (-4.9, 0.8), EH2 = (4.9, 0.8), EH3 = (4.4, 2.2), 
EH4 = (4.1, -2.9), EH5 = (-3.6, -3.4), EH6 = (2.9, -3.9), 
EH7 = (2.5, 3.2), EH8 = (-2.1, 3.7), EH9 = (0.7, -4.8), 
GF1 = (4.9, 0.8), GF2 = (4.9, 0.8), GF3 = (4.4, 2.2), 
GF4 = (4.1, -2.9), GF5 = (-3.6, -3.4), GF6 = (2.9, -3.9), 
GF7 = (2.5, 3.2), GF8 = (-2.1, 3.7), GF9 = (0.7, -4.8), 
GH1 = (4.9, 0.8), GH2 = (4.9, 0.8), GH3 = (4.4, 2.2), 
GH4 = (4.1, -2.9), GH5 = (-3.6, -3.4), GH6 = (2.9, -3.9), 
GH7 = (2.5, 3.2), GH8 = (-2.1, 3.7), GH9 = (0.7, -4.8) \]

**region**

\[ A = B \cup A B_1 \cup A B_2 \cup A B_3 \cup A B_4 \cup A B_5 \cup A B_6 \cup A B_7 \cup A B_8 \cup A B_9 \cup B \]

\[ B C \cup B C_1 \cup B C_2 \cup B C_3 \cup B C_4 \cup B C_5 \cup B C_6 \cup B C_7 \cup B C_8 \cup B C_9 \cup C \]

\[ C D \cup C D_1 \cup C D_2 \cup C D_3 \cup C D_4 \cup C D_5 \cup C D_6 \cup C D_7 \cup C D_8 \cup C D_9 \cup D \]

\[ E F \cup E F_1 \cup E F_2 \cup E F_3 \cup E F_4 \cup E F_5 \cup E F_6 \cup E F_7 \cup E F_8 \cup E F_9 \cup F \]

\[ G H \cup G H_1 \cup G H_2 \cup G H_3 \cup G H_4 \cup G H_5 \cup G H_6 \cup G H_7 \cup G H_8 \cup G H_9 \cup H \]

\[ A E \cup A \cup A_1 \cup E \]

\[ B F \cup B \cup F \]

\[ C G \cup C \cup G \]

\[ D H \cup D \cup H \]

**FREGION**

\[ A B F E + B C G F + C D H G + D A H E \]

**BOUND**

\[ A B + B C + C D + D A + E F + F G + G H + H E \]

**MSHB**

\[ A E = D(A, A), B C = D(B, B) \]

\[ C D = D(C, C), D A = D(D, D) \]

\[ E F = D(E, E), F G = D(F, F), G H = D(G, G) \]

\[ A E = D(A, A), B F = D(B, B), C G = D(C, C) \]

**INSIDE**

\[ = \text{AUTOSIM} \]

**VAR**

\[ U, \text{ERROR}, \]

**SVAR**

\[ \text{ERMAX}, \text{TMAX}, \text{UMAX}, \]

**CONST**

\[ \text{TRUE} = \text{TRUEF}(X, Y) \]

**BOUND**

\[ U = 0 \]

**SCHEME**

**SOLVE**

\[ \text{U OF} \]

**LAPL(U) = 0**

**BY GAUSS**

**ERROR = U \cdot \text{TRUEF}(X, Y)**

**CALL NORMM(TMAX, TRUE); CALL NORMM(UMAX, U); CALL NORMM(ERMAX, ERROR);**

**PRINT TMAX, UMAX, ERMAX;**

**WRITE U to FILE11;**

**WRITE TRUE to FILE12;**

**END SCHEME;**

**END**

**double precision function truef(x, y)**

**double precision x, y**

\[ \text{write}(6, ") x =", x, \" , y =", y \]

\[ r1 = \text{sqr}(x+1.0) \quad r2 = \text{sqr}(x+1.0) \quad u = \text{acos}(0.5 \times (r1+r2)) \quad \text{truef} = (3.0 \times u)/(3.0 \times 2.3) \]

\[ \text{write}(6, ") truef", truef \]

**return**

**end**

**function acos(x)**

\[ \text{acos} = \text{alog}(x+\text{sqr}(x ** 2+1.0)) \]

**return**

**end**

22
PROG EX5A1;
/*
 ELLPACK Example 5.A1. */
/* Special Interior Boundary Condition - Arc */
/* Arc is modeled as a thin diamond. */
/* Coded by Y. Umetsu July 2, 1992 */
METHOD FEM;
DOMAIN X=[0.5:5.0), Y=[0.5:5.0);
EFUNC TRUEF;

