TWO MODIFICATIONS TO THE SOFTWARE INTERFACE PACKAGE FOR NONLINEAR PARTIAL DIFFERENTIAL EQUATIONS

by

RICHARD LEE MORSE

B.S., Kansas State University, 1971

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1974

Approved by:

Major Professor

TABLE OF CONTENTS

TABLE OF FIGURES

Chapter 1

INTRODUCTION

PDEONE is a software interface for nonlinear partial differential equations jointly developed by Dr. Richard F. Sincovec, Kansas State University and Dr. Niel K. Madsen, Lawrence Livermore Laboratory (1). It is a piece of computer software which can serve as an interface which will allow many of the recent significant developments in the field of ODE's to be applied directly to the numerical solution of PDE's.

The method being implemented by this software package is the so-called numerical method of lines (2). Roughly speaking, the method of lines can be described as follows; if one has a time dependent PDE and discretizes the spatial variables, an approximating system of ordinary differential equations results. To solve the resulting equations one uses ODE methods and obtains numerical approximations of the original PDE.

The software package is designed with user convenience as a goal. To use this package the user simply defines his system of PDE's and supplies a spatial mesh to be used for the discretization of the problem in PDEONE. Then an ODE integrator with its built-in error and stability controls may be used (3, 4, 5).

This report will discuss two recent changes to the PDE interface package developed by Sincovec and Madsen (1). The first change being

the modification of the routine PDEONE to handle systems of PDE's that are coupled in the time derivative terms. Next is the addition of the routine PDEJAC to efficiently generate the Jacobian matrix needed when stiff methods are used to solve ordinary differential equations.

Chapter 2

MODIFICATION OF PDEONE AND ADDITION OF USER SUPPLIED AMATRX ROUTINE TO HANDLE COUPLED SYSTEMS OF NONLINEAR PARTIAL DIFFERENTIAL EQUATIONS

2.1. Definition of Problem

Partial differential equations may have an unlimited number of mathematical structures. Following is the structure chosen by Dr. Richard Sincovec and Dr. Niel Madsen to solve a wide class of realistic problems (1):

Let NPDE denote the number of PDE's on the interval $[a, b]$ and let

$$(2.1.1)\qquad \sum_{j=1}^{NPDE} a_{k,j} \frac{\partial u_j}{\partial t} = f_k\left(t,x,u_1,u_2,\ldots,u_{NPDE}, \frac{\partial u_1}{\partial x}, \frac{\partial u_2}{\partial x},\ldots, \frac{\partial u_{NPDE}}{\partial x}, \frac{1}{x^c}\frac{\partial}{\partial x}\left(x^c D_{k,1}\frac{\partial u_1}{\partial x}\right), \frac{1}{x^c}\frac{\partial}{\partial x}\left(x^c D_{k,2}\frac{\partial u_2}{\partial x}\right),\ldots, \frac{1}{x^c}\frac{\partial}{\partial x}\left(x^c D_{k,NPDE}\frac{\partial u_{NPDE}}{\partial x}\right)\right),$$

$$a < x < b, \quad t > t_o, \quad k = 1,2,\ldots,NPDE,$$

denote the coupled systems of PDE's with boundary conditions,

$$(2.1.2)\qquad \alpha_k u_k + \beta_k \frac{\partial u_k}{\partial x} = \gamma_k \quad \text{at } x = a \text{ and } b, \quad t > t_o, \quad k = 1,2,\ldots,NPDE,$$

and initial conditions,

(2.1.3) $u_k(t_o,x) = \phi_k(x)$, $a \leq x \geq b$, $k = 1,2,\ldots,NPDE$.

If $\beta_k \neq 0$ then α_k, β_k and γ_k may be functions of t, x, and $\vec{u} \equiv (u_1,u_2,\ldots,u_{NPDE})$. but only functions of x and t otherwise; $D_{k,j}$ and $a_{k,j}$ (k,j = 1,2,...,NPDE) are functions of x, t, and $\vec{u}$; and c is 0, 1, or 2 depending on whether the problem is in Cartesian, cylindrical, or spherical coordinates, respectively.

By assuming that all the coefficient functions, α_k, β_k, γ_k, $D_{k,j}$, $a_{k,j}$, f_k, and ϕ_k, are at least piecewise continuous functions of all their respective variables; problems with physical discontinuities can be defined using the software interface.

