A SURVEY OF THE STATE OF SOFTWARE FOR PARTIAL DIFFERENTIAL EQUATIONS

by

Roland A. Sweet

STAN-CS-79-704
January 19 7 9

C O M P U T E R S C I E N C E D E P A R T M E N T
School of Humanities and Sciences
STANFORD UNIVERSITY

A SURVEY OF THE STATE OF SOFTWARE FOR PARTIAL DIFFERENTIAL EQUATIONS*

by

ROLAND A. SWEET+

*This paper was presented at the Texas Conference on Mathematical Software, University of Texas at Austin, March 29-31, 1978.

+Mathematics Department, University of Colorado at Denver, Denver, CO 80202, and National Center for Atmospheric Research, Boulder, CO 80307.

This work was supported by United States Army Grant No. DAAG29-77-G-0029.

Abstract.

This paper surveys the state of general purpose software for the solution of partial differential equations. A discussion of the purported capabilities of twenty-one programs is presented. No testing of the routines was performed.

1. Introduction

Mathematical software for partial differential equations (pde's) is a relatively new area. It is only within the last five years that programs have been published which are capable of solving relatively large classes of partial differential equations using effective numerical approximations.

The first attempts at general purpose software were oriented toward providing programming languages that facilitated writing the equation and its approximation (usually finite differences). Two examples of such languages were PDELAN [10] and PDEL [5]. These seem not to have been too widely used and are currently not supported.

Current popular software now is produced in the form of subroutines with user-provided subprograms defining the equation. There are, however, several notable exceptions to this that will be discussed in detail below.

This paper resulted from a reasonable extensive literature search undertaken to determine what software was available. The guiding principle in the selection process was generality. Hence, no coverage has been provided in the very large special application areas such as finite element codes for structural engineering, nuclear reactor codes, hydrodynamics codes (produced principally at government weapons laboratories), and petroleum reservoir modelling codes. Information on some of the programs in these areas may be obtained from the Argonne (National Laboratory) Code Center and the COSMIC program library at the University of Georgia computation center. Also, no attempt has been made to determine the existence of codes based on integral equation techniques.

Before proceeding further this caveat must be tendered: no attempt has been made to ascertain the availability, portability, utility, or accuracy

of any of the software mentioned in the remainder of this paper.

2. Mathematical Preliminaries.

In this section are discussed some mathematical ideas that are pertinent to the description of pde software.

For time dependent pde's the most popular approximation is the method of lines wherein the spatial derivatives are approximated in some manner in order to yield a system of ordinary differential equations (ode's). The time integration of this system of ode's is then accomplished by the use of existing software for ode's.

For example, suppose we wish to approximate the single equation

$$u_t = f(x,t,u_x,u_{xx}) \quad , \quad 0 < x < 1 \ , \quad t > 0 \tag{2.1}$$

with boundary conditions

$$u(0,t) = u(1,t) = 0 \ , \ t > 0 \tag{2.2}$$

and initial condition

$$u(x,0) = g(x) \ , 0 \leq x \leq 1 \quad . \tag{2.3}$$

We may choose a finite difference grid $\{x_i\}$ given by

$$x_1 = ih \ , \quad i = 0,1,2\ldots,N+1 \ , \quad h = \frac{1}{n+1} \ ,$$

and supposing $v_i(t)$ to be an approximation to $u(x_i,t)$, we replace u_x and u_{xx} in (2.1) by, say, second-order central finite difference approximations resulting in the ode system

$$\begin{aligned} \frac{dv_i}{dt} &= f(x_i,t,(v_{i+1}-v_{i-1})/2h,(v_{i-1}-2v_i+v_{i+1})/h^2) \\ &= F_i(t,v_{i-1},v_i,v_{i+1}) \ , \end{aligned} \tag{2.4}$$

for $i = 1,2,\ldots,N$. From (2.2) we have

$$v_0(t) = v_{n+1}(t) = 0$$

and from (2.3) we obtain the initial conditions

$$v_i(0) = g(x_i) \qquad i = 1,2,\ldots,N \ . \tag{2.5}$$

Equations (2.4) and (2.5) form an initial value ode problem which can be integrated by some ode package.

It is not necessary to use finite difference approximations to remove the spatial variation. One might assume that the solution u of (2.1) may be written

$$u(x,t) = \sum_{i=1}^{n} c_i(t) b_i(x)$$

where the known functions $\{ b_i(x) \}$ form a suitable basis. Depending on the choice of method, e.g. collocation or Galerkin, for removing the spatial variation one obtains an initial value problem for a system of ordinary differential equations in the unknowns $\{c_i(t)\}$.

Most developers of pde software who have chosen to use the method of lines rely heavily on existing software for ode's and for the computation of the basis functions. Specifically the ode software used almost exclusively is one of the several codes based on Gear's method written by Hindmarsh [12,13,14]. The basis functions chosen are frequently B-splines and the package of de Boor [2] is used.

The use of the method of lines for two-space dimension parabolic pde's introduces a significant complication in that it is generally accepted that the ode system to be solved may be stiff. Hence, this dictates the use of implicit methods which give rise to very large, sparse Jacobians which must be inverted. The inversion of such matrices is not at all trivial and seems to be

the principle obstacle in the development of multi-space dimension pde software. This same problem exists for the development of robust software for elliptic pde's defined over general regions.