POINT A=(0.3.762,0), B=(0,.3.627), C=(3.762,0), D=(0,3.627),
E=(0.1.0,0), F=(1.0,0), G=(0,.0.0), H=(0,-.0.0),
AB1=(·3.716,-0.567), AB2=(.3.578,-1.121), AB3=(.3.352,-1.647),
AB4=(·3.044,·2.132), AB5=(·2.660,-2.565), AB6=(·2.211,-2.934),
AB7=(·1.708,·3.322), AB8=(·1.163,-3.449), AB9=(·0.588,-3.582),
AB10=(·3.716,+0.567), AB11=(·3.578,+1.121), AB12=(·3.352,+1.647),
AB13=(·3.044,+2.132), AB14=(·2.660,+2.565), AB15=(·2.211,+2.934),
AB16=(·1.708,+3.322), AB17=(·1.163,+3.449), AB18=(·0.588,+3.582),
CD1=(·3.716,+0.567), CD2=(·3.578,+1.121), CD3=(·3.352,+1.647),
CD4=(·3.044,+2.132), CD5=(·2.660,+2.565), CD6=(·2.211,+2.934),
CD7=(·1.708,+3.322), CD8=(·1.163,+3.449), CD9=(·0.588,+3.582),

region ABE=SPI(A,AB1,AB2,AB3,AB4,AB5,AB6,AB7,AB8,AB9,B),
    BCF=SPI(B,BC9,BC8,BC7,BC6,BC5,BC4,BC3,BC2,BC1,C),
    CDG=SPI(C,CD1,CD2,CD3,CD4,CD5,CD6,CD7,CD8,CD9,D),
    DA=B=SPI(D,DA9,DA4,DA7,DA6,DA5,DA4,DA3,DA2,DA1),A),
    AE=LN(A,E), BE=LN(B,E), DE=LN(D,E),
    BF=LN(B,F), CF=LN(C,F), DF=LN(D,F),
    DG=LN(D,G), EG=LN(E,G), FG=LN(F,G),
    AB=TRI(A,B,E), BC=TRI(B,C,F), CD=TRI(C,D,G),
    DA=TRI(D,A), BE=TRI(B,E), DE=TRI(D,E),
    BF=TRI(B,F), CF=TRI(C,F), DF=TRI(D,F),

FREGION ABE+BCF+CDF+DAE+BEH+BFH+DEG+DFG;
BOUND AB+BC+CD+DA+EH+FH+EG+FG;
MESH AB=D(A,5), BC=D(B,5),
    CD=D(C,5), DA=D(D,5),
    AE=D(A,5), BE=D(B,5), DE=D(D,5),
    BF=D(B,5), CF=D(C,5), DF=D(D,5),
    DG=D(D,5), EG=D(E,5), FG=D(F,5),
    BN=D(B,5), EN=D(D,5), PH=D(P,5),
    INSIDE=AUTOSIM;
VAR U,ERROR;
SVAR ERMAX,TRMAX,UMAX;
CONST TRUE=TRUEF(X,Y);
BCOND U=0 AT AB+BC+CD+DA,
    U=1 AT EH+FH,
    U=1 AT EG+FG;
SCHEME;
SOLVE U OF
    LAPL(U) = 0
    BY 'GAUSS';
ERROR = U;TRUE;
CALL NORMM(TRMAX,TRUE);
CALL NORMM(UMAX,U);
CALL NORMM(ERMAX,ERROR);
PRINT TRMAX,UMAX,ERMAX;
WRITE U to FILE11;
WRITE TRUE to FILE11;
END SCHEME;
END;

double precision function truef(x,y)
double precision x,y
c write(6,*) 'truef',truef
r1 = sqrt((x-1.0)**2+y**2)
r2 = sqrt((x+1.0)**2+y**2)
u = acos((0.5*(r1+r2)))
truef = (2.0-u)/2.0
write(6,*) 'truef',truef
return
end function acos(x)
acos = alog(x+sqrt(x**2-1.0))
return
end

PROG EXA2;
/*
ELLPACK Example 5.A2
*/
/* Special Interior Boundary - Double Valued */
/* Conditions on a Sli */
/*
Coded by Y. Umetani July 2, 1992 */
METHOD FEM;

DOMAIN X=(-5.0.5.0), Y=(-5.0.5.0);

POINT A=(-3.762,0), B=(0,-3.627), C=(3.762,0), D=(0,3.627),
E=(-1.0,0), F=(1.0,0), G=(0,0.01), H=(0,-0.01),
AB1=(-3.716,-0.567), AB2=(-3.758,-1.121), AB3=(-3.352,-1.647),
AB4=(-3.044,-2.132), AB5=(-2.660,-2.565), AB6=(-2.211,-2.934),
AB7=(-1.708,-3.22), AB8=(-1.163,-3.449), AB9=(-0.588,-3.562),
AD1=(-3.716,0.567), AD2=(-3.758,1.121), AD3=(-3.352,1.647),
AD4=(-3.044,2.132), AD5=(-2.660,2.565), AD6=(-2.211,2.934),
AD7=(-1.708,3.22), AD8=(-1.163,3.449), AD9=(-0.588,3.562),
CB1=(-1.0,0), CB2=(1.0,0), CB3=(0,0.01), CB4=(0,-0.01),
CD1=(+3.716,-0.567), CD2=(+3.758,-1.121), CD3=(+3.352,-1.647),
CD4=(+3.044,-2.132), CD5=(+2.660,-2.565), CD6=(+2.211,-2.934),
CD7=(+1.708,-3.22), CD8=(+1.163,-3.449), CD9=(+0.588,-3.562),
CD10=(+3.716,0.567), CD11=(+3.758,1.121), CD12=(+3.352,1.647),
CD13=(+3.044,2.132), CD14=(+2.660,2.565), CD15=(+2.211,2.934),
CD16=(+1.708,3.22), CD17=(+1.163,3.449), CD18=(+0.588,3.562),