Boundary conditions for PDE's are often classified into three types: Dirichlet ($\beta_k = 0$), Neumann ($\alpha_k = 0$), or mixed ($\alpha_k \neq 0$, $\beta_k \neq 0$). The boundary condition may change with respect to time, as well as from equation to equation. Also the initial condition is not required to satisfy the boundary conditions as x approaches either a or b.

The $(a_{k,j})$ (k,j = 1,2,...,NPDE) matrix allows for a coupling of the time derivative of systems of parabolic PDE's and/or hyperbolic PDE's. The coupling may be nonlinear as each $a_{k,j}$ may be a function of $\vec{u}$.

Note that problem (2.1.1) - (2.1.3) is completely defined if one specifies the interval, $[a, b]$; the initial time, t_o; the vector functions $f = (f_k)$, $\alpha = (\alpha_k)$, $\beta = (\beta_k)$, $\gamma = (\gamma_k)$ (k = 1,2,...,NPDE); the matrix functions $D = (D_{k,j})$, $A = (a_{k,j})$ (k,j = 1,2,...,NPDE); and the initial conditions $\phi_k(x)$, k = 1,2,...,NPDE. With the software interface

developed here, the user will be required to write four basic subprograms to define: the matrix D, the matrix A, the vector f, and the boundary conditions (i.e., the vectors α, β, and γ); plus a main program to specify the spatial mesh and initial conditions, to set a flag to signal the presence of an A matrix, to call the integrator, and to print the results.

To prevent PDEONE from continually decomposing the A matrix and solving for the $A^{-1} f_k$, k = 1,2,...,NPDE on the right-hand side the user has the option of setting the value of MATRIX either to 0 or 1. If MATRIX = 1, the routine will perform a LU decomposition of the A matrix and solve for the right-hand side where $\frac{\partial u_k}{\partial t} = A^{-1} f_k$, k = 1,2,...,NPDE. If MATRIX = 0, PDEONE will solve $\frac{\partial u_k}{\partial t} = f_k$, k = 1,2,...,NPDE and there will be no coupling of the PDE systems. The user will not have to write the identity matrix for A and continually have it decomposed.

2.2. Difference Approximations

Once the user has defined an appropriate PDE system and the corresponding spatial mesh, the software interface uses consistent centered difference approximations to convert the PDE system into an approximating initial value system of ordinary differential equations (1).

2.3. Use of the PDE Interface Package

To use the PDE interface the user should have both a stiff and nonstiff integrator. Unless a problem is known a priori to be stiff, the problem should be used with a nonstiff integrator first. Because the integrators developed by Gear (3) and Hindmarsh (4, 5) have both stiff and nonstiff methods built into the same program they should be

considered in implementing the PDE interface. Refer to Madsen and Sincovec (2) for a more complete discussion of the use of ODE integrators for solving PDE's.

To use the PDE interface (given in appendix A) the user is required to write four subroutines defining his system of PDE's in order for the interface to discretize the PDE's and convert them into a system of approximating ODE's. These subroutines are: D(T, X, U, DVAL, NPDE) defining the $D_{k,j}$ (k,j = 1,2,...,NPDE) of equation (2.1.1), F(T, X, U, UX, DUXX, FVAL, NPDE) defining the right-hand side of (2.1.1), BNDRY (T, X, U, ALPHA, BETA, GAMMA, NPDE) defining the boundary conditions (2.1.2), and AMATRX(NPDE, T, X, U) defining the coupling of the PDE's. Refer to Sincovec and Madsen (1) for specific details on subroutines D, F, and BNDRY. Specific details for subroutine AMATRX will be presented in this paper.

In subroutine AMATRX, T and X are scalar quantities representing respectively the current time and spatial variable, while U is a vector quantity with entries U_k (k = 1,2,...,NPDE). Approximate values of the preceding variables are passed to the user's routines by the PDE interface. The user must be careful to neither divide by zero nor perform other noncomputable operations. The user is cautioned to make sure the A matrix is neither singular nor apt to become singular at any point during the integration.

a. Subroutine AMATRX is constructed as follows:

```
SUBROUTINE AMATRX(NPDE, T, X, U)
COMMON/AMAT/ A(NPDE, NPDE), IP(NPDE), MATRIX
DIMENSION U(NPDE)
```

Here A(k,j) (k,j = 1,2,...,NPDE) are defined. These are the values of the coupling equations $a_{k,j}$, appearing in (2.1.1). All the matrix entries A(k,j) must be defined by the user (even if zero) unless the value of MATRIX = 0 is set in the main calling program, signifying no A matrix is needed. Should MATRIX = 0 the values of A(k,j) need not be defined.

```
RETURN
END
```

The COMMON/AMAT/ card must be replaced entirely while the actual value of NPDE (number of partial differential equations) needs to replace NPDE at each occurrence.

