DEQSOL and ELLPACK : Problem Solving Environments For Partial Differential Equations*

Ronald F. Boisvert
Center for Computing and Applied Mathematics
National Institute of Standards and Technology
Technology Building, Room A151
Gaithersburg, MD 20899 USA
Phone: (301) 975-3812
Email: BOISVERT@CAM.NIST.GOV

David K. Kahaner
Office of Naval Research, Far East
23-17, 7-chome, Roppingi
Minato-ku, Tokyo 106 JAPAN
Phone: (03) 401-8978
Email: KAHANER@XROADS.CC.U-TOKYO.AC.JP

July 23, 1990

Abstract

A number of integrated problem-solving environments for the solution of partial differential equations have been developed in recent years. Such systems permit very-high-level descriptions of the mathematical problem and its solution algorithm. These descriptions are translated into efficient Fortran programs which produce solutions in both tabular and graphical forms. In this paper we compare and contrast two such systems: DEQSOL, a Japanese project, and ELLPACK, an American project. For each system, we present a brief history, describe its capabilities, and present several examples of its use.

*This paper was published in the *Office of Naval Research Far East Information Bulletin*, vol. 16, no. 1, 1991, pp. 7–19.

1. Introduction

The numerical solution of partial differential equations (PDEs) was one of the earliest applications of electronic digital computers, and it remains the source of many challenging computational problems today. Such problems can be found in every scientific discipline. They range from computing the currents and fields in VLSI devices to determining the flow about an airfoil.

Due to the importance of mathematical models based on PDEs, and their general resistance to analytical treatment, the numerical solution of PDEs has been the focus of a great deal of research over the years. In spite of great algorithmic advances which have been achieved, very few general-purpose software packages for solving PDEs have appeared, and those that have tend to concentrate on very narrow problem areas. General-purpose PDE software is just extraordinarily difficult to build. This is due to difficulties in designing both solution algorithms and user interfaces.

Since PDE problems exhibit such a wide range of behaviors, no software package built upon a single numerical method can succeed. As a result, the internal structure of such a solver must be very sophisticated. PDE problems cannot be expressed naturally in an algebraic language such as Fortran, which remains the computer language of choice for scientific computation. Because of this, the user interface must either be very complex, or a new high-level language must be designed and implemented for describing PDE problems and their solution. A traditional Fortran subprogram cannot do this job; instead, an integrated problem-solving environment must be designed. Such systems must be designed by computer scientists in close collaboration with the numerical analysts who develop efficient methods and physical scientists and engineers who need to use them.

The potential payoffs for the development of such systems is great. Solving PDE problems by writing Fortran code is a very time-consuming and error prone task which often duplicates work of many previous programmers. Scientists and engineers may not be aware of the most appropriate algorithms to use, and hence the programs produced by these efforts may well be very slow or even produce erroneous results. Such difficulties are increased in the face of vector and parallel processing computers which are very difficult to make efficient use of by the casual user. Using software parts from existing program libraries can alleviate some of these problems, but this still requires much low-level programming effort, even to solve easy problems.

A number of researchers have made efforts to produce integrated problem-solving environments for the solution of PDEs (see the short bibliography at the end of this article). Two major systems which remain active foci of research are DEQSOL, a Japanese project, and ELLPACK, an American project. These systems have many overt similarities. Both are preprocessor-based; they read very-high-level descriptions of PDE problems and solution algorithms and produce a Fortran 77 program as output. When linked with their respective run-time libraries and executed, the generated programs will solve the given PDE problem.

Both support graphical output. Linear second-order steady-state PDE problems in fairly general domains in two or three spatial dimensions are the most straightforward to solve using each system. Nonlinear problems, time-dependent problems, and systems of equations, are also solvable, but doing so requires more of the user. There are, of course, many differences between DEQSOL and ELLPACK. In what follows we will try to give you a feeling for these two projects in more detail.

To illustrate the use of the two packages, we will present programs designed to solve two simple problems. (Note that ∇^2 represents the Laplacian in cartesian coordinates, i.e., $\nabla^2 = \partial^2/\partial x^2 + \partial^2/\partial y^2$.)

Problem 1 : Stress distribution in a thin plate containing a hole

The domain for this problem is the square $(-2,2) \times (-2,2)$ with a circle of radius 1 remove from its center. The stress function satisfies

$$\nabla^2 \phi = -2$$

on the plate, with $\phi = 1$ at the outside edge and $\phi = 0$ at the edge of the hole. Due to symmetry we need only consider the region $x > 0$, $y > 0$, with

$$\partial\phi/\partial n = 0, \quad \text{for } x = 0 \text{ and } y = 0.$$

Problem 2 : Thermal diffusion problem

Find the function $T(t, x, y)$ which satisfies

$$\partial T/\partial t = a\nabla^2 T$$

on the rectangle $(0,1) \times (0,1.2)$ for $0 < t < 1$ with initial condition

$$T(0,x,y) = 100\exp(-[(x-0.5)^2 + y^2])$$

and boundary conditions

$$\begin{aligned} \partial T/\partial n &= 0 && \text{for } x = 0 \text{ and } x = 1 \\ T &= 200 && \text{for } y = 0 \\ \partial T/\partial n - \lambda T &= \mu && \text{for } y = 1.2 \end{aligned}$$

for the constants $a = 0.62$, $\lambda = 0.8$, and $\mu = 50$.

2. DEQSOL

The DEQSOL project, headed by Dr. Yukio Umetani, is based at the Central Research Laboratory of Hitachi Ltd. The laboratory employs about 1300 people in a heavily wooded park-like setting about 30 minutes by local train from central Tokyo. Contributions to the system have also been made by staff at Hitachi VLSI Engineering Ltd. From the outset, the goal of the DEQSOL project was to develop an easy-to-use system in which practical PDE models could be expressed and solved while taking the best advantage possible of the supercomputer performance possible using Hitachi's vector processors.