REGION AB=SPL(A,AB1,AB2,AB3,AB4,AB5,AB6,AB7,AB8,AB9,B),
BC=SPL(B,CB1,CB2,CB3,CB4,CB5,CB6,CB7,CB8,CB9,C,B),
CD=SPL(C,CD1,CD2,CD3,CD4,CD5,CD6,CD7,CD8,CD9,D),
DA=SPL(D,DA1,DA2,DA3,DA4,DA5,DA6,DA7,DA8,DA9,D),
AB=LN(A,E), BE=LN(B,E), DE=LN(D,E),
BF=LN(B,F), CF=LN(C,F), DF=LN(D,F),
DG=LN(D,G), EG=LN(E,G), FG=LN(F,G),
BH=LN(B,H), EH=LN(E,H), FH=LN(F,H),
AB=TRI(AB,AE,DE), BCF=TRI(BC,BF,CP),
DA=TRI(DA,AE,DE),
BEH=TRI(BE,EE,BH), BFH=TRI(BF,FR,BH),
DGF=TRI(DG,FG,DF),
REGION ABE+BCF+CDG+DAE+BEH+BFH+DGF;
BOUND AB+BC+CD+DA+EH+FH+FG+G;

MESH AB=D(A,E), BC=D(E,B),
CD=D(C,F), DA=D(D,E),
AB=D(A,B), BE=D(B,E), DE=D(D,E),
BF=D(B,F), CF=D(C,F), DF=D(D,F),
DG=D(D,G), EG=D(E,G), FG=D(F,G),
BN=D(E,B), EN=D(D,E), FN=D(D,F),
INSIDE=AUTO;

VAR U;
SVAR UMAX;

BCOND U=0 AT AB+BC+CD+DA,
U=1 AT BH+FH,
U=2.0-X**2 AT EG+FG;

SCHEME;

SOLVE U OF
LAPL(U) = 0
BY 'GAUSS';
CALL NORMM(UMAX,U);
PRINT UMAX;
WRITE U to FILE11;

END SCHEME;
END;
PROG EX5Bl;
	/* ELLPACK Example 5.Bl */
	/* Two-Phase Diffusion Problem */
	/* - Without Jump Condition */
	/* Coded by Y. Umelani July 2, 1992 */
METHOD FEM;
DOMAIN X=[0:1.0], Y=[-0.5:1.0];
EFUNC F;
POINT A=(0,0), B=(1,0), C=(1,1), D=(0,1);
REGION AB=LN(A,B), BC=LN(B,C),
CD=LN(C,D), DA=LN(D,A),
ABCD=QUAD(AB,BC,CD,DA);
REGION ABCD;
BOUND AB+BC+CD+DA;
MESH AB=DN(A,8), BC=DN(B,12),
CD=DN(C,8), DA=DN(D,12),
INSIDE=AUTOSIM;
VAR U;
BCOND NGRAD(U)=0.0 AT DA+BC,
U=0.0 AT AB,
U=1.0 AT CD;
SCHEME;
SOLVE U OF
LAPL(U) = F(X,Y)
BY 'GAUSS';
PRINT U;
WRITE U to FILE11;
END SCHEME;
END;