At the present time, the user must include the following card in the main program, PDEONE, PDEJAC, and AMATRX.

```
COMMON/AMAT/A(NPDE, NPDE),
    IP(NPDE), MATRIX
```

The remainder of this paper will assume the usage of GEARB (4) with the driver, DRIVEB, and its modifications as used by Sincovec and Madsen (1). GEARB will be further modified to calculate the block tridiagonal Jacobian matrix by calling the subroutine PDEJAC once as opposed to the usual method of repeated calls to PDEONE. Refer to Chapter 3 of this paper for a discussion of the change.

2.4. Numerical tests for AMATRX

If AMATRX is a linear combination of systems of equations (i.e., constants, T, and X are the only quantities allowed in the values of $(a_{k,j})$ $(k,j = 1,2,\ldots,NPDE)$ it is relatively simple to show that AMATRX is working. One combines the FVAL terms of subroutine F in the same order as AMATRX. The PDE's solved are $\frac{\partial u}{\partial t} = A^{-1}Af = f$. This holds true for all the numerical examples given by Sincovec and Madsen (1).

Given the following problem:

$$a_{1,1}\frac{\partial u_1}{\partial t} + a_{1,2}\frac{\partial u_2}{\partial t} = \frac{1}{x}\frac{\partial}{\partial x}\left(x\frac{\partial u_1}{\partial x}\right)$$

$$a_{2,1}\frac{\partial u_1}{\partial t} + a_{2,2}\frac{\partial u_2}{\partial t} = 0.$$

This problem was solved with

$$a_{1,1} = u_2 \qquad a_{1,2} = 0.$$

$$a_{2,1} = 0. \qquad a_{2,2} = 1.$$

with boundary conditions

$$\frac{\partial u_1}{\partial x}(t, 0) = 0.$$

$$\frac{\partial u_1}{\partial x}(t, 1) = 1.73E\text{-}9(6.25E+10-\left[u(t, 1)\right]^4)$$

$$u_2(t, 0) = u_2(t, 1) = 2.$$

and initial conditions given by

$$u_1(0, x) = 600. \qquad u_2(0, x) = 2.$$

Chapter 3

SUBROUTINE PDEJAC TO EFFICIENTLY GENERATE THE JACOBIAN MATRIX NEEDED FOR STIFF ODE METHODS

3.1. Definition of Problem

An ODE system is said to be stiff if it involves both very rapidly changing and very slowly changing terms all of a decaying nature. When the problems are stiff they must be solved implicitly with the following system of nonlinear equations being solved at each step (3, 5).

Most of the methods used in solving stiff problems are based on the linear multistep formula.

$$(3.1.1)\quad y_n = \sum_{j=1}^{K_1} \alpha_j\, y_{n-j} + h \sum_{j=0}^{K_2} \beta_j\, \dot{y}_{n-j}$$

For Gear's method of order q $K_1 = q$ and $K_2 = 0$, α_j and β_j are constants associated with the method, h is the stepsize, y_k is an approximation to $y(t_k)$, and $\dot{y}_k$ is an approximation to $\dot{y}(t_k) = f(y_k, t_k)$. Equation (3.1.1) can be written in the following form.

$$(3.1.2)\quad g(y_n) \equiv y_n - h\beta_o f(y_n, t_n) - \sum_{1}^{K_1} \alpha_j\, y_{n-j} - h \sum_{1}^{K_2} \beta_j\, y_{n-j} = 0$$

One can then solve the nonlinear system $g(y_n)$ by Newton's method.

$$(3.1.3)\quad y_{n(m+1)} = y_{n(m)} - P^{-1}_{n(m)}\, g(y_{n(m)}),$$

$$P_{n(m)} = \frac{\partial g}{\partial y}\bigg|_{y_{n(m)}} = I - h\beta_0 \frac{\partial f}{\partial y}\bigg|_{y_{n(m)}}$$

In practice, it is much less costly to replace the matrix $P_{n(m)}$ by

$$(3.1.4)\quad P_n = I - h\beta_0 J_n \ , \ J_n = \left.\frac{\partial f}{\partial y}\right|_{y_{n(0)}}$$

at a slight loss of the rate of convergence. One can also use a "chord method" by utilizing P_n for some $n' \leq n$ instead of P_n at step n.