The project began in 1980, with language design. Umetani set to work on DEQSOL after many years developing vectorizing compilers for Hitachi. Thus he had a language designer's understanding of how user Fortran needed to be structured in order for it to compile into efficient vectorizable code. He also felt that Fortran was too low level a language to allow scientists to express concepts that would vectorize and parallelize easily, and that a higher level language would not only make it easier for scientists to develop algorithms but also make it easier for compilers to generate efficient code. The earliest version (1982) only dealt with two-dimensional diffusion problems on rectangular grids, but by 1983 three-dimensional problems and more general PDEs could be solved. In 1986 a "Basic System" was officially available as a product from Hitachi. This included the ability to solve linear partial differential equations using either finite difference or finite element methods. A separate "Enhanced System" with many additional features was released in 1987. Solution functions to support the solution of fluid flow problems were added to the Enhanced System in 1989 and 1990.

The DEQSOL project currently has more than 20 people involved in research, development, and commercialization. DEQSOL runs on Hitachi's M680 and S810 supercomputers, for which approximately 70 copies of the package have been sold, all in Japan. Umetani claims that it would be very easy to modify the DEQSOL source to generate efficient Fortran for other vector computers, but that it would require more work to take advantage of multiprocessors.

DEQSOL's Basic System is designed to automatically solve linear elliptic problems on fairly general two- and three-dimensional domains with general linear boundary conditions. The system easily handles problems with material interfaces. Nonlinear problems and time-dependent problems may also be solved, but the user must program an iterative scheme to solve the problem. This is quite natural to do in DEQSOL.

The language syntax allows differential equations to be expressed in a natural mathematical syntax. Scalar, vector, and tensor variables are supported, and differential operators such as first partial derivatives (`DX`, `DY`, `DZ`), the normal derivative (`NGRAD`), the Laplacian (`LAPL`), the divergence (`DIV`), and the gradient (`GRAD`) are defined on them. A variety of declaration statements allow users to specify the problem geometry (e.g., `DOMAIN`, `POINT`, `REGION`, `FREGION`), the finite difference or finite element mesh (`MESH`), problem variables and constants (e.g., `VAR`, `VEC`, `CONST`), and initial conditions (`ICOND`), and boundary conditions (`BCOND`). The solution algorithm is specified in the `SCHEME` block, which comprises the executable portion of a DEQSOL program.

Both finite difference (FDM) and finite element (FEM) discretizations are supported.

For finite differences, boundaries must be made up of straight lines (or planes in 3D) parallel to the coordinate axes. Second-order accurate differences are supported for general linear differential operators. For finite elements, boundaries in two dimensions may be made up of straight lines, circular arcs, or cubic splines. In three dimensions boundaries can be built using cubic or tetrahedral parts. The problem is discretized using the Galerkin method on linear triangular elements.

Graphical output is available via the SGRAF postprocessor. SGRAF allows users to obtain output in various forms including contour plots, surface plots, vector plots, and intensity plots. Plots of domains with superimposed grids may also be obtained; color output is supported.

DEQSOL's Enhanced System provides a number of additional capabilities in geometry processing, types of equations handled, and numerical techniques supported. Among these improvements are:

- A boundary-fitted coordinate transformation method (BFM) permits general domains to be automatically mapped to rectangular ones where FDM techniques may be applied.
- An interface to a CAD system allows easier description of complex domains for the FEM.
- Coupled systems of elliptic equations may be expressed and solved automatically.
- Language enhancements allow block iteration schemes to be easily expressed.
- Staggered grids and up-wind differencing may be selected for the FDM.
- Higher-order basis functions and up-wind techniques may be used in the FEM.

The DEQSOL preprocessor is written in Pascal, and generates Fortran 77 code. The form of the user interface to DEQSOL is illustrated by the following DEQSOL solutions to the two sample problems.

Problem 1 : Stress distribution in a thin plate containing a hole

The problem is solved by Galerkin finite element method on linear triangular elements. A nonuniform triangular mesh is defined. The linear equations are solved using Gaussian elimination for banded matrices.

```
PROG POIS;
METHOD    FEM;
DOMAIN    X=[0:2], Y=[0:2];
POINT     A=(0,2), B=(2,2), C=(2,0), D=(1,0),
          E=(0.7071,0.7071), F=(0,1),
          I1=(0.5,0.8660), I2=(0.8660,0.5);
REGION    AB=LN(A,B), BC=LN(B,C), CD=LN(C,D), AF=LN(A,F),
          DE=ARC(D,I2,E), EF=ARC(E,I1,F), BE=LN(B,E),
```

```
          ABEF=QUAD(AB,BE,EF,AF), CBED=(CB,BE,ED,CD);
FREGION   ABEF+CBED;
BOUND     AB+BC+CD+DE+EF+AF;
MESH      AB=D(A,8), BE=D(B,7), EF=D(E,5), AF=D(A,5),
          BC=D(B,8), CD=D(C,5), DE=D(D,5),
          ABEF=AS(A,B), CBED=AS(C,B);
VAR       PHI;
BCOND     PHI=0        AT AB+BC,
          PHI=1        AT DE+EF;
          NGRAD(PHI)=0 AT CD+AF;
SCHEME;
   SOLVE PHI OF   LAPL(PHI) + 2 = 0   BY 'GAUSS';
   PRINT X; PRINT Y; PRINT PHI; SAVE PHI;
END SCHEME;
END;
```