3. Software for Hyperbolic Systems

Hyperbolic equations are, by far, the most difficult class of partial differential equations to approximate numerically. The principal difficulties lie in determining the correct approximation at the boundaries of the domain and accurately representing discontinuities in the solution.

To illustrate the former difficulty, suppose we want to approximate the solution of the simple system

$$u_t = Au_x \qquad (3.1)$$

where $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$, $A = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, $0 < x < 1$.

It is easily seen that the general solution is $u_1(x,t) = f(t+x)$ and $u_2(x,t) = g(t-x)$, where f and g are arbitrary functions determined by the boundary and initial conditions. Clearly, at the boundary $x = 0$, (if $t < 1$), u_1 must be specified and u_2 cannot be specified unless the solution is known. At $x = 1$, u_2 must be specified, and u_1 cannot be.

However, a straightforward finite difference approximation to equation (3.1) requires that two conditions be given at each boundary. The problem here, of course, is that the approximation near the boundary must take into account the nature of the characteristics, i.e., the lines $x + t$ = constant and $x - t$ = constant. At $x = 0$, the characteristics indicate that u_2 should be computed from known interior values, e.g., by extrapolation along the outgoing characteristics $x + t$ = constant. Analogous statements apply at the boundary $x = 1$.

Although the preceding comments may seem rather obvious for the system

(3.1), where A is a constant diagonal matrix, the construction of accurate difference approximations becomes more difficult when A is a function of x, t, and u . In this case the characteristics must be computed by an eigenvalue/eigenvector routine at each time step and at each boundary.

3.1 DCG

This program, written by Engquist and Smedsaas [9] , was designed to provide software for the numerical solution of hyperbolic systems in one space dimension. The user specifies symbolically the equation

$$u_t = f(x,t,u,u_x) ,$$

where $a < x < b$, and $u \in R^n$. Initial conditions, boundary conditions, and certain characteristics of the problem, e.g. uniform or non-uniform grid in space, or that some particular terms should be considered stiff (and, therefore, integrated implicitly in time) are also specified. These specifications are accepted by the first part of DCG, the analyser. A syntactical analysis of the equations and boundary conditions is performed to detect errors in the specification and the problem is classified by linearity, time and space dependence of coefficients, etc. The principal symbol, i.e., the matrix $\partial f/\partial u_x$, is examined to determine, as far as possible symbolically, whether the correct number of boundary conditions have been specified. FORTRAN code is prepared to compute the eigenvalue and eigenvectors during execution, if necessary. (This part of the 'program is the analysis of the matrix analogous to the matrix A in equation (3.1)). Difference methods and boundary strategies, i.e. which variables should be extrapolated at the boundaries, are selected according to the preceding analysis and user specification, and a solution algorithm is determined. This algorithm is then optimized.

The second part of DCG, the synthesizer, accepts the algorithm, and,

using a library of FORTRAN codes, constructs a FORTRAN program to solve the given problem. This program contains output statements that save all intermediate execution output on a disk. After execution, a graphics program operates interactively with the user to *provide graphs of any of the variables as functions of space or time.

As alluded to above, the approximation Is done by finite differences using different types of second and fourth order differencing, Leapfrog, Crank-Nicolson, or semi-implicit integration, all of which are energy conserving. However, the user may specify dissipation if desired.

DCG seems to be a fairly thorough treatment of the one-space-dimension problem, but its availability may be limited by the fact that the analyzer is written in SIMULA , a language not generally available in numerical computation centers.

3.2 RKFPDE

This subroutine, written by Gary [11], was motivated by a desire to provide software for a class of problems commonly called the "stream function-vorticity equation". Problems in this class occur frequently at the National Center for Atmospheric Research, where this software was developed.

The system of equations is assumed to have the form

$$u_t = f_x + g_y + h(x,y,t,w,w_x,w_y,w_{xx},w_{xy},w_{yy},w_{xxxx},w_{yyyy})$$

plus an optional elliptic equation

$$Lp = F(x,y,t,u,u_x,u_{xx},u_{xy},u_{xxxx},u_{yyyy})$$

where L is assumed to be separable, and where $u \in R^n$, $w = (u_1,u_2,\cdots,u_n, p)$, f and g are functions of x,y,t, and w , The domain is assumed to be a rectangle in the plane. On the boundaries of the rectangle the user may

specify one of the conditions:

1) periodicity

2) symmetry with respect to the boundary

3) skew symmetry with respect to the boundary

4) a mixed condition, or

5) extrapolation of the characteristic variables.

(This last boundary condition requires that the user determine those quantities which should not be specified at a boundary).

The solution is accomplished by a finite difference approximation on a user-supplied uniform or non-uniform mesh. The method of lines is used. The user may select second or fourth order central differences in space. The time integration is carried out by a Runge-Kutta-Fehlberg 3-4 scheme modified from a routine provided by Shampine. The elliptic equation is solved by the subroutine SEPELI, a part of FISHPAK (cf. 5.1). The functions f,g,h, boundary data, and the coefficients of the elliptic operator are given in user-supplied subprograms.

This software was written for the Control Data 7600 and designed to use extended core memory. However, portability should not be too affected as the calls to this memory have been isolated in the code and may be replaced by statements appropriate for the user's computer. A version of this code is currently running on the CRAY-1.

4. Software for Parabolic Systems.

In contrast to the lack of general purpose codes for hyperbolic systems there are a number of programs available for parabolic systems. The method of lines is used exclusively in these codes. Hence, the discussion of the approximation in each code will be separated into two parts: space and time. Nine codes will be discussed. They will be presented in an order corresponding

to increasing capabilities of the programs. (One program, GENEPI, may also be used for solving a parabolic equation, see 5.9).