double precision function f(x,y)
double precision x,y
if (y .le. 0.0) then
  f = 4.*y*(1.-y)*sin(1.57080*(3.*x+0.5))
else
  f = 0.0
endif
return
end
PROG  EX5B2;
/*  ELLPACK Example 5.B2  */
/*  Two-Phase Diffusion Problem  */
/*  - With Jump Condition  */
/*  Coded by Y.Umezumi July 7,1992  */
METHOD  FEM;
DOMAIN  X=[0.1,1.0], Y=[0.5,1.0];
EFUNC  FN;
POINT  A=(0,0.5), B=(1.0,0.5), C=(1.0,1.0),
      D=(0.1), E=(0.0), F=(1.0,0.0);
REGION  AB=LN(A,B), BF=LN(B,F), FC=LN(F,C),
      CD=LN(C,D), DE=LN(D,E), EA=LN(E,A), EF=LN(E,F),
      ABFE=QUAD(AB,BF,EF,EA), CDEF=QUAD(CD,DE,EF,FC);
REGION  ABFE+CDEF;
BOUND  AB+BF+CD+DE+EA;
MESH  AB=DN(A,B), BF=DN(B,F), FC=DN(F,C),
      CD=DN(C,D), DE=DN(D,E), EA=DN(E,A),
      EF=DN(E,F),
      INSIDE=AUTOSIM;
VAR  U, UOLD, DYA, DYB, UDIFF;
SVAR  UDIF, EPSO, K;
BOUND  BGRP01 U=0.0 AT AB,
       NGRAD(U)=0.0 AT BF+EA,
       BGRP02 U=1.0 AT CD,
       NGRAD(U)=0.0 AT DE+FC,
       BGRP03 U=1.0 AT CD,
       U=0.0 AT AB,
       NGRAD(U)=0.0 AT DE+FC+BF+EA;
COUNT  CTR;
SCHEME;
  SOLVE U OF LAPL(U)=FN(X,Y) BY 'GAUSS' AT ABFE+CDEF UNDER BGRP03;
  EPSO=1.0D-5; K=0.5;
  ITER CTR UNTIL UDIF LT EPSO;
  UOLD=U;
  DY=DN(Y) AT ABFE;
  SOLVE U OF
  LAPL(U) = FN(X,Y)
  BY 'GAUSS' AT CDEF UNDER BGRP02;
  DY=DN(Y) AT CDEF;
  SOLVE U OF
  LAPL(U) = 0
  BY 'GAUSS' AT ABFE UNDER BGRP01;
  UDIF=U-UOLD;
  CALL NORMM(UDIF,UDIFF);
  PRINT UDIF;
END ITER;
PRINT DY, DY AT EF;
WRITE U to FILE11;
END SCHEME;
END;

double precision function fn(x,y)
double precision x,y
if (y.ge.0.0) then
  fn=4.0*x*(1.0-y)*sin(1.5708*(3.0*y+0.5))
else
  fn=0.0
endif
return
end
PROG UNEXCL;

METHOD FEM;

PROGRAM ELLPACK Example 5.cl

/*
/* Apply Newton's Method to the Nonlinear Problem */
/*
/* \n/* Uxx + U**2 + Uyy = EXP(U) + F(X,Y) */
/*
/* Coded by Y. Umelani July 8, 1992 */
/*
/* Revised by Y. Umelani Oct. 2, 1992 */
*/

METHOD FEM;

DOMAIN X=[0,1.0], Y=[0,1.0];

POINT A=(0,0), B=(1,0), C=(1,1), D=(0,1);

METHOD FEM;

DOMAIN X=[0,1.0], Y=[0,1.0];

POINT A=(0,0), B=(1,0), C=(1,1), D=(0,1);

EFUNC F;

REGION AB=LN(A,B), BC=LN(B,C),
CD=LN(C,D), DA=LN(D,A),
ABCD=QUAD(AB,BC,CD,DA);

PRIVED ABCD;

BOUND AB=DC+CD+DA;

MESH AB=DA, BC=DA, CD=DA, DA=DA,

INSIDE=AUTOSIM;

VAR U, UO, ERROR, COEFF, USQY,
UOX, UOY;

SVAR ERMAX, ITRN;

TENS CT=(ONE.ZERO.ZERO.ZERO.USQ);

CONST TRUE=SIN(X)*COS(Y);

BCOND U=TRUE AT AB+BC+CD+DA,
UO=U,

USQY=SIN(X)*SIN(Y),
USXY=SIN(X)*SIN(Y);

INTEGER ITRN;

COUNT NITER;

ELMTYPE QUADRIC for ALLNAMES;

SCHEME;

ITRN=FILE51;

ITER NITER UNTIL NITER GE ITRN;

UO=DY(UO), UOY=DY(UOY);

USQY=UO**2; USQY=UO**2*USQY;

COEFF=2.*UO*USQY*EXP(UO);

SOLVE U OF

DIV(CT,GRAD(U)).USQY*DY(U)+COEFF*U =
2*UO**2*USQY*EXP(UO)+(UO-1.0)+F(X,Y)

BY 'GAUSS';

ERROR = TRUE-U ;

CALL NORMM(ERMAX,ERROR);

PRINT NITER,

PRINT ERMAX,

WRITE U to FILE52;

WRITE TRUE to FILE52;

END ITER;

END SCHEME;

END;

doouble precision function f(x,y)
doouble precision x,y
trueexp(x)*trueexp(y)
trueexp=true
trueexp=true
f = trueexp**trueexp
+ (2.*trueexp-trueexp)*trueexp
+ (-2.*trueexp+trueexp+exp(trueexp=1.)
return
end

function f(x,y)
doouble precision x,y
trueexp(x)*trueexp(y)
trueexp=true
trueexp=true
f = trueexp**trueexp
+ (2.*trueexp-trueexp)*trueexp
+ (-2.*trueexp+trueexp+exp(trueexp=1.)
return
end
PROG EX5C3;
/* ELLPACK Example 5.C3 */
/* Apply Newton's Method to the Nonlinear Problem */
/* W(U)UXX + WY(U)UY + W(U)UX + WY(U)UY = G */
/* W(U) = SQRT (UX**2 + UY**2) */
/* Coded by Y. Umemi July 9, 1992 */