Problem 2 : Thermal diffusion problem

The problem is solved using a finite difference spatial discretization and a backward Euler implicit time step. At each time step the implicit equations are solved by a conjugate gradient iteration with preconditioning by an incomplete LU factorization.

```
PROG    THERM;
METHOD  FDM;
DOMAIN  X=(0:1), Y=(0:1.2);
TDOM    T=(0:1);
MESH    X=(0:1:5), Y=(0:1.2:6), T=(0:1:0.02);
VAR     TEMP, TOLD;
REGION  LEFT=(0,*), RIGHT=(1,*), BOTTOM=(*,0), TOP=(*,1.2),
        TOTAL=(*,*);
ICOND   TEMP=100*EXP(-(X-0.5)**2-Y**2) AT TOTAL;
BCOND   DX(TEMP)=0 AT RIGHT+LEFT,
        TEMP=200 AT BOTTOM,
        DY(TEMP)=LM*TEMP+MU AT TOP;
SCHEME;
        ITER NT UNTIL NT GT 50;
           TOLD = TEMP;
           SOLVE TEMP OF (TEMP-TOLD)/DLT = A*LAPL(TEMP)
              BY 'ILUBCG';
           PRINT TEMP EVERY 10 TIMES;
        END ITER;
END SCHEME;
END;
```

The Fortran code to discretize the problem is generated by the DEQSOL preprocessor. This code is expressed in such a way as to be easily vectorized by Hitachi vector processors. Solution of linear equations in implicit equations (the `SOLVE` statement) is done by calling library routines; several methods are available, including banded Gauss elimination and preconditioned conjugate gradient iteration. The library routines are also highly vectorized. DEQSOL does not have any explicit method to allow the user to indicate parallelism in the problem description, but Umetani and other Hitachi colleagues are also working on a new programming language PARAGRAM that does. At the moment DEQSOL and PARAGRAM are independent projects.

Considerable effort has been expended to apply DEQSOL to realistic mathematical models in a variety of application areas. Some of these include (the number of spatial dimensions and the solution method is given in parenthesis):

- impurity distribution analysis in LSI process CAD (2D-FDM)
- device simulation of WEL layer in LSI (2D-FDM)
- air cooling analysis for a baseball dome (2D-BFM)
- electric field analysis of sheet resistor (2D-BFM)
- analysis of flow past an automobile (2D-FEM)
- magnetic field analysis of Josephson Junction device (2D-FEM)
- field potential analysis of electron beam tube (2D-FEM)
- magnetic analysis for JIEE model (3D-FDM)
- flow analysis of chemical vapor deposition (3D-FDM)
- magnetic field analysis of a disk head (3D-FEM)

Some of the models were quite complex, requiring hundreds of lines of DEQSOL input. The resulting Fortran programs were highly vectorized (more than 90 percent of the loops vectorized in each case).

In addition, DEQSOL has been evaluated by its authors for several application problems for which hand-coded Fortran programs exist. DEQSOL programs were 1/10th the size of these hand-coded Fortran programs, illustrating the potential for increased programmer productivity using DEQSOL. In some cases the DEQSOL programs ran faster than the hand-coded programs due to the high level of vectorization in the DEQSOL-produced codes.

An example of a moderately difficult problem which is easily expressed using DEQSOL's Enhanced System follows. The model is of viscous (low Reynold's number) incompressible fluid flow from a channel to a larger cavity in two dimensions (see Figure 1).

The equations representing the velocity vector (u, v) and the pressure p are the steady-state Stokes equations,

$$\begin{aligned} -\partial p/\partial x + \nabla^2 u &= 0 \\ -\partial p/\partial y + \nabla^2 v &= 0 \\ \partial u/\partial x + \partial v/\partial y &= 0. \end{aligned}$$

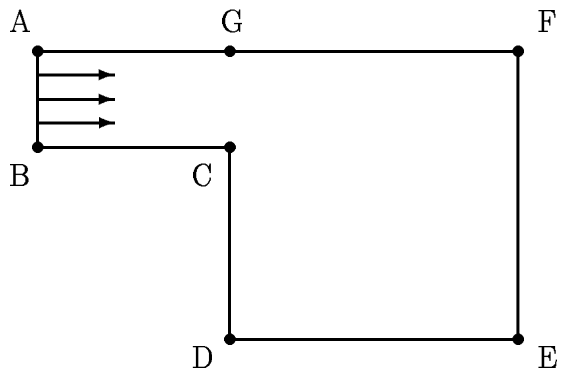

Figure 1: Domain for Stokes problem

The DEQSOL program for this problem, shown below, illustrates the ease with which simultaneous linear time-independent equations may be solved in DEQSOL. Note that DEQSOL assumes no-flux boundary conditions for all variables where boundary conditions are left unspecified. It also illustrates how finite element basis functions may be selected (linear for p, quadratic for u and v). A triangular mesh is automatically generated by DEQSOL; users control the meshing by specifying boundary nodes in the `MESH` statement. The phrase `CF=R(C,3,1.1)` inserts three nodes on the segment DE such that the ratio of distances between successive points is 1.1 (geometric series). The function `D` is used to specify equispaced nodes.

```
PROG     STOKES;
METHOD   FEM;
DOMAIN   X = [0:50],  Y=[0,30];
POINT    A = ( 0,30),  B = ( 0,20),  C = (20,20),  D = (20, 0),
         E = (50, 0),  F = (50,20),  G = (50,30),  H = (20,30);
REGION   AB = LN(A,B),  BC = LN(B,C),  CD = LN(C,D),
         DE = LN(D,E),  EF = LN(E,F),  FG = LN(F,G),
         GH = LN(G,H),  AH = LN(A,H),  CF = LN(C,F),
         CH = LN(C,H),
         ABCH = QUAD(AB,BC,CH,AH),  CFGH = QUAD(CF,FG,GH,CH),
         CDEF = QUAD(CD,DE,EF,CF);
FREGION  ABCH + CFGH + CDEF;
BOUND    AB + BC + CD + DE + EF + FG + GH + AH;
MESH     AB = D(A,2),       CH = D(C,2),       FG = D(G,2),
         DE = R(D,3,1.1),  AH = R(A,2,0.9),  CD = R(D,3,0.9),
         GH = R(H,3,1.1),  BC = R(B,2,0.9),  EF = R(E,3,0.9),
         CF = R(C,3,1.1),  INSIDE=AUTOSIM;
```

```
VAR      U, V, P;
ELMTYPE  LINEAR FOR P
         QUADRIC FOR U, V;
BCOND    U = -0.04*(Y-20)*(Y-30)  AT  AB,
         U = 0                    AT  BC + CD + DE + AH + GH,
         V = 0                    AT  AB,
         V = 0                    AT  BC + CD + DE + AH + GH,
         P = 0                    AT  EF + FG;
SCHEME;
             SOLVE  U, V, P  OF
                -DX(P) + LAPL(U) = 0
                -DY(P) + LAPL(V) = 0
                   DX(U) + DY(V) = 0  BY 'GAUSS';
             PRINT U, V, P;
END SCHEME;
END;
```