ODE integrators approximate the Jacobian matrix, J_n, by successive calls to a routine that computes DY/DT for a given T and Y. This is accomplished by Sincovec and Madsen (1) by NPDE*NPTS calls to PDEONE with each call calculating NPDE*NPTS values. Since the Jacobian matrix of equation (3.1.4) is block tridiagonal for partial differential equations of the structure (2.1.1), one need only calculate (3*NPTS)-2 blocks of NPDE*NPDE values of the Jacobian with NPDE being the number of partial differential equations and NPTS being the number of spatial grid points (Figure 1).

Because h and β_0 are known, one can calculate the values for $-h\beta_0 J_n$ in one subroutine call, thus reducing the number of times the program will set up linkage to the subroutine.

Subroutine PDEJAC is written to calculate $-h\beta_0 J_n$ in one call. It calculates only the partial derivative needed for the block tridiagonal matrix of Figure 1.

3.2. Use of PDEJAC with Stiff ODE Integrators

PDEJAC is an interface subroutine which generates the block tridiagonal matrix used in solving partial differential equations. It generates values for $-h\beta_0 J_n$, $J_n = \left.\frac{\partial f}{\partial y}\right|_{y_{n(0)}}$ and stores them in a vector

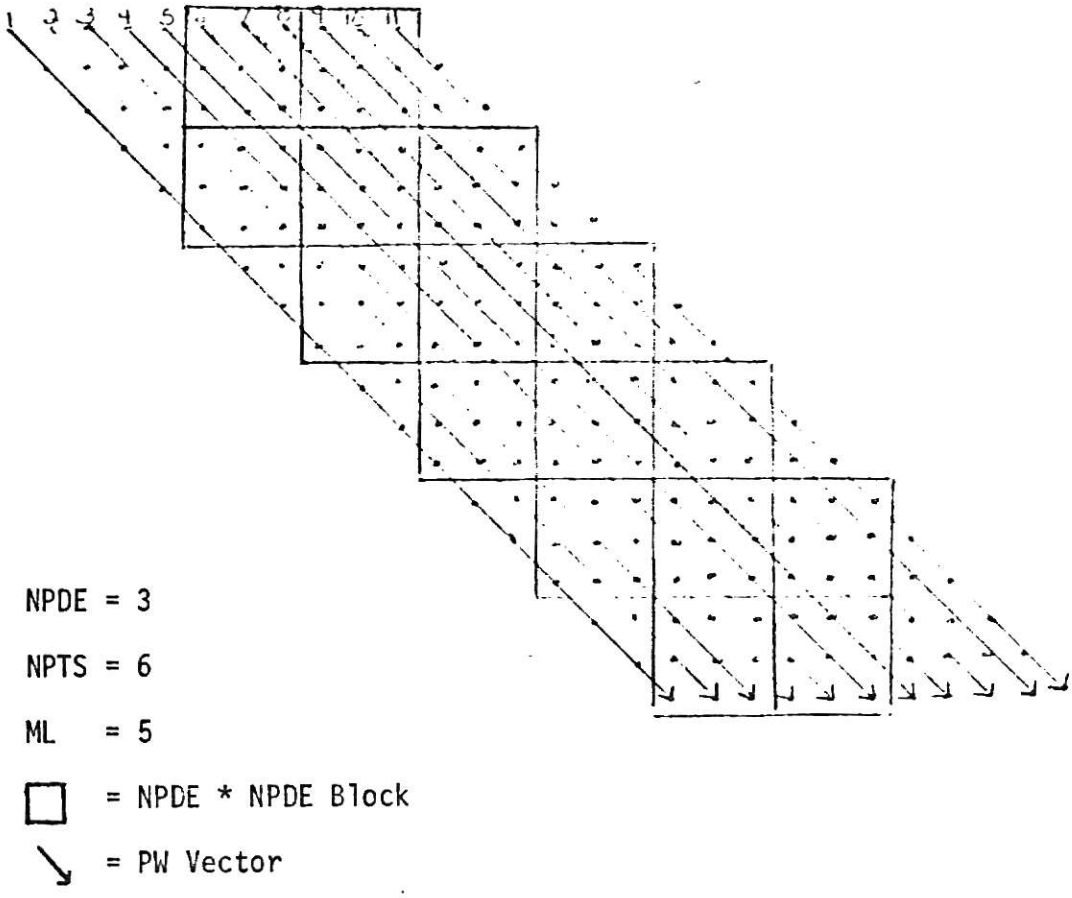

Figure 1: Block Tridiagonal Jacobian Matrix for PDE's and Its Storage in the PW Vector

called PW. This routine will not calculate any partials outside the block tridiagonal form of Figure 1.