METHOD FEM;
DOMAIN X=[0:1.0], Y=[0:1.0];
POINT A=(0,0), B=(1.0,0), C=(1.0,1.0),
D=(0,1);
REGION AB=LN(A,B), BC=LN(B,C),
CD=LN(C,D), DA=LN(D,A),
ABCD=QUAD{AB,BC,CD,DA};

REGION ABCD;
BOUND ABfBCfCDfDA;
MESH AB=D{A,t), BC=D{B,t),
CD=D{C,t), DA=D{D,t),
INSIDE=AUTOSIM;

VAR U, UO, UX, UY, UX0, UY0, UXX, UYY, ERROR,
FN, RS;
SVAR ERMAX, ITMAX;
INTEGER ITMAX;
VEC CV={DN, EN};
TENS CT={AN, BN, ZERON, CN),
DNT={DTXX, DTXY, ZERO, DTYY),
ENT={ETXX, ETXY, ZERO, ETYY);
CONST PI=3.1415926,
TRUE={1.0+EXP(-Y)*COS(PITX),
TRUEX=PPI*SIN(PITX)*EXP(-Y)+1),
TRUEY=2.0*UOX*UOY;
TRUXX=PPI*COS(PITX)*EXP(-Y)+1),
TRUXY=PPI*SIN(PITX)*EXP(-Y),
TRUY=2.0*UOX*UOY;

CTENS TXX={I, O, O, O), TXY={O, I, O, O),
TYX={O, O, I, O), TYY={O, O, O, I);

UTCOUNT NITER;
/* ELEMENT QUADRIC for ALLNAMES; */
SCHEME;
SOLVE UO OF LAPL{UO)=O BY 'GAUSS';
ZERO=O.O;
ITMAX=FILEOl;
PRINT UO;
ITER NITER UNTIL NITER GE ITMAX;
UOX=DX(UO); UOY=DY(UO);
UXX=DIV(TXX,GRAD(UO));
UXX=DIV(TX0,GRAD(UO));
UYY=DIV(TY0,GRAD(UO));

AN=UOY**2+2.0*UOY**2;
BN=3.0*UOX*UOY;
CN=2.0*UOX**2+4.0*UOY**2;
DTXX=2.0*UOX**2+3.0*UOX*UOY;
DTXY=2.0*UOY**2;

DN={DTXX*UXX+DTXY*UXY+DTYY*UY0)/(UOX**2+UOY**2),
ETXX=UOY**2;
ETXY=2.0*UOY**2;
ETYY=2.0*UOY**2;
EH={ETXX*UXX+ETXY*UXY+ETYY*UY0)/(UOX**2+UOY**2);
FN=0.0;
RS=AN*TRUEX+BN*TRUEX+CN*TRUEX+DN*TRUEY+EN*TRUEY+FN*TRUE;
DN=0.0;
EN=EN+UOY*UXX+UOX*UYY;
ERROR = TRUE - U;
COUNT NITER;
WRITE U to FILEl1;
WRITE TRUE to FILEl2;
WRITE ERROR to FILEl3;
UO = U;
END ITER;
END SCHEME;
END;
/* ELLPACK Example 5.Ct */
/* Apply Newton's Method to the Nonlinear Problem */
/* Reynolds' Equation for Compressible Fluid Lubrication */
/* Coded by Y. Umetani July 9, 1992 */

METHOD FEM;

DOMAIN X=[0:1.5], Y=[0:1.0],
POINT A=(0,0), B=(1.0,0),
C=(1.0,1.0),
D=(0,1), E=(1.5,0.5),
F=(1.353,0.853),
H=(1.0,0.5), I=(0,0.5);
REGION AB=LN(A,B), BH=LN(B,H), HC=LN(H,C),
CD=LN(C,D), D1=LN(D,I), IA=LN(I,A),
EH=LN(E,H),
HI=LN(H,I), BE=ARC(B,F,E), CE=ARC(C,G,E),
ABE=QUAD(AB,BH,HI,IA),DCH=QUAD(D1,HI,HC,CD),
BEH=TRI(BE,EH,BH), CEH=TRI(CE,EB,HC);

FREGION ABHI+DCHI+BEH+CEH;
BOUND AB+BE+CE+CD+DI+IA;
MESH AB=D(A,8), BH=D(B,t), HC=D(H,t),
CD=D(C,8), DI=D(D,t), IA=D(I,t),
EH=D(E,t),
HI=D(H,8), BE=D(B,t), CE=D(C,t),
INSIDE=AUTOSIM;