A still more complex example is presented in the appendix.

DEQSOL remains under active development at Hitachi. Continuing effort is being expended to make the user interface both more flexible and also easier to use, and to build databases of problems and solutions that will be useful as templates for users. Further work in workstation-mainframe interaction and in the application of DEQSOL to parallel computers is also underway.

3. ELLPACK

The ELLPACK project began in 1975 as a cooperative research project among Purdue University, the University of Texas at Austin, Yale University, and others. Headed by Professor John Rice of Purdue, the aim was to develop a research tool to evaluate and compare mathematical software for solving elliptic PDE problems. The idea was to create a system where individuals could contribute software modules which either completely solved a class of PDE problems or performed one of the major steps in a numerical method, e.g., discretization or linear algebraic equation solution. All the software would operate in a rigidly-defined environment. Some forty researchers and programmers have been associated with the project in one form or another since its outset. The development of the system was supported by the National Science Foundation, the Department of Energy, the Office of Naval Research, as well as many of the participant's institutions.

Considerable efforts to make the system easy to use lead to the development of the ELLPACK language and its preprocessor at Purdue. The resulting flexibility and ease-of-use of the system has made it a popular tool not only for the comparison of methods, but also for education and actual problem-solving. The first release of the ELLPACK occurred in 1978; the system was completely redesigned and re-released in 1984.

The ELLPACK system routinely solves linear elliptic equations with general linear boundary conditions in two and three dimensions. Arbitrary two-dimensional domains may be specified by describing the boundary in parametric form; holes and slits in the domain are allowed. Only rectangular boxes are supported in three dimensions. Unequally spaced rectangular grids are supported.

The equation to be solved, the domain, and the boundary conditions are expressed using a natural mathematical syntax. The reserved word `U` is defined to be the unknown function, and `UX`, `UY`, `UZ`, `UXX`, `UXY`, and so on are its partial derivatives. Expressions use Fortran syntax, and any Fortran or user-defined functions may be part of an expression.

Users specify the solution algorithm by simply naming a `DISCRETIZATION` module and a linear equation `SOLUTION` module or by naming a combination discretization-solution module (called `TRIPLE` modules, these are used to implement fast solvers for special problems). Other types of modules implement various utility tasks such as computing the eigenvalues of discretization matrices, displaying their non-zero structure, or reordering their rows and columns to provide for more efficient solution. Many problem-solving modules are available: 11 `DISCRETIZATION`, 18 `SOLUTION`, 13 `TRIPLE`, and 18 other. These may be easily combined in many ways to provide alternate PDE solution algorithms. Functions `U(X,Y)`, `UX(X,Y)`, etc. are automatically available after `SOLUTION` or `TRIPLE` modules execute to evaluate the solution or its derivatives. Both tabular and graphical output are supported. Graphics primitives are defined which allow ELLPACK to easily interface to a wide variety of graphical display systems.

One feature which considerably extends the range of applicability of ELLPACK is the fact that users may insert arbitrary Fortran code in ELLPACK programs. This allows expert users of the system to solve nonlinear problems, time-dependent problems, and systems of equations by designing iterations in which the basic operation is the solution of a single linear elliptic problem.

The form of the user interface to ELLPACK is illustrated by the following ELLPACK solutions to the two sample problems.

Problem 1 : Stress distribution in thin plate with hole

The problem is solved using finite element collocation on piecewise cubic rectangular elements with one continuous derivative. The generated equations are solved by Gaussian elimination for banded matrices.

```
EQUATION.   UXX + UYY = -2
OPTION.     CLOCKWISE=.TRUE.
BOUNDARY.   U  = 0  ON LINE 0,2 TO 2,2 TO 2,0
            UY = 0  ON LINE 2,0 TO 1,0
```

```
              U  = 1  ON X=COS(PI*T), Y=SIN(PI*T) FOR T=0 TO 0.5
              UX = 0  ON LINE 0,1 TO 0,2
GRID.         8 X POINTS  0 TO 2
              8 Y POINTS  0 TO 2
DISCRETIZE. COLLOCATION
SOLVE.        BAND GE
OUTPUT.       TABLE(U)
              PLOT(U)
END.
```