The PW vector is stored from the lowermost band upward (Figure 1). PDEJAC uses the four subroutines referred to in Chapter 2 (D, F, BNDRY, and AMATRX) while calculating and storing the $-h\beta_0 J_n$ matrix in the PW vector.

Since the Jacobian matrix is calculated in one call, the user must be certain to pass the appropriate values to subroutine PDEJAC from the stiff ODE integrator. The call for PDEJAC is of the form:

PDEJAC(N, NO, NPDE, NPTS, ML, T, U, UDOT, EL, FSAVE, YMAX, PW, H)

with the arguments described in Appendix B.

3.3. Numerical Tests Using PDEJAC

This writer has found if one used PDEJAC with the stiff ODE integrator, GEARB modified slightly to call PDEJAC only once, one can use the PDE interface developed by Sincovec and Madsen (1) much more efficiently.

Using the examples given by Sincovec and Madsen (1), this writer has realized a forty percent decrease in computer time to solve their example E (Cylindrical Problem) with twenty-one grid points. Five of the remaining six problems were solved with even more significant savings as the number of grid points increased.

To verify the correctness of PDEJAC, the user need only check to see if the answers using PDEJAC are identical to the answers not using PDEJAC. The user may also see if the PW vectors are the same in both cases.

Chapter 4

CONCLUSION

The author of this paper believes that the PDE interface presented by Sincovec and Madsen (1) is a significant beginning for robust software for reasonably broad classes of partial differential equations. With the additions and modifications presented in this paper, the PDE interface package will more efficiently solve more general types of equations.

With the addition of AMATRX the user may now solve systems of coupled parabolic and hyperbolic equations. If the user does not have a system of coupled equations, essentially no efficiency will be lost in the original PDE interface (1) when MATRIX is set to zero.

One of the largest areas of concern with the PDE interface is the generation of the Jacobian matrix for stiff equations. Most ODE integrators are capable of automatically generating the Jacobian matrix of equation 3.1.4. It is a convenient but frequently expensive feature if either NPDE*NPTS or the number of Jacobian generations are large.

Since a stiff integrator is absolutely essential for Sincovec and Madsen's approach to solving PDE's to be robust, the author feels his subroutine PDEJAC may create a significant change in the computer time used in solving partial differential equations. By using PDEJAC the user can save forty percent of his computer time and have the ODE integrator generate the Jacobian matrix needed for stiff equations.

REFERENCES

1. Sincovec, R. F. and N. K. Madsen, "Software for Nonlinear Partial Differential Equations", UCRL-75658, Lawrence Livermore Laboratory, May 1974.

2. Madsen, N. K. and R. F. Sincovec, "The Numerical Method of Lines for the Solution of Nonlinear Partial Differential Equations", UCRL-75142, Lawrence Livermore Laboratory, September 1973; also to appear in S.I.A.M. Journal on Numerical Analysis.

3. Gear, C. W., Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs, N. J. 1971.

4. Hindmarsh, A. C., "GEARB: Solution of Ordinary Differential Equations Having Banded Jacobian", UCID-30059, Lawrence Livermore Laboratory, May 1973.

5. Hindmarsh, A. C., "GEAR: Ordinary Differential Equation System Solver", UCID-30001 Rev. 2, Lawrence Livermore Laboratory, August 1972.