VAR U,UX,UY,UXX,UYY,DIFFS,
AN,DN,EN,FN,RS;
SVAR DFMAX,ITMAX,
CC;
INTEGER ITMAX;
CONST HH=1.+2.*X, Hx=2.0, Hy=0.0;
BCOND U=1.0,
UO:=U;
COUNT NITER;

SCHEME;
UO .. UO + ITMAX=5;
ITMAX=5;
CC=12.0;
ITER NITER UNTIL NITER GE ITMAX;
UXX=DX(U); UYY=DY(U); UOOY=DX(UO);
UOOY=DU(UY);
AN=2.*HH**3*UXX+3.*HH**2*Hx*U+CC*HH;
EN=2.*HH**3*UYY+3.*HH**2*Hy*U+CC*HH;
FN=HH**3*(UXX+UYY)+3.*HH**2*(Hx*UX+Hy*UY)+CC*HH;
RS=HH**3*(UO+UYY)+HH**3*(UO+UXX)+HH**3*(HH**2*Hx*UO+Hx*UO*UX);

SOLVE U OF
AN*LAPL(U) + DN*DX(U) + EN*DY(U) + FN*U = RS
BY 'GAUSS'.
DIFFF = U-UO;
WRITE DIFFS to FILE13;
WRITE U to FILE11;
WRITE DIFFS to FILE13;
UO = U;
END ITER;
END SCHEME;
END;
PROG EX5D1;
/* ELLPACK Example 5.1. Time Dependent Program */
/* by Crank-Nicolson Scheme. Start time is changed */
/* from 1 to 0 due to the DEQSOL restriction. */
/* U_t=4*y**2*U_{xx}+U_{yy}+(2+tan(x+y)+t^4)U_y+U+f(x,y,t) */
/* Coded by Y. Umetani July 10, 1992 */

METHOD FEM;
DOMAIN X=[0:1.0], Y=[0:1.0];
TIME T=[0:2];

EFUNC F;
POINT A=(0,0), B=(1.0,0), C=(1.0,1.0),
      D=(0,1);
REGION AB=LN(A,B), BC=LN(B,C),
       CD=LN(C,D), DA=LN(D,A),
       ABCD=QUAD(AB,BC,CD,DA);
FRREGION ABCD;
BOUND AB+BC+CD+DA,
MESH AB=D(A,2), BC=D(B,2),
     CD=D(C,2), DA=D(D,2),
INSIDE=AUTOSIM;
TSTEP DLT=[0(0.125)];
VAR U,UO,ERROR,
    DN,DNO;
SVAR ERMAX,ITMAX,TRMAX,REMX;
INTEGER ITMAX;
CONST TRUE=SIN((X+Y+T)/4.)*EXP(-Y**2-T),
       CXX=4.*Y**2,
       CTENS CT=(CXX,0,0,1);
BCOND U=TRUE,
     UO:=U;
ICOND UO=TRUE, U=TRUE;
/* ELMTYPE QUADRIC for ALLNAMES; */
SCHEME;
   ITMAX=2.0/0.125;
   ITER NT UNTIL NT GE ITMAX;
   DN=2.4*TAN((X+Y+T)/4.);
   DNO=2.4*TAN((X+Y+T/DTL)/4.);
   SOLVE U OF
   2.0/DTL^2*U = 2.0/DTL^2*U0 + DIV(CT..GRAD(U)) + DN*DY(U) + U + F(X,Y,T) +
   DIV(CT..GRAD(U0)) + DNO*DY(U0) + U0 + F(X,Y,T/DTL)
   BY 'GAUSS' ;
   ERROR = TRUE-U;
   CALL NORMM(ERMAX,ERROR); CALL NORMM(TRAEX,TRUE);
   REMX = ERMX/TRAEX;
/* PRINT NT,TRMAX,ERMAX,REMX; */
   WRITE U to FILE11;
   WRITE TRUE to FILE12;
   WRITE ERROR to FILE13; /*
   U0 = U;
END ITER;
END SCHEME;
END;