4.1 DSS/2

This package, produced by Schiesser [28], is a successor to an earlier code LEANS [27]. DSS/2 is primarily an educational tool. It was designed as a main program which the user supplies with three subroutines that define the actual pde to be solved. It is assumed that the pde can be written in the form

$$u_t = F(x,t,u,u_x,u_{xx}) \quad , \quad a < x < b \ , \ t > 0 \qquad (4.1.1)$$

where $u \in R^n$. (The expressions u_x and u_{xx} mean the vectors of first and second partial derivatives, resp.) The boundary conditions are not provided in a separate subroutine, but rather, are incorporated by the user into the subroutine that defines F . It appears, therefore, that only relatively simple boundary conditions may be handled with ease.

In order to evaluate F , the user selects from a large collection of subprograms within DSS/2 the one that automatically computes finite difference approximations to u_x or u_{xx} . Centered and non-centered differences of even order two through ten are available.

The time integration scheme may be chosen from one of fourteen different Runge-Kutta one-step methods of orders one through five, or the user may select the Hindmarsh GEARB [14] package.

4.2 PDEPACK

This software was developed by Sincovec and Madsen [22], and may be purchased or leased from Scientific Computing Consulting Services. The package consists of eleven subroutines with communication of the problem through user-provided subroutines. The system of pde's is assumed to be in the form

$$u_t^{(k)} = f^{(k)}\left(t,x,u,u_x, \frac{1}{x^c}\left[x^c D_{k,1} u_x^{(1)}\right]_x, \ldots, \frac{1}{x^c}\left[x^c D_{k,n} u_x^{(n)}\right]_x\right) \quad (4.2.1)$$

where

$$k = 1,2,\ldots,n,\ a < x < b,\ t > t_0, \qquad D_{k,j} = D_{k,j}(t,x,u),$$

and c may be 0,1, or 2 .Terms of the form

$$\frac{1}{x^c}\left[x^c D_{k,j} u_x^{(j)}\right]_x$$

allow the program to handle discontinuous coefficients, e.g., when a problem has material interfaces. The inclusion of factors x^c facilitates the treatment of common expressions occurring in Cartesian, cylindrical, or spherical coordinates.

The boundary conditions are assumed to be prescribed at each boundary and to be in the form

$$\alpha_k u^{(k)} + \beta_k u_x^{(k)} = \gamma_k, \quad k = 1,2,\ldots,n .$$

If $\beta_k \neq 0$, then α_k and γ_k may be functions of t and u , but if $\beta_k = 0$, α_k and γ_k may only be functions of t .

The space discretization is accomplished via central finite differences on a grid of points specified by the user. The differences are second order on a uniform grid, but only first order on a non-uniform grid. The differencing has been designed to conserve such quantities as Du_x across material interfaces. The finite-difference grid may not change with time.

The time integration is accomplished by use of a modification of the Hindmarsh package GEARB.

4.3 PDECOL

This software was also produced by Sincovec and Madsen [20]. The user specifies the problem via a set of subroutines and calls the subroutine PDECOL for the solution.

The equation is assumed to be in the form

$$u_t = f\ (t,x,u,u_x,u_{xx})\ ,$$

where $u \in R^n$, $a < x < b$, and $t > t_0$. The boundary conditions are assumed to be in the form

$$b(u,u_x) = z(t)\ .$$

However, the user need not specify a boundary condition at all. This feature may be useful if the user attempts to solve a hyperbolic system or a coupled ode-pde system.

The spatial approximation is accomplished by the use of collocation using B-splines as the basis. The breakpoints for the B-splines are provided by the user, as are the degrees of the piecewise polynomials and the order of continuity. The B-splines are computed using the package of de Boor [2]. The collocation points are chosen automatically by the software.

Since the resulting system of ordinary differential equations is implicit, the authors use GEARIB, a modification of GEARB developed by Hindmarsh [14].

4.4 MOL1D

This subroutine package was developed by Hyman [17]. As in the previous two codes, the user writes subroutines defining the problem and then calls MOL1D. The equation must be in the form

$$u_t = g(x,t,u,u_x,u_{xx},f_x) \qquad (4.4.1)$$

where $u \in R^n$, $a < x < b$, $t > t_0$, and $f = f(x,t,u,u_x,u_{xx})$. The inclusion of the argument f_x on the right side of (4.4.1) allows conservative differencing of advective terms which arise naturally in fluid dynamics equations.

The acceptable boundary conditions are given by:

a) periodicity in x,

b) $au + bu_x = c$, where a,b, and c may be functions of t and u if $b \neq 0$, otherwise a and c may only be functions of t ,

c) some differential equation, a free boundary, or no boundary condition prescribed.

The spatial discretization is accomplished automatically using second , fourth, or sixth order centered differences, fast Fourier transform approximation (in the case of periodicity only) with or without linear filtering of higher modes, or unsymmetric second or third order differences.

The time integration is carried out by the GEARB package.

Graphical output is available. The plotting procedures use standard CALCOMP routines, and, as such, represent a departure from the portability standard the author has set.