APPENDIX A

```
      SUBROUTINE PDEONE(NPDE,NPTS,T,U,UDOT)                              PDE00010
      IMPLICIT REAL*8(A-H,O-Z)                                           PDE00020
      DIMENSION U(NPDE,NPTS),UDOT(NPDE,NPTS)                             PDE00030
C                                                                        PDE00040
C PDEONE IS AN INTERFACE SUBROUTINE WHICH CONVERTS COUPLED ONE           PDE00050
C DIMENSIONAL SYSTEMS OF PARTIAL DIFFERENTIAL EQUATIONS                  PDE00060
C INTO A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS, UDOT                 PDE00070
C = F(T,X,U), BY CENTERED DIFFERENCE APPROXIMATIONS.                     PDE00080
C THIS ROUTINE IS INTENDED TO BE USED WITH A ROBUST                      PDE00090
C ODE INTEGRATOR.                                                        PDE00100
C                                                                        PDE00110
C INPUT..                                                                PDE00120
C  NPDE = NUMBER OF PARTIAL DIFFERENTIAL EQUATIONS.                      PDE00130
C  NPTS = NUMBER OF SPATIAL GRID POINTS.                                 PDE00140
C     T = CURRENT VALUE OF TIME.                                         PDE00150
C     U = AN NPDE BY NPTS ARRAY CONTAINING THE COMPUTED                  PDE00160
C         SOLUTION AT TIME T.                                            PDE00170
C                                                                        PDE00180
C OUTPUT..                                                               PDE00190
C  UDOT = AN NPDE BY NPTS ARRAY CONTAINING THE RIGHT HAND                PDE00200
C         SIDE OF THE RESULTING SYSTEM OF ODE'S, F(T,X,U)                PDE00210
C         OBTAINED BY DISCRETIZING THE GIVEN PDE'S.                      PDE00220
C                                                                        PDE00230
C THE USER MUST INSERT A DIMENSION STATEMENT AND A COMMON                PDE00240
C STATEMENT OF THE FOLLOWING FORMS#                                      PDE00250
C                                                                        PDE00260
C     DIMENSION DVAL(**,**,2),UX(**),UAVG(**),ALPHA(**),                 PDE00270
C    *          BETA(**),GAMMA(**)                                       PDE00280
C                                                                        PDE00290
C     COMMON /AMAT/ A(**,**),IP(**),MATRIX                               PDE00300
C                                                                        PDE00310
C THE SYMBOLS ** DENOTE THE ACTUAL NUMERICAL VALUE OF NPDE               PDE00320
C FOR THE PROBLEM BEING SOLVED.                                          PDE00330
C                                                                        PDE00340
C COMMON BLOCK AMAT CONTAINS THE (A) MATRIX, THE TRANSFORMATION          PDE00350
C VECTOR USED IN DECOMPOSING THE MATRIX, AND THE FLAG                    PDE00360
C SIGNIFYING THE MATRIX IS PRESENT.                                      PDE00370
C                                                                        PDE00380
C COMMON BLOCK COORD CONTAINS 0,1, OR 2 DEPENDING ON WHETHER             PDE00390
C THE PROBLEM IS IN CARTESIAN,CYLINDRICAL, OR SPHERICAL                  PDE00400
C COORDINATES, RESPECTIVELY.                                             PDE00410
C                                                                        PDE00420
C COMMON BLOCK MESH CONTAINS THE USER SPECIFIED SPATIAL                  PDE00430
C GRID POINTS.                                                           PDE00440
C                                                                        PDE00450
C                                                                        PDE00460
      COMMON /MESH/ X(1)                                                 PDE00470
      COMMON /COORD/ ICORD                                               PDE00480
      ICORD1 = ICORD + 1                                                 PDE00490
C                                                                        PDE00500
C UPDATE BOUNDARY VALUES AT THE LEFT BOUNDARY                            PDE00510
C                                                                        PDE00520
      CALL BNDRY(T,X(1),U,ALPHA,BETA,GAMMA,NPDE)                         PDE00530
      ITEST = 0                                                          PDE00540
```

```
      DO 10 K=1,NPDE                                                    PDE00550
        IF (BETA(K).NE.0.0)  GO TO 10                                   PDE00560
        U(K,1) = GAMMA(K)/ALPHA(K)                                      PDE00570
        ITEST = ITEST + 1                                               PDE00580
   10 CONTINUE                                                          PDE00590
      DXI = 1./(X(2)-X(1))                                              PDE00600
      IF (ITEST.EQ.0) GO TO 20                                          PDE00610
      IF (ITEST.EQ.NPDE)  GO TO 45                                      PDE00620
      CALL BNDRY(T,X(1),U,ALPHA,BETA,GAMMA,NPDE)                        PDE00630