Problem 2 : Thermal diffusion problem

The problem is solved using a finite difference spatial discretization and a backward Euler implicit time step. At each time step the implicit equations are solved by a conjugate gradient iteration with Jacobi preconditioning. Note that `U(X,Y)` is, by definition, the most recent solution of the PDE problem. `SET (U = U0)` initializes `U(X,Y)` to be the function `U0(X,Y)` defined below.

```
GLOBAL.
              COMMON /PARAM/ ADT, ALAM, AMU
EQUATION.     UXX + UYY - ADT*U = -ADT*U(X,Y)
BOUNDARY.     UX =   0           ON  X = 0
                                 ON  X = 1
              U  = 200           ON  Y = 0
              UY - ALAM*U = AMU  ON  Y = 1.2
GRID.         5 X POINTS
              6 Y POINTS
TRIPLE.       SET (U = U0)
FORTRAN.
              ALAM = 0.8
              AMU  = 50.0
              A    = 0.62
              DT   = 0.02
              ADT  = 1.0/(A*DT)
              DO 100 ITER=1,50

DISCRETIZE.     5-POINT STAR
SOLVE.          JACOBI CG
FORTRAN.
                IF (MOD(ITER,10) .EQ. 0) THEN
OUTPUT.           TABLE(U)
FORTRAN.
                ENDIF
```

```
                I1LEVL = 0
  100       CONTINUE
SUBPROG.
      FUNCTION U0(X,Y)
      U0 = 100.0*EXP(-(X-0.5)**2-Y**2)
      RETURN
      END
END.
```

The entire ELLPACK system, including the preprocessor, is written in portable Fortran 77 and runs on nearly every type of computing device which supports Fortran. ELLPACK is an "open system"; a complete internal specification for the system is found in [Rice and Boisvert, 1985]. It was designed to be easy to modify and enhance, and others are free to do so. ELLPACK is distributed in source form for a nominal charge by the Purdue University Research Foundation. Since 1984 about 100 copies have been distributed in the US and about 60 outside the US. Proceeds from the distribution help to maintain a small staff which provides a nominal level of support for the system at Purdue.

The ELLPACK system is being used at Purdue as the basis for ongoing research in the design of problem-solving environments for scientific computing. Three such systems are currently under development: XELLPACK, Parallel ELLPACK, and Elliptic Expert.

The XELLPACK project has two goals: (a) to develop an interactive, graphical interface to ELLPACK for the X11 window system, and (b) to adapt the internal operation of the system to facilitate distributed computation. In XELLPACK problem-solving modules run as a separate processes, possibly on different machines. The X11 interface provides menu-based dialogue, with interactive grid generation and function plotting. XELLPACK is expected to be available for external distribution by early 1991.

The goal of the Parallel ELLPACK (//ELLPACK) project is to design a uniform programming environment for implementing parallel MIMD PDE solvers. A new user interface based upon X11 facilities for composing, editing, and executing //ELLPACK programs is being developed. These include graphics-oriented tools to specify the problem domain, to generate a grid, to decompose it for parallel execution, and to visualize the solution and data about the performance of parallel solution algorithms. General three-dimensional domains will be supported in this version. The user interface runs on an X11 workstation, with the PDE solution modules executing on a parallel machine connected through a wide bandwidth network. Specialized problem-solving modules for the NCUBE have been developed; modules for other parallel machines such as the SEQUENT and ALLIANT are planned.

In spite of its high-level user interface, ELLPACK users still face many difficulties when using the system. An example of this is the selection of modules to solve the PDE problem. ELLPACK's wealth of problem-solving modules admit 1147 distinct solution paths. For the casual user choosing a single valid path is difficult, while selecting the best method is nearly impossible. A rule-based expert system, called Elliptic Expert, is being developed to provide advice to users on problem formulation, algorithm and machine selection, and solution assessment. A prototype system based upon the OPS5 system is currently operational. A related system, called ATHENA, is being developed to provide expert advice to users of //ELLPACK. Plans for this system make extensive use of a database of performance profiles detailing the performance of solution algorithms for benchmark problems on available parallel machines.

4. Conclusions

DEQSOL and ELLPACK represent two successful approaches to the design of PDE problem-solving environments, each with its own domain of applicability.

The development of DEQSOL has been driven by the desire to solve PDE systems resulting from relatively complex mathematical models. Problems with challenging features such as those in fluid mechanics or problems with material interfaces are straightforward to express. The expression of solution algorithms for nonlinear, time dependent, and systems of equations is also more naturally achieved in DEQSOL. Much attention has been paid to obtaining high performance of the generated code on the Hitachi vector processor.

ELLPACK is better suited as an experimental tool such as in education, in solving small prototype problems, and in evaluating the performance of software modules. Simple linear problems are more succinctly expressed in ELLPACK. Its large library of problem-solving modules allows users to easily experiment with a variety of numerical methods and assess their performance. Its open system aspects allow users to easily modify the system for their own use, allowing the system to become a basis for experimental research in problem-solving environments for prototype mathematical modeling problems.

The reasons for the chosen emphasis in each of these projects arises from the driving forces behind them. DEQSOL, being a commercial product, must be useful to potential customers who need to solve realistic problems. Since DEQSOL is developed by a computer manufacturer perhaps there is little motivation to make it available for systems other than Hitachi's own. In fact, it can be claimed that it is necessary to keep the system closed to protect the company's investment. Unfortunately, this means that one group must be relied upon to follow all new developments in numerical methods for PDEs and to get them correctly implemented–a very difficult task. Further there is mounting evidence that open systems benefit everyone in the long run, including the developers. ELLPACK, on the other hand, was developed as a university-based research project under government support. Since there are no real "paying customers" there has been little motivation to demonstrate the performance of the system on difficult real-world applications, or to add features which would make the solution of such problems easier. Unfortunately, this has restricted the practical applicability of the system. The system's open, extendable nature follows from the desire to evaluate the performance of PDE software on model problems where it is necessary to be able to easily import external software. Portability is also crucial to reduce maintenance and to make the fruits of the research available to as wide an audience as possible. These properties are also necessary if the system is to be successful as the basis for on-going research in the design of problem-solving environments.

Surely, if the state of the art in mathematical problem-solving enviroments is to be advanced there is room for efforts of the types embodied by both DEQSOL and ELLPACK.

Acknowledgements

We would like to thank Dr. Yukio Umetani and Chisato Konno of Hitachi Ltd. for kindly supplying us with many details concerning the DEQSOL project.