4.5 LSQPDE

This software by Eason and Mote [8] is designed to solve the equation

$$u_t = F(x,t,u,u^{(1)}, u^{(2)}, \ldots) u^{(p)})$$

where $a < x < b$, $t > t_0$, $u \in R^n$, and $u^{(j)}$ represents the vector of partial derivatives of u of order j . At the boundaries, general conditions of the form

$$G_i(x,t,u,u^{(1)}, \ldots, u^{(p-1)}) = 0 \qquad x = a,b;\ i = 1,2,\ldots,q$$

are assumed to hold.

The approximation technique is a combination of time integration and least square approximation. The user selects a set of points $\{x_i\}_{i=1}^{n}$ in

the interior and on the boundary of [a,b] at which points the equation will be integrated in time. A particular set of basis functions (provided by the user) is used to represent an approximation $v(a_j,x)$ to the solution $u(x,t_j)$, where a_j represents the vector of coefficients in the approximation at time t_j. The first step of the algorithm is to form a vector of residuals, evaluated at the points $\{x_i\}$, for initial and boundary conditions. The vector a_0, corresponding to the initial time t_0, is determined by minimizing the residual in the least squares sense. The minimization is accomplished by Powell's non-linear least square technique. Assuming a_j is known, the approximation $v(a_j,x)$ is used in a modified Gear-Hindmarsh predictor-corrector algorithm. Firstly, values of the solution are predicted at time level t_{j+1} by a Taylor series expansion of $v(a_j,x)$ in time. These predicted values are used in conjunction with the corrector equation to form a vector of residuals at the points $\{x_i\}$. This vector is a function of the unknown coefficient vector a_{j+1}, which is determined by a least squares minimization of the residual vector.

4.6 POST

This software, written by Schryer [29], is available through the PORT library of Bell Laboratories. It is designed to solve coupled ordinary -partial differential equation systems. The equation treated is

$$f(x,t,u,u_x,u_t,u_{xt}) = a(x,t,u,u_t,u_{xt})_x$$

where $a \leq x \leq b$, $u \in R^n$, with boundary conditions

$$b(t,u,u_x,u_t,u_{xt}) = 0 \qquad \text{at} \quad x = a,b.$$

It is also possible to impose non-local conditions on the solution such as periodicity or that the integral of the solution over someinterval assumes

a given value.

The spatial variation is approximated using a Galerkin technique with a B-spline basis (provided by the de Boor package). The time integration is a one-step method, explicit or implicit, coupled with an extrapolation algorithm to provide automatic error control in the time integration.

The user provides a call to subroutine POSTS providing information about the B-spline basis, time integration limits, and an error tolerance. He also provides two subroutines for the evaluation of the equation and boundary conditions.

4.7 PDETWO

PDETWO was written by Melgaard and Sincovec [21]. It is a collection of subroutines that provides an interface between a two dimensional pde and an ode integration package. The user specifies the problem by providing subroutines and calls PDETWO for a solution.

The equation is assumed to have the form

$$u_t^{(k)} = f^{(k)}(t,x,y,u,u_x,\ (DH_{k,1}\ u_x^{(1)})_x,\ldots,(DH_{k,n}\ u_x^{(n)})_x\ ,$$

$$(DV_{k,1}\ u_y^{(1)})_y,\ldots\ ,\ (DV_{k,n}\ u_y^{(n)})_y\ )$$

where $a_1 < x < b_1$, $a_2 < y < b_2$, $t > t_0$, $k = 1,2,\ldots,n$, and the coefficients $DH_{k,j}$ and $DV_{k,j}$ are piecewise continuous functions of t,x,y and u.

Boundary conditions are assumed to have the form

$$a_k u^{(k)} + b_k u_n^{(k)} = c_k\ , \qquad k = 1,2,\ldots,n\ ,$$

where a_k, b_k, and c_k are piecewise continuous functions and u_n represents a derivative normal to the boundary.

The user provides a two dimensional finite difference grid and the package automatically produces an approximation at each grid point corresponding to a five-point star.

The time integration is accomplished by the use of GEARB. The use of Newton's method to solve the system of non-linear equations resulting from the use of implicit integration schemes necessitates the solution at each Newton iteration of a large sparse banded system of equations. The coefficient matrix, the Jacobian of the non-linear system, must be evaluated at each iteration and represents a significant computational expense. The authors have examined various techniques for this task and have in their view implemented the most efficient one.

4.8 DISPL

This software package is the collaborative effort of Leaf, Minkoff, Byrne, Bleakney and Saltzman [19] of the Argonne National Laboratory. The package is designed specifically for one and two spatially dimensioned kinetics-diffusion equations. It represents an effort to provide reasonably general purpose software designed for a particular application.

The equation is assumed to thave the form

$$[\rho C_p 1_k(t,r,z,\vec{u})\ u_t^{(k)} + \theta\ \nabla\cdot(\vec{v}_k(t,r,z,u)u^{(k)}) + (1-\theta)\vec{v}_k(t,r,z,\vec{u})\cdot\nabla\ u^{(k)}$$

$$= \nabla\cdot(\vec{D}_k(t,r,z,\vec{u},\vec{\nabla u})\ \nabla u^{(k)}) + \sum_{i=1}^{n} c_{ki}\ u^{(i)} + \sum_{i,j=1}^{n} d_{kij}\ u^{(i)}u^{(j)}$$

$$+ f_k(t,r,z,\vec{u},\vec{\nabla u})\ ,$$

where $k = 1,2,\ldots,n$. The equation is assumed to hold over a rectangle with material interfaces in the (r,z)-plane. To increase the applicability of the code, two forms of advection terms ($\theta = 0$ or $\theta = 1$) are allowed.