double precision function f(x,y,t)
double precision x,y,t
  t1 = 2*SIN(x+y+1)
  t2 = exp(y**2+1)
  f = - (25*cos(11) - 2.75*y*sin(11) - t2 * (tan(11) + 2.))
  a = + (0.625 - 3.75*y**2) * t2 * sin(11)
  b = + (25 + y) * t2 * cos(11)
return
PROG EX5El;
/* ELLPACK Example 5.E1 */
/* Two Equations · Simple Interaction Between Them */
/* Vxx + (1+W)Vyy + Y(1-V) = G1(X,Y) */
/* (1+V)Wxx + Wyy + Y(1+V)W = G2(X,Y) */
/* V, W given on Y=0, Vx, Wx given on X=0 */
/* Known solution is V = (X+Y)**4 W = EXP(X-Y) */
/* Quadric Element. Coded by Y. Umelani July 13, 1992 */
METHOD FEM;
DOMAIN X=[0:1], Y=[0:1],
EFUNC G1, G2;
POINT A=(0,0), B=(1,0), C=(1,1),
D=(0,1);
REGION AB=LN(A,B), BC=LN(B,C),
CD=LN(C,D), DA=LN(D,A),
ABCD=QUAD(AB,BC,CD,DA);
FREGION ABCD;
BOUND AB+BC+CD+DA;
MESH AB=D(A,'), BC=D(B,'),
CD=D(C,'), DA=D(D,');
INSIDE=AUTOSIM;
VAR V, W, D, E, F, R, ERR;
SVAR NITRS, ERMX, ERMX;
TENS CT=(ONE, ZERO, ZERO, CV),
CTW=(AW, ZERO, ZERO, ONE);
CONST TRUEV=(X+Y)**4, TRUEW=EXP(X-Y),
TRUEW=EXP(X-Y);
INTEGER NITRS;
BCOND V=TRUEV at AB+CD,
W=TRUEW at AB+CD,
N(CTV.GRAD(V))=TRUEV at BC,
N(CTV.GRAD(V))=TRUEV at DA,
N(CTW.GRAD(W))=(1+TRUEV)*TRUEW at BC,
N(CTW.GRAD(W))=(1+TRUEV)*TRUEW at DA;
COUNT NITER;
/* ELM TYPE = QUADRIC for ALLNAMES; */
SCHEME;
ONE=1.0; ZERO=0.0;
NITRS=7;
V=0.0; W=0.0;
ITER NITER UNTIL NITER GE NITRS;
CV=1.0; EV=EY(W); FW=YN(1.0-W);
SOLVE V OF
DIV(CTV.GRAD(V)) - EV*DY(V) + FW*V = G1(X,Y)
BY 'GAUSS';
ERRR=TRUEV-V; CALL NORMM(ERMX, ERMX);
PRINT ERMX;
WRITE V to FILE11;
WRITE TRUEV to FILE12;
WRITE ERMX to FILE13;
WRITE W to FILE14;
WRITE TRUEW to FILE15;
WRITE ERMX to FILE16;
END ITER;
END SCHEME;
END;

double precision function g1(x,y)
double precision x,y
v = (x+y)**4
w = exp(x-y)
vxx = 12.0*(x+y)**2
g1 = (2.0+w)*vxx + (1.0-w)*y*v
return end
double precision function g2(x,y)
double precision x,y
v = (x+y)**4
w = exp(x+y)
g2 = (1.0 + (1.0+y)*(1.0+y))*w
return end
**PROG EX5E2;**

/* ELLPACK Example 5.E2 */

/* Two Equations - Simple Interaction Between Them */

/* V(x) + (1+W) V(y) + Y(1-W) V = G1(X,Y) */

/* (1+V) W(x) + W(y) + Y(1+V) W = G2(X,Y) */

/* Known solution is V = (X+Y)**4 W = EXP(X-Y) */

/* Linear element. Coded by Y. Umelani July 13, 1992 */

/* Revised by Y. Umelani August 6, 1992 */

**METHOD FEM,**

**DOMAIN** X=[0:1], Y=[0:1];

**EFUNC** G1,G2;

**POINT** A=(0,0), B=(1,0), C=(1,1), D=(0,1);

**REGION** AB=LN(A,B), BC=LN(B,C), CD=LN(C,D), DA=LN(D,A), ABCD=QUAD(AB,BC,CD,DA);

**BOUND** ABCD:

**MESH** AB=D(A,16), BC=D(B,16), CD=D(C,16), DA=D(D,16), INSIDE=AUTOSIM;

**VAR** V,W,DV,DW,EV,FW,ERRV,ERRW;

**SVAR** NITRS,ERMXV,ERMXW;

**TENS** CTV=(ONE,ZERO,ZERO,CTV),

CTW=(AW,ZERO,ZERO,ONE);

**CONST** TRUEV=(X+Y)**4, TRUEW=EXP(X-Y), TRUVX=-(X+Y)**4, TRUWX=EXP(X-Y);

**INTEGER** NITRS;

**BCOND** V=TRUEV at AB+CD,

W=TRUEW at AB+CD,

N(CTV .. GRAD(V»= TRUVX at BC,

N(CTV .. GRAD(V»= TRUVX at DA,

N(CTW .. GRAD(W»= (1+TRUEV) TRUWX at BC,

N(CTW .. GRAD(W»= (1+TRUEV) TRUWX at DA;

**COUNT** NITER;

**SCHEME**;

ONE=1.0; ZERO=0.0;

NITRS=7;

V=0; W=0;

ITER NITER UNTIL NITER GE NITRS;

CV=1+V; DW=DY(W); FW=Y*(1+V);

SOLVE V OF

DIV(CTV .. GRAD(V) .. EV*DY(V) + FW*V = G1(X,Y)

BY 'GAUSS';

ERRV=TRUEV-V; CALL NORMM(ERMXV,ERRV);

PRINT EV;

WRITE V to FILE11;

WRITE TRUEV to FILE12;

WRITE ERRV to FILE13;