C                                                                       PDE00640
C EVALUATE 'DIFFUSION COEFFICIENTS', D, AT THE LEFT BOUNDARY            PDE00650
C                                                                       PDE00660
   20 CALL D(T,X(1),U,DVAL,NPDE)                                        PDE00670
C                                                                       PDE00680
C FORM APPROXIMATION TO DU/DX AT THE LEFT BOUNDARY                      PDE00690
C                                                                       PDE00700
      DO 40 K=1,NPDE                                                    PDE00710
        IF(BETA(K).NE.0.0)  GO TO 30                                    PDE00720
        UX(K) = DXI *(U(K,2) - U(K,1))                                  PDE00730
        GO TO 40                                                        PDE00740
   30   UX(K) = (GAMMA(K) - ALPHA(K)*U(K,1))/BETA(K)                    PDE00750
   40 CONTINUE                                                          PDE00760
C                                                                       PDE00770
C EVALUATE U-AVERAGE IN THE FIRST INTERVAL                              PDE00780
C                                                                       PDE00790
   45 DO 50 K=1,NPDE                                                    PDE00800
        UAVG(K) = .5*(U(K,2) + U(K,1))                                  PDE00810
   50 CONTINUE                                                          PDE00820
C                                                                       PDE00830
C EVALUATE 'DIFFUSION COEFFICIENTS', D, IN THE FIRST INTERVAL           PDE00840
C                                                                       PDE00850
      XAVGR = .5 * ( X(2) + X(1) )                                      PDE00860
      CALL D(T,XAVGR,UAVG,DVAL(1,1,2),NPDE)                             PDE00870
      DXIR = DXI                                                        PDE00880
      DXIL = 1.                                                         PDE00890
      IF (ICORD.EQ.0) GO TO 55                                          PDE00900
      DXIL = X(1)**ICORD                                                PDE00910
      CXIR = XAVGR**ICORD * DXI                                         PDE00920
   55 DXIC = FLOAT(ICORD1) / (XAVGR**ICORD1 - X(1)**ICORD1)             PDE00930
C                                                                       PDE00940
C EVALUATE DUXX AT THE LEFT BOUNDARY                                    PDE00950
C                                                                       PDE00960
      DO 60 K=1,NPDE                                                    PDE00970
        DO 60 L=1,NPDE                                                  PDE00980
          DVAL(K,L,1)=DXIC*(DVAL(K,L,2)*(U(L,2)-U(L,1))*DXIR-           PDE00990
     *        DVAL(K,L,1)*UX(L)*DXIL)                                   PDE01000
   60 CONTINUE                                                          PDE01010
      IF (ITEST.EQ.NPDE) GO TO 65                                       PDE01020
C                                                                       PDE01030
C EVALUATE RIGHT HAND SIDE OF PDE'S AT THE LEFT BOUNDARY                PDE01040
C                                                                       PDE01050
      CALL F(T,X(1),U,UX,DVAL,UDOT,NPDE)                                PDE01060
C                                                                       PDE01070
C DECOMPOSE THE (A) MATRIX AND SOLVE FOR THE RIGHT HAND                 PDE01080
C SIDE IF NECESSARY.                                                    PDE01090
C                                                                       PDE01100
      IF (MATRIX.EQ.0) GO TO 65                                         PDE01110
```

```
      CALL AMATRX(NPDE,T,X(1),U)                                        PDE01120
      CALL DECOMP(NPDE,NPDE,A,IP)                                       PDE01130
      CALL SOLVE(NPDE,NPDE,A,UDOT(1,1),IP)                              PDE01140
C                                                                       PDE01150
C SET UDOT = 0 FOR KNOWN LEFT BOUNDARY VALUES                           PDE01160
C                                                                       PDE01170
   65 DO 70 K=1,NPDE                                                    PDE01180
        IF (BETA(K).EQ.0.0) UDOT(K,1)=0.0                               PDE01190
   70 CONTINUE                                                          PDE01200
C                                                                       PDE01210
C UPDATE BOUNDARY VALUES AT THE RIGHT BOUNDARY                          PDE01220
C                                                                       PDE01230
      CALL BNDRY(T,X(NPTS),U(1,NPTS),ALPHA,BETA,GAMMA,NPDE)             PDE01240
      ITEST = 0                                                         PDE01250
      DO 75 K=1,NPDE                                                    PDE01260