The boundary conditions on the sides of the rectangle are assumed to have the form

$$\alpha\, hu^{(k)} + \beta\, \vec{D}_k \nabla u^{(k)} \cdot \vec{n} = \gamma\, h\rho_k^0$$

where α, β, and γ may depend on k and the side of the rectangle, $\vec{n}$ is the exterior unit normal, and ρ_k^0 may depend on $\vec{u}$ and $\nabla\vec{u}\cdot\vec{n}$.

A Galerkin procedure is used as the spatial approximation using B-splines as the basis functions. The mesh is provided by the user. The B-splines are computed using de Boor's package.

The time integration uses the same method as that found in Hindmarsh's GEARIB [13], although the authors do not use GEARIB itself.

Extensive graphics capabilities have been provided. The software has been written in MORTRAN -- a FORTRAN preprocessor. This should present no problems since the authors are including in the external distribution a portable copy of the MORTRAN translator.

4.9 FORSIM VI

As the name indicates, the FORSIM package of Carver et al [6] is in the sixth edition, a testament to the longevity of this software effort.

This package contains a main program plus a large number of subprograms that provide centered, non-centered, and upwind finite difference approximations of various orders and cubic spline approximations. Time integration methods provided are a variable step Runge-Kutta-Fehlberg, a fixed step fourth order Runge-Kutta with Fehlberg corrections, a fixed step Euler, and the Hindmarsh-Gear algorithm. For this last algorithm the user may select a number of different approximations to the Jacobian, ranging from no Jacobian (functional iteration) to the full Jacobian, in which case the sparse matrix package of

Curtis and Reid [7] is used.

To use this package, the user provides a subroutine UPDATE, that contains storage declaration, initial conditions, equation and boundary condition specifications, and output specifications. To define the pde the user selects those subprograms corresponding to the spatial discretization, creates a section of code which performs the evaluation of the right side of the equation, e.g., the F in (4.1.1), and incorporates this into UPDATE. The time integration is performed by a call within UPDATE to another subroutine.

This package may also be used to solve two- and three-space dimension problems defined on rectangles or parallelepipeds, respectively. Graphical output is also available.

5. Software for Elliptic Equations.

Due to the extensive theory that exists for elliptic partial differential equations and the large amount of work that has been done on the numerical approximation of the solutions of such equations, it is not surprising to see a large selection of software in this area. However, it is true that for complicated problems, e.g. non-linear or non-separable, the solution techniques are not as reliable and software is not generally available.

The software packages will be described in an order corresponding to increasing complexity of the type of problems solved.

5.1 FISHPAK (Version 3)

This package of subroutines was written by Adams, Swarztrauber and Sweet [1] at the National Center for Atmospheric Research, Boulder, Colorado. It is a continuing effort brought about originally in response to the need for a complete set of readily available well-tested, reliable, efficient, and well-documented software to solve a subclass of elliptic equations which occur frequently in the study of geophysical fluids.

The subclass consists of separable elliptic equations with particular emphasis on the Poisson equation defined on a rectangle in Cartesian, cylindrical or spherical coordinates, and with Dirichlet, Neumann or periodic boundary conditions prescribed. Of particular importance was the need to automatically treat coordinate singularities, e.g., the origin in spherical coordinates, and equation singularities, e.g., fully periodic solutions.

The package consists of eighteen subroutines and one subpackage. These are:

(a) Twelve drivers that define second order, central finite difference approximations on staggered and unstaggered uniform grids, incorporate boundary data, and treat singularities for two-dimensional modified Helmholtz equations in Cartesian, polar, cylindrical, surface spherical, and spherical cross-section coordinate systems, for a three-dimensional Helmholtz equation, and a general separable two-dimensional elliptic equation without coordinate singularities,

(b) six solvers that are used to solve the linear systems of equations arising above,and two solvers that can be used to solve finite difference approximations to complex-valued separable elliptic equations, and

(c) a subpackage of fast Fourier transform routines that provide periodic, sine, cosine, sine quarter wave, and cosine quarter wave transforms as well as the full complex transform.

The solution techniques are based on generalizations of the Buneman variant of cyclic odd-even reduction [3] and some of the routines provide, at the users option, fourth order approximations using the method of deferred corrections.

Buzbee et al [4] tested version 1 of the package and made criticisms

and suggestions [30] to the authors. These suggestions were implemented by the authors resulting in the second version of the package.

5.2 A Package for the Helmholtz Equation on an Arbitrary Region.

This package of four subroutines was written by Proskurowski [23] at the Lawrence BerkeleyLaboratories, Berkeley, California. It may be used to solve the Helmholtz equation

$$\nabla^2 u + cu = f \tag{5.1}$$

defined on a general bounded planar region Ω with either Dirichlet or Neumann conditions prescribed on the boundary of the region.

The package solves the standard, second order, central finite difference approximation to equation (5.1) by embedding the region in a rectangle R and using the capacitance matrix technique coupled with fast Poisson solvers. Briefly, the method consists of four steps:

1) generate the capacitance matrix C the order of which is equal to the number of grid points within Ω and adjacent to the boundary,

2) solve (5.1) on R with f arbitrarily extended to R using a fast Poisson solver, e.g., a routine from FISHPAK,

3) solve the capacitance matrix equation

$$Cz = b \tag{5.2}$$

and use z to correct the values of f , and

4) solve (5.1) on R with the corrected f .