AW=1+V; DW=DX(V); FW=Y*(1+V);

SOLVE W OF

DIV(CTW .. GRAD(W) .. DW*DX(W) + FW*W = G2(X,Y)

BY 'GAUSS';

ERRW=TRUEW-W; CALL NORMM(ERMXW,ERRW);

PRINT ERMXW;

WRITE W to FILE14;

WRITE TRUEW to FILE15;

WRITE ERRW to FILE16;

END ITER;

END SCHEME;

**END**;

```plaintext
double precision function g1(x,y)
double precision x,y
v = (x+y)**4
w = exp(x+y)
vxx = 12.0*(x+y)**2
g1 = (2.0+w)*vxx + (1.0-w)*y*v
returns
double precision function g2(x,y)
double precision x,y
v = (x+y)**4
w = exp(x-y)
g2 = (1.0 + (1.0+y)**(1.0+y))**w
returns
double precision function g1(x,y)
double precision x,y
v = (x+y)**4
w = exp(x+y)
vxx = 12.0*(x+y)**2
g1 = (2.0+w)*vxx + (1.0-w)*y*v
returns
double precision function g2(x,y)
double precision x,y
v = (x+y)**4
w = exp(x-y)
g2 = (1.0 + (1.0+y)**(1.0+y))**w
returns
```
/* ELLPACK Example 5.E6 */

/* Two Equations - Tight Interaction Between Them */

/* Uxx + (1+sin2Y)/2)Uyy + (Y+X)Ux + exp(VXY)U = g1(X,Y) */
/* Vxx + (1+sin(2X)U)Uyy + (4U+X)Vx + sin(U)**2VV = g2(X,Y) */

/* V, W given on Y=0, X=0,1 */

/* Known solution is V = (X+Y)**4, W = EXP(X-Y) */

/* Simultaneous Solve. Coded by Y. Umemi July 13, 1992 */

METHOD PFM;

DOMAIN X=[0:1], Y=[0:1];
EFUNC G1,G2;
POINT A=(0,0), B=(1,0), C=(1,1), D=(0,1);
REGION AB=LN(A,B), BC=LN(B,C),
CD=LN(C,D), DA=LN(D,A),
ABCD=QUAD(AB,BC,CD,DA);
REGION ABCD;
BOUND AB+BC+CD+DA,
MESH AB=D(A,3), BC=D(B,3),
CD=D(C,3), DA=D(D,3),
INSIDE=AUTOSIM;
VAR U, V, UO, VO, EU, EV, ERRU, ERRV;
SVAR NITRS, ERMXU, ERMXV;
TENS CTU=(ONE,ZERO,ZERO,CU),
CTV=(ONE,ZERO,ZERO,CV);
CONST TRUEU=(X+Y)**4, TRUEV=EXP(X,Y);
BCOND U=TRUEU, UO:=U,
V=TRUEV, VO:=V;
INTEGER NITRS;
COUNT NITER;
ELMTYPE QUADRIC for ALLNAMES;
SCHEME;
NITRS=7;
ONE=1.0; ZERO=0.0;
UO=0.0; VO=0.0;
ITER NITER UNTIL NITER GE NITRS;
CU=1.0+SIGN(2.*X*UO);
EV=SIGN(2.*UO+X*COS(2.*X*UO))*SIN(2.*X*UO)*SIN(2.*UO)*DY(UO);
SOLVE U.V OF
D1V(CTU..GRAD(U)+(U+Y)*DX(U)-EU*DY(U)+EXP(U)*X*Y*U = G1(X,Y),
D1V(CTV..GRAD(V)+(U+X)*DX(V)-EV*DY(V)+SIN(U)**2*Y*V = G2(X,Y)
BY 'GAUSS';
ERRRU=TRUEU-U, CALL NORMM(ERMXU,ERRRU);
ERRRV=TRUEV-V, CALL NORMM(ERMXV,ERRRV);
PRINT NITER,ERMXU,ERMXV;
WRITE U TO FILE11;
WRITE TRUEU to FILE12;
WRITE ERRU to FILE13;
WRITE V TO FILE14;
WRITE TRUEV to FILE15;
WRITE ERRV to FILE16;
UO=U; VO=V;
END ITER;
END SCHEME;
END;

double precision function g1(x,y)
double precision x,y
u = (x+y)**4
v = exp(x+y)
ux = 4*(x+y)**3
uxx = 12*(x+y)**2
uy = 12.0*(x+y)**2
uyy = 12.0*(x+y)**2
g1 = ux*(1-sin(2.*x))/2.*uyy+(v+4.*y)*ux*exp(v*x*y)*u
return end
double precision function g2(x,y)
double precision x,y
u = (x+y)**4
v = exp(x+y)
vx = exp(x+y)
vxx = exp(x+y)
vy = exp(x+y)
g2 = vx*(1.0+u*sin(2.*x*u))*vyy+(4.*u+x)*vx+sin(u)**2*y*vy
return end