Disclaimer

Certain commercial products are identified in this report in order to adequately describe the DEQSOL and ELLPACK systems. Identification of these products does not imply recommendation or endorsement by NIST, ONR, or any other agency of the US government, nor does it imply that the identified products are necessarily the best available for the purpose.

Bibliography

DEQSOL

(Further information about DEQSOL may be obtained from Dr. Yukio Umetani or Mr. Chisato Konno at Central Research Laboratory, Hitachi Ltd., Kokubunji, Tokyo 185, Japan. Phone: (0423) 23-1111, Fax: (0423) 23-1900, Email: UMETANI@CRL.HITACHI.CO.JP.)

C. Konno and Y. Umetani, A high level programming language for numerical simulation: DEQSOL, *IEEE Tokyo Section 25* (1986), 50–53.

C. Konno, M. Saji, N. Sagawa and Y. Umetani, Advanced implicit solution function of DEQSOL and its evaluation, *Proceedings of the Fall Joint Computer Conference* (1986), IEEE, 1026–1033.

C. Konno, M. Yamabe, M. Saji, N. Sagawa, Y. Umetani, H. Hirayama and T. Ohta, Automatic code generation method of DEQSOL (Differential EQuation SOlver Language), *Journal of Information Processing 11* (1987), 15–21.

Y. Umetani, M. Tsuji, K. Iwasawa, H. Hirayama, DEQSOL: A numerical simulation language for vector/parallel processors, in *Problem Solving Environments for Scientific Computing*, B. Ford and F. Chatelin editors, Elsevier Science Publishers B.V. (North-Holland), IFIP 1987.

C. Konno, M. Yamabe, M. Saji and Y. Umetani, The BF (boundary-fitted) coordinate transformation technique of DEQSOL (Differential EQuation SOlver Language), in *Parallel Processing for Scientific Computing* (G. Rodrigue, ed.), SIAM, Philadelphia, 1988.

C. Konno, Y. Umetani, M. Igai and T. Ohta, Interactive/visual DEQSOL: interactive creation, debugging, diagnosis, and visualization of numerical simulation, *Mathematics and Computers in Simulation 31* (1989), 353–369.

N. Sagawa, C. Konno and Y. Umetani, Numerical simulation language DEQSOL (Japanese), *Journal of the Information Processing Society of Japan 30* (1989), 36–45.

F. Yamamoto, Y. Umetani and M. Demoto, PARAGRAM: A High-Level Programming Language for Parallel Processors, *Systems and Computers in Japan 20* (1989), no. 8, 100–109.

ELLPACK

(Further information about ELLPACK may be obtained by writing to ELLPACK, Department of Computer Sciences, Purdue University, West Lafayette, IN 47907.)

R.F. Boisvert, E.N. Houstis and J.R. Rice, A system for performance evaluation of partial differential equations software, *IEEE Transactions on Software Engineering SE-5* (1979), 418–425.

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

J.R. Rice, W.R. Dyksen, E.N. Houstis and C.J. Ribbens, *ELLPACK Project Status Report*, CSD-TR-579, Computer Sciences Department, Purdue University, March 1986.

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

W.R. Dyksen and C.R. Gritter, Elliptic expert: an expert system for elliptic boundary value problems, *Mathematics and Computers in Simulation 31* (1989), 497–508.

E.N. Houstis, J.R. Rice and T. Papatheodorou, Parallel Ellpack: an expert system for parallel solution of partial differential equations, *Mathematics and Computers in Simulation 31* (1989), 497–508.

E.N. Houstis, J.R. Rice, N.P. Chrisochoides, H.C. Karathanasis, P.N. Papachiou, M.K. Samartzis, E.A. Vavalis, K.Y. Wang, and S. Weerawarana, //ELLPACK: a numerical simulation programming environment for parallel MIMD machines, *Proceedings of the International Conference on Supercomputing*, Amsterdam, 1990, to appear.

J. Bonomo and W.R. Dyksen, XELLPACK: An interactive problem- solving environment for elliptic partial differential equations, in *Intelligent Mathematical Software Systems* (E.N. Houstis, J.R. Rice, and R. Vichnevetsky, eds.), Elsevier, 1990, to appear.

Related work

A.F. Cardenas and W.J. Karplus, PDEL — A language for partial differential equations, *Communications of the ACM* (1970), 184–191.

M. Machura and R.A. Sweet, A survey of software for partial differential equations, *ACM Transactions on Mathematical Software 6* (1980), 461–488.

R.F. Boisvert and R.A. Sweet, Mathematical software for elliptic boundary value problems, in *Sources and Development of Mathematical Software* (W. Cowell, ed.), Prentice-Hall, Englewood Cliffs, NJ, 1984, 200–263.

G. Sewell, *Analysis of a Finite Element Method — PDE/PROTRAN*, Springer-Verlag, New York, 1985.

A.D. Kowalski, R.L. Peskin and M.F. Russo, Anatomy of AGNES: an automatic generator of numerical equations solutions, *Mathematics and Computers in Simulation 31* (1989), 343–352.

Appendix

The following illustrates the use of the DEQSOL Enhanced System for the solution of a more realistic model. In particular, we present a DEQSOL program for the solution of the nonlinear time-dependent Navier-Stokes equations for incompressible viscous fluid flow. The equations to be solved are

$$\begin{aligned} \partial u/\partial t &= -u\ \partial u/\partial x - v\ \partial u/\partial y - 1/\rho\ \partial p/\partial x + \nabla^2 u \\ \partial v/\partial t &= -u\ \partial v/\partial x - v\ \partial v/\partial y - 1/\rho\ \partial p/\partial y + \nabla^2 v \\ 0 &= \partial u/\partial x + \partial v/\partial y, \end{aligned}$$

where (u, v) is the velocity vector, p is the pressure, and ρ is the (constant) density. The equations are defined in a rectangular region $(0, 22) \times (0, 10)$ with a circular hole of radius 1 centered at $(5, 5)$. The hole could represent an obstruction in a pipe; the boundary conditions are chosen to model fluid flowing in from the left (see Figure 2).