For a general region, step 3) can be very time consuming since the order of C may be large. This also may make the direct internal storage of C prohibitive.

To efficiently overcome these difficulties, the package provides four subroutines:

1) HLMHLZ solves equation (5.2) implicitly via a conjugant gradient iteration, thus obviating the storage of C ,

2) HELMIT generates and stores C explicitly and solves (5.2) directly,

3) HELMSIX solves the Dirichlet problem only storing C explicitly and has the option of obtaining fourth or sixth order approximations via deferred corrections, and

4) HELSYM produces a symmetric approximation to the Dirichlet problem, using an explicitly generated C . This routine may be used in conjunction with an algorithm for the computation of eigenvalues and eigenvectors of large symmetric matrices, e.g., the Lanczos method, in order to approximate the eigenvalues of the Laplacian.

To describe the irregular boundary, the user must supply a subroutine, DOMAIN, that specifies the coordinates of the irregular grid points, i.e. those points on the finite difference grid for which some of its neighbors are not within the domain. The user must specify the distance from the grid point to the boundary and the associated boundary values. The right side of equation (5.1) is furnished in the user-supplied subroutine CHARGE.

5.3 EIGEN

Ryder and Sanderson [26] have developed a package for approximating the eigenvalues of Laplace's equation. Specifically, the program finds approximate solutions to

$$\nabla^2 u + \omega^2 u = 0 \qquad (5.3)$$

defined on a bounded, simply connected domain in the plane, subject to Dirichlet or Neumann boundary conditions on various portions of the boundary of the domain.

A major difficulty in the approximation is the treatment of singularities in the solution induced by re-entrant corners, i.e. an interior angle greater than π, on the boundary. They overcome this difficulty by seeking solutions to equation (5.3) of the form

$$u(r,\theta) = \sum_{i=1}^{k} \sum_{j=0}^{N_i} \left(c_{ij} J_{\alpha_{ij}}(\omega r_i) \sin \alpha_{ij} \theta_i + d_{ij} J_{\alpha_{ij}}(\omega r_i) \cos \alpha_{ij} \theta_i \right) \qquad (5.4)$$

where k is the number of corners on the boundary, α_{ij} are determined from the angle of the $i^{\underline{th}}$ corner, and (r_i, θ_i) are the coordinates of the point (r,θ) in a polar coordinate system centered at the $i^{\underline{th}}$ corner. Since each term of equation (5.4) is a solution of equation (5.3), the coefficients c_{ij} and d_{ij} are determined by attempting to satisfy the given boundary condition in a least squares technique. It is known also that such an expansion of the solution produces the correct asymptotic singularity at each corner.

The procedure used is to minimize a certain error functional in ω by selecting an ω, solving for the c_{ij} and d_{ij}, then trying to correct ω by minimizing the residual of the least squares solution. Such local minima are "strong candidates" for eigenvalues even though convergence proofs for this technique do not exist. The routine EIGEN uses the routine LOCALM written by R. Brent for finding local minima, and the routine DSVD written by P. Businger, for finding the singular value decomposition of the matrix.

5.4 ITPACK/REGION

This software [18] is a result of the continuing research at the Center for Numerical Analysis of the University of Texas at Austin. It is a package of routines which approximate the solution of the linear, self-adjoint elliptic equation

$$(au_x)_x + (bu_y)_y + cu = f \qquad (5.5)$$

defined on a somewhat general region in the plane, with Dirichlet boundary conditions assumed on the boundary of the region. The coefficients a,b, and c may be functions of x and y .

The solution is approximated using second order central finite differences defined on a uniform grid with equal spacing in x and y . The boundary of the region over which the equation is assumed to hold may consist of horizontal or vertical grid lines or lines of slope ± 1 connecting grid points. The region may have holes so long as the boundaries of the holes satisfy the above conditions. Such restrictions eliminate the occurence of irregular grid points near boundaries.

To solve the sparse linear system arising from the approximation to equation (5.5), the user may select one of six iterative algorithms:

1. Jacobi iteration with Chebyshev acceleration.
2. Compresssed Jacobi iteration with conjugate gradient acceleration.
3. Jacobi iteration on a reduced system with Chebyshev acceleration.
4. Jacobi iteration on a reduced system with conjugant gradient acceleration.
5. Symmetric successive overrelaxation with Chebyshev acceleration.
6. Symmetric successive overrelaxation with conjugant gradient acceleration.

The selection of the acceleration parameter and the stopping criterion are automatic.

The subroutine REGION defines the grid after the user has supplied a polynomial parameterization of the boundary.

5.5 POTENT

This piece of software, written by Thomas [32], was produced to aid engineers in the solution of problems in electrostatics and magnetostatics.

The equation, written in divergence form, is

$$\nabla\cdot(\epsilon(x,y)\ \nabla u) = (\epsilon u_x)_x + (\epsilon u_y)_y = f(x,y) \qquad (5.6)$$

that is assumed to hold on a general, bounded planar region. Dirichlet or mixed boundary conditions are assumed to hold on the boundary.

The method of approximation is to use a finite difference approximation to equation (5.6) and then solve the resulting system by ADI, point SOR, or line SOR with the acceleration parameter either specified by the user or set internally (by assuming Dirichlet boundary conditions for Laplace's equation defined on a bounding rectangle).

The user must provide a subprogram which declares whether a point is inside, on, or outside the boundary, and , in the latter case, its distance to the boundary. Once the boundary has been determined it is outputed graphically for the user to check for errors.