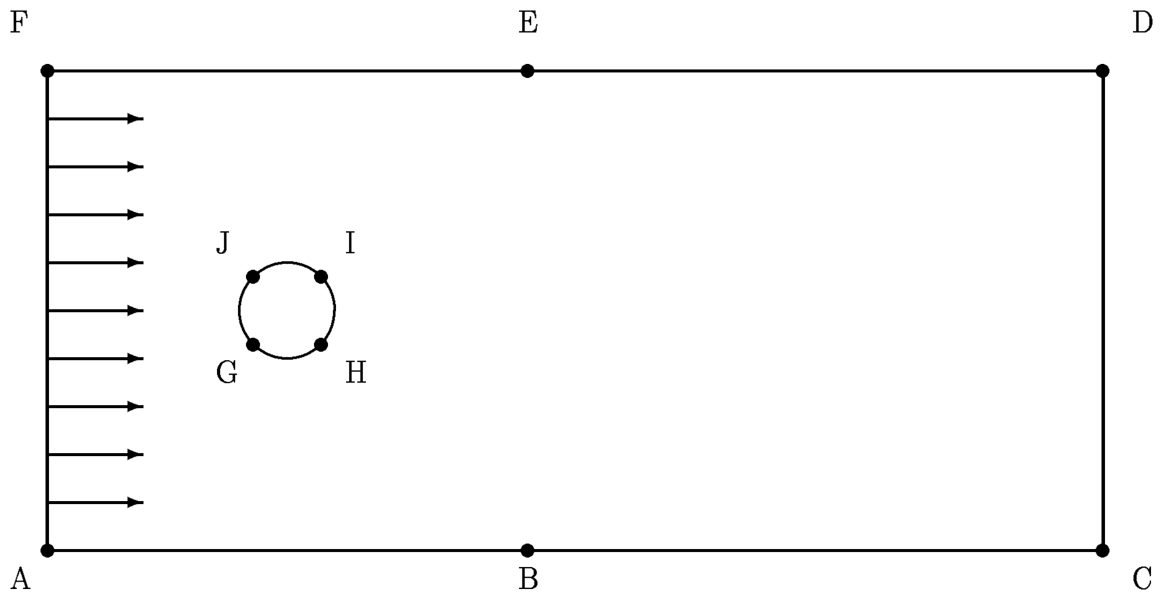

Figure 2: Domain for Navier-Stokes problem

In order to reduce discretization error in the numerical processing, it is desireable to transform the system to the following equivalent conservative form

$$\begin{aligned}
\partial u/\partial t &= -\partial[\, u^2 \,]/\partial x - \partial[\, uv \,]/\partial y - 1/\rho\ \partial p/\partial x + \operatorname{div}(A\ \nabla u) + \operatorname{div}(B\ \nabla v) && (1)\\
\partial v/\partial t &= -\partial[\, uv \,]/\partial x - \partial[\, v^2 \,]/\partial y - 1/\rho\ \partial p/\partial y + \operatorname{div}(B^T \nabla u) + \operatorname{div}(C\ \nabla v) && (2)\\
0 &= \partial u/\partial x + \partial v/\partial y, && (3)
\end{aligned}$$

where

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}.$$

Since the problem is time-dependent (and nonlinear), an iteration scheme for solving the problem must be expressed in the `SCHEME` statement. The following three-stage algorithm is used to advance the solution from time level n to $n+1$ (here superscripts refer to time levels):

(a) Predict (u, v) from equations (1) and (2) using explicit Euler steps:

$$\begin{aligned}
\bar{u} &= u^n - \Delta t\{\ \partial[\, (u^n)^2 \,]/\partial x + \partial[\, u^n v^n \,]/\partial y + 1/\rho\ \partial p^n/\partial x \\
&\qquad - \operatorname{div}(A\ \nabla u^n) - \operatorname{div}(B\ \nabla v^n)\ \ \} \\
\bar{v} &= v^n - \Delta t\{\ \partial[\, u^n v^n \,]/\partial x + \partial[\, (v^n)^2 \,]/\partial y + 1/\rho\ \partial p^n/\partial y \\
&\qquad - \operatorname{div}(B^T \nabla u^n) - \operatorname{div}(C\ \nabla v^n)\ \ \}
\end{aligned}$$

(b) Correct (u, v) so that (3) is satisfied:

$$\text{Solve} \quad \nabla^2\phi = -\partial\bar{u}/\partial x - \partial\bar{v}/\partial y$$

$$\begin{aligned} u^{n+1} &= \bar{u} + \partial\phi/\partial x \\ v^{n+1} &= \bar{v} + \partial\phi/\partial y \end{aligned}$$

(c) Correct p so that (1) and (2) are satisfied:

$$p^{n+1} = p^n - \rho\phi/\Delta t$$

The DEQSOL program which describes this problem and solution algorithm follows. The operator ".." is defined to be matrix-vector product, as in `A..GRAD(U)`. The function `N(W)` is defined for vectors `W` to be the dot product `N`·`W`, where `N` is the outward-pointing unit normal vector. The notation `U[LM]` refers to the variable `U` with the "lumping" option applied. Lumping is a common technique used in the finite element method to reduce discretization matrices to diagonal form, resulting in more efficient processing.

```
PROG     NAVIER;
METHOD   FEM;

/* ----------------------
   DEFINE TEMPORAL DOMAIN
   ---------------------- */

TIME     T   = [0:0.4];
TSTEP    DLT = [0(0.001)0.4];

/* ---------------------
   DEFINE SPATIAL DOMAIN
   --------------------- */

DOMAIN   X = [0:22],  Y = [0:10];

POINT    A = ( 0, 0),  B = (10, 0),  C = (22, 0),
         D = (22,10),  E = (10,10),  F = ( 0 10),
         G = (4.293,4.293),  H = (5.707,4.293),
         I = (5.707,5.707),  J = (4.293,5.707),
         K = (5,4), L = (6,5), M = (5,6), N = (4,5);

REGION   AB = LN(A,B),  BC = LN(B,C),  CD = LN(C,D),
         DE = LN(D,E),  EF = LN(E,F),  FA = LN(F,A),
```

```
         AG = LN(A,G),  BH = LN(B,H),
         EI = LN(E,I),  FJ = LN(F,J),
         BE = LN(B,E),
         GH = ARC(G,K,H),  HI = ARC(H,L,I),
         IJ = ARC(I,M,J),  JG = ARC(J,N,G),
         ABHG = QUAD(AB,BH,GH,AG),
         BEIH = QUAD(BE,EI,HI,BH),
         EFJI = QUAD(EF,FJ,IJ,EI),
         FAGJ = QUAD(FA,AG,JG,FJ),
         BCDE = QUAD(BC,CD,DE,BE);

FREGION  ABHG + BEIH + EFJI + FAGJ + BCDE;

/* --------------------------
   DEFINE FINITE ELEMENT MESH
   -------------------------- */

MESH     AB = D(A,10),  BC = D(B,12),  CD = D(C,15),
         DE = D(D,12),  EF = D(E,10),  FA = D(F,16),
         AG = R(A,15,0.8),  BH = R(B,15,0.8),
         EI = R(E,15,0.8),  FJ = R(F,15,0.8),
         BE = D(B,15),
         GH = D(H,10),  HI = D(I,15),
         IJ = D(J,10),  JG = D(G,16),
         INSIDE = AUTOSIM;

/* ----------------
   DEFINE VARIABLES
   ---------------- */

CONST    RHO = 1.2;

CTENS    A  = ( 2, 0, 0, 1 ),
         B  = ( 0, 0, 1, 0 ),
         BT = ( 0, 1, 0, 0 ),
         C  = ( 1, 0, 0, 2 );

VAR      U, V, P, PHI, U2, V2, UV, PSI;

VEC      VEL = (U,V),  PA = (P,0),  PC = (0,P);

ELMTYPE  LINEAR FOR U, V, PHI, PSI
```

```
        FLAT   FOR P;

/* -------------------
   BOUNDARY CONDITIONS
   ------------------- */

BCOND
              PSI  =  -1  AT AB+BC,
              PSI  =   1  AT DE+EF,
              PSI  =   0  AT GH+HI+IJ+JG,
        NGRAD(PSI) =   0  AT FA+CD,

        NGRAD(PHI) =   0  AT FA,
        NGRAD(PHI) =   0  AT AB+BC+DE+EF+GH+HI+IJ+JG,
              PHI  =   0  AT CD,

                U  = 100  AT FA,
                U  =   0  AT AB+BC+DE+EF+GH+HI+IJ+JG,

                V  =   0  AT FA,
                V  =   0  AT AB+BC+DE+EF+GH+HI+IJ+JG,

  N( (1/RHO)*PA - A ..GRAD(U) - B..GRAD(V) ) = 0  AT CD,
  N( (1/RHO)*PC - BT..GRAD(U) - C..GRAD(V) ) = 0  AT CD;


/* ---------------
   SOLUTION SCHEME
   --------------- */

SCHEME;

   /* SET INITIAL CONDITIONS */

   SOLVE PSI OF  LAPL(PSI) = 0  BY 'PCG';
   U =  DY(PSI);
   V = -DX(PSI);
   P =  0;

   /* TIME STEPS */

   ITER NT UNTIL EQ 401;
```

```
      U2 = U*U;    V2 = V*V;    UV = U*V;

      /* PREDICT (U,V) */

      U[LM] = U[LM] - DLT*( DX(U2) + DY(UV) + DX(P)/RHO
                     - DIV( A..GRAD(U)) - DIV(B..GRAD(V)) );
      V[LM] = V[LM] - DLT*( DX(UV) + DY(V2) + DY(P)/RHO
                     - DIV(BT..GRAD(U)) - DIV(C..GRAD(V)) );

      /* CORRECT (U,V) */

      SOLVE PHI OF  LAPL(PHI) = -(DX(U)+DY(V))  BY 'PCG';
      U[LM] = U[LM] + DX(PHI);
      V[LM] = V[LM] + DY(PHI);

      /* UPDATE P */

      P[LM] = P[LM] - RHO*PHI/DLT;

      /* OUTPUT */

      PRINT NT  EVERY 50 TIMES;
      PRINT VEL EVERY 50 TIMES;
      PRINT P   EVERY 50 TIMES;

   END ITER;

END SCHEME;
END;
```

Author's Biographies

Ronald F. Boisvert is a computer scientist in the Center for Computing and Applied Mathematics of the National Institute of Standards and Technology. He received a B.S. degree in mathematics from Keene State College (NH) in 1973, an M.S. in Applied Science from the College of William and Mary in 1975, and M.S. and Ph.D. degrees in Computer Science from Purdue University in 1975 and 1979. He is one of the original developers of the ELLPACK system for solving elliptic boundary-value problems, and currently leads the NIST Guide to Available Mathematical Software project. His research interests include numerical methods for partial differential equations, mathematical software, and expert advisory systems.

David K. Kahaner joined the staff of the Office of Naval Research Far East as a specialist in scientific computing in November 1989. He obtained his Ph.D. from Stevens Institute of Technology in 1968. From 1978 until 1989 he was a group leader in the Center for Computing and Applied Mathematics at the National Institute of Standards and Technology (formerly the National Bureau of Standards). From 1968 until 1979 he was in the Computing Division at Los Alamos National Laboratory. His major research interests are in the development of algorithms and associated software.