5.6 ELLPACK 77

This software research project is coordinated by Rice [24]. It is a cooperative effort among many people interested in the development and evaluation of algorithms related to solving the large, sparse linear systems of equations arising from approximations-to linear elliptic equations. The goal of the project is to facilitate the testing of algorithms that only deal with a portion of the total solution process. This is done by defining a fixed number of subproblems (modules), and defining fixed interfaces between these modules. 'There are currently four modules: equation formation, equation indexing, equation solution, and output. Researchers may contribute software designed for one of the modules. More importantly, they may select existing software from the other modules for testing their own software, thereby relieving themselves of the burden of unnecessary coding. The interested

reader is referred to the above reference for a detailed discussion of the ELLPACK project.

5.7 ELLCOL

The first piece of software from the ELLPACK project has been designed by Houstis and Rice [16]. It is designed to solve the linear equation

$$\alpha u_{xx} + 2\beta u_{xy} + \gamma u_{y-y} + \delta u_x + EU_y + \zeta u = f$$

defined on a general two-dimensional region with the boundary condition

$$au_x + bu_y + cu = g$$

assumed on the boundary. The coefficients $\alpha, \beta, \ldots \zeta, a, b,$ and c may be functions of x and y .

The method of approximation is to embed the region in a rectangular grid and then use collocation at four Gaussian points within each rectangle, using bicubic piecewise Hermite polynomials for the basis. The approximation is also required to interpolate the boundary condition at "appropriate" points. The system is solved using profile Gauss elimination.

To use this software, the user must provide information to the ELLPACK input and output modules. For input the user provides:

1. subroutine BCOORD that provides a parametric representation of the boundary of the region,

2. functions COEF and BCOEF that provide the equation and boundary condition coefficients, and

3. function F that provides the functions f and g .

The user may select various levels of output of intermediate results, approximations to u_x, u_y, and u_{xy}, and the execution time.

5.8 EPDE1

This main program was writ-ten by Hornsby [15] at CERN. It is designed to approximate the solution of

$$au_{xx} + bu_{yy} + cu_x \text{ t } du_y + eu = f$$

where a,b,...,f may be functions of x,y, and u . The equation is assumed to hold over a region bounded by a simple closed curves $C_1, C_2, .., C_k$, where $C_2, ..., C_k$ must lie within C_1 , Along the curves C_j the following boundary conditions are assumed to hold:

1. Dirichlet, or

2. if C_i is a line which coincides with the finite difference grid, then a mixed condition

$$pu_x + qu_y + ru = s$$

may hold.

A uniform finite difference grid is defined, and at each grid point a finite difference equation is developed using the four neighboring 'points. The resulting system of equations is solved by point SOR. The acceleration parameter ω is estimated using the method of Carré. The convergence criterion, based on estimates of the spectral radius of the iteration matrix involves a percentage accuracy specified by the user. If the equations are non-linear, functional iteration is coupled with the SOR iteration.

The input for this program is complicated. Not only does the user provide subroutines to evaluate the coefficients of the finite difference equations, but the user must define each boundary by a sequence of grid points supplied on cards.

5.9 GENEPI

This software, written by Roux et al [25], is designed to solve a general

linear or non-linear elliptic or parabolic equation on a two-dimensional rectangle with Dirichlet, Neumann, or mixed boundary conditions. This program accepts the problem definition in symbolic form and generates a FORTRAN program that solves the approximate equations.

The user may specify a uniform or non-uniform grid in space and time. The program generates the finite difference approximation using the four nearest neighbors and solves the equations using point or line relaxation or ADI, but leaves the acceleration parameter selection to the user.

This program requires disk files, so portability may be a problem.

5.10 ELIPTI

This software, written by Taylor and Taylor [31], is designed to solve a general non-linear elliptic equation defined on an arbitrary region in the plane, and assuming Dirichlet boundary conditions. The solution is obtained by approximating the steady-state solution of a related time-dependent parabolic equation.

The approximation is by finite differences on a uniform rectangular grid. The time integration is performed in such a way that the scheme is equivalent to an AD1 scheme. Aitken-Shanks acceleration is used in time also.

6. Acknowledgements.

The author would like to express his appreciation to Professor Gene H. Golub of Stanford University for his hospitality at Serra House, a singular environment for research in numerical analysis, and to the faculty, staff, and graduate students at Serra House for their friendship during the author's tenure there.

REFERENCES.

[1] Adams, J. Swarztrauber, P., and Sweet, R., "FISHPAK: Efficient FORTRAN Subprograms for the Solution of Separable Elliptic Partial Differential Equations, Version 3", National Center for Atmospheric Research, Box 3000, Boulder, Colorado 80307, (1978).

[2] De Boor, C., "Package for Computing with B-Splines", J. SIAM Numr. Anal. 14 (1977), 441-472.

3] Buzbee, B., Golub, G.H., and Nielson, C., "On Direct Methods for Solving Poissons' Equation', J. SIAM Numer. Anal. 7 (1970), 627-656.

[4] Buzbee, B., et al, Private Communication.

[5] Cardenas, A., and Karplus, W., "PDEL - A Language for Partial Differential Equations", Comm. ACM 13 (1970), 184-191.

[6] Carver, M., et al, "The FORSIM VI Simulation Package for the Automated Solution of Arbitrarily Defined Partial Differential and/or Ordinary Differential Equation Systems", Report AECL-5821, Chalk River Nuclear Laboratories, Ontario, Canada (1978).

[7] Curtis, A., and Reid, J., "FORTRAN routines for the Solution of Sparse Sets of Linear Equations', Report AERE-R-6844, Harwell , England (1971).

[8] Eason, E., and Mote, Jr., C., "Solution of Nonlinear Differential Equations by Discrete Least Squares", Int. J. for Numerical Methods in Engineering 12(1978), 597-m.

[9] Engquist, B., and Smedsaas, T., 'DCG - A System for Automatic Code Generation for Hyperbolic Problems", Dept. of Computer Science, University of Uppsala, Sweden (1975).

[10] Gary, J., and Helgason, R., "An Extension of FORTRAN containing Finite Difference Operators", Software - Practice and Experience 2 (1972) 321-336.

[11] Gary, J., "HYPACK: A Subroutine for a Hyperbolic System Coupled with a Single Elliptic Equation," CS-142-78 Report, Dept. of Computer Science, University of Colorado, Boulder, Colorado, (1978).

[12] Hindmarsh, A., 'GEAR :Ordinary Differential Equation System Solver', Report UCID-30001, Lawrence Livermore Laboratory (1972).

[13] _____, "Preliminary Documentation of GEARIB: Solution of Implicit System of Ordinary Differential Equations with Banded Jacobian", Report UCID-30130, Lawrence Livermore Laboratory (1976).

[14] _____, 'GEARB: Solution of Ordinary Differential Equations having Banded Jacobian", Report UCID-30059, Lawrence Livermore Laboratory

[15] Hornsby, J., " EPDE1 - A Computer Programme for the Solution of Elliptic Partial Differential Equations (Potential Problems)", CERN (Geneva) Computer Centre Program Library Long Write-Up D-300 (1977).

[16] Houstis, E., and Rice, J., "Software for Linear Elliptic Problems On General Two Dimensional Domains", Advances in Computer Methods for Partial Differential Equations - II, R. Vichnevetsky (ed.), IMACS (AICA), Rutgers University, (1977), 7-12.

[17] Hyman, M., "The Method of Lines Solution of Partial Differential Equations", Report COO-3077-139, Courant Institute of Mathematical Sciences (1976).

[18] Kincaid, D., and Grimes, R., "Numerical Studies of Several Adaptive Iterative Algorithms", Report 126, Center for Numerical Analysis, University of Texas, Austin, Texas (1977).

[19] Leaf, G., et al, "DISPL : A Software Package for One and Two Spatially Dimensional Kinetics-Diffusion Problems", Report ANL-77-12, Argonne National Laboratory (1977).

[20] Madsen, N., and Sincovec, R., "General Software for Partial Differential Equations", Numerical Methods for Differntial Equations, Academic Press (1976) 229-242,.

[21] Melgaard, D., and Sincovec, R., "General Software for Two Dimensional Partial Differential Equations", Report CS76-21, Dept. of Computer Science, Kansas State University (1976).

[22] "PDEPACK: Partial Differential Equations Package Users Guide", Scientific Computing Consulting Services, P.O. Box 335, Manhattan, Kansas 66502 (1975).

[23] Proskurowski, W., "Four FORTRAN Programs for Numerically Solving Helmholtz's Equation in an Arbitrary Bounded Planar Region", Report LBL-7516, Lawrence Berkeley Laboratory (1978).

[24] Rice, J., "ELLPACK: A Research Tool for Elliptic Partial Differential Equation Software", Mathematical Software III, Academic Press (1977), 319-341.

[25] Roux, J., Tourneur, J., and Vandorpe, D., "Automatic Generation of Partial Differential Equations Integration Programs", Advances in Computer Methods for Partial Differential Equations - II, R. Vichnevetsky (ed.), IMACS (AICA), Rutgers University (1977).

[26] Ryder, B., and Sanderson, J., "A Program for Computing the Eigenvalue of Laplace's Equation", Computing Science Technical Report # 20, Bell Laboratories (1974).

[27] Schiesser, W., "A Digital Simulation System for Mixed Ordinary/Partial Differential Equation Modes", Proc. Sym. on Digital Simulation of Continuous Processors, Gyor, Hungary, 2 , (1971) 52-1 to 52-9.

[28] Schiesser, W., "DSS/2 - An Introduction to the Numerical Method of Lines Integration of Partial Differential Equations" 2 vols., Lehigh University (1976).

[29] Schryer, N., "Numerical Solution of Time-Varying Partial Differential Equations in One Space Variable", Computing Science Technical Report #53, Bell Laboratories (1977).

[30] Steuerwalt, M., "Certification Report on "Efficient FORTRAN Subprograms for the Solution of Elliptic Partial Differential Equations" by Paul Swarztrauber and Roland Sweet", LA-7524-MS Informal Report, Los Alamos Scientific Laboratory, Los Alamos, N.M., (1978).

[31] Taylor, J., and Taylor, J., "ELIPTI-TORMAC: A Code for the Solution of General Nonlinear Elliptic Problems over 2-D Regions of Arbitrary Shape", Advances in Computer Methods for Partial Differential Equations - II, R. Vichnevetsky (ed.) IMACS (AICA), Rutgers University (1977).

[32] Thomas, C., "POTENT - A Package for the Numerical Solution of Potential Problems in General Two-Dimensional Regions", Software for Numerical Mathematics, D. Evans (ed.), Academic Press (1974), 315-336.