        IF(BETA(K).NE.0.0)  GO TO 75                                    PDE01270
        U(K,NPTS) = GAMMA(K)/ALPHA(K)                                   PDE01280
        ITEST = ITEST + 1                                               PDE01290
   75 CONTINUE                                                          PDE01300
C                                                                       PDE01310
C MAIN LOOP TO FORM ODE'S AT THE INTERIOR GRID POINTS                   PDE01320
C                                                                       PDE01330
      IBCK = 1                                                          PDE01340
      IFWD = 2                                                          PDE01350
      ILIM = NPTS-1                                                     PDE01360
      DO 100 I=2,ILIM                                                   PDE01370
        ITEMP = IBCK                                                    PDE01380
        IBCK = IFWD                                                     PDE01390
        IFWD = ITEMP                                                    PDE01400
        XAVGL = XAVGR                                                   PDE01410
        XAVGR = .5 * (X(I+1) + X(I))                                    PDE01420
        DXI = 1./(X(I+1)-X(I-1))                                        PDE01430
        DXIL = DXIR                                                     PDE01440
        DXIR = 1.                                                       PDE01450
        IF (ICORD.NE.0) DXIR =XAVGR**ICORD                              PDE01460
        DXIR = DXIR / (X(I+1)-X(I))                                     PDE01470
        DXIC = FLOAT(ICORD1) / (XAVGR**ICORD1 - XAVGL**ICORD1)          PDE01480
C                                                                       PDE01490
C EVALUATE DU/DX AND U-AVERAGE AT THE I-TH GRID POINT                   PDE01500
C                                                                       PDE01510
        DO 80 K=1,NPDE                                                  PDE01520
          UX(K) = DXI*(U(K,I+1) - U(K,I-1))                             PDE01530
          UAVG(K) = .5*(U(K,I+1) + U(K,I))                              PDE01540
   80   CONTINUE                                                        PDE01550
C                                                                       PDE01560
C EVALUATE 'DIFFUSION COEFFICIENTS', D, IN THE I-TH INTERVAL            PDE01570
C                                                                       PDE01580
      CALL D(T,XAVGR,UAVG,DVAL(1,1,IFWD),NPDE)                          PDE01590
C                                                                       PDE01600
C EVALUATE DUXX AT THE I-TH GRID POINT                                  PDE01610
C                                                                       PDE01620
        DO 90 K=1,NPDE                                                  PDE01630
          DO 90 L=1,NPDE                                                PDE01640
            DVAL(K,L,IBCK)=DXIC*(DVAL(K,L,IFWD)*(U(L,I+1)-              PDE01650
     *           U(L,I))*DXIR - DVAL(K,L,IBCK)*(U(L,I)                  PDE01660
     *           -U(L,I-1))*DXIL)                                       PDE01670
   90   CONTINUE                                                        PDE01680
```

```
C                                                                       PDE01690
C  EVALUATE RIGHT HAND SIDE OF PDE'S AT THE I-TH GRID POINT             PDE01700
C                                                                       PDE01710
      CALL F(T,X(I),U(1,I),UX,DVAL(1,1,IBCK),UDOT(1,I),NPDE)            PDE01720
C                                                                       PDE01730
C  DECOMPOSE THE (A) MATRIX AND SOLVE FOR THE RIGHT HAND                PDE01740
C  SIDE IF NECESSARY.                                                   PDE01750
C                                                                       PDE01760
      IF (MATRIX.EQ.0) GO TO 100                                        PDE01770
      CALL AMATRX(NPDE,T,X(I),U(1,I))                                   PDE01780
      CALL DECOMP(NPDE,NPDE,A,IP)                                       PDE01790
      CALL SOLVE(NPDE,NPDE,A,UDOT(1,I),IP)                              PDE01800
  100 CONTINUE                                                          PDE01810
C                                                                       PDE01820
C  FINISH UPDATING THE RIGHT BOUNDARY IF NECESSARY                      PDE01830
C                                                                       PDE01840
      IF (ITEST.EQ.0) GO TO 120                                         PDE01850
      IF(ITEST.EQ.NPDE)   GO TO 155                                     PDE01860
      CALL BNDRY(T,X(NPTS),U(1,NPTS),ALPHA,BETA,GAMMA,NPDE)             PDE01870
C                                                                       PDE01880
C  EVALUATE 'DIFFUSION COEFFICIENTS', D, AT THE RIGHT BOUNDARY          PDE01890
C                                                                       PDE01900
  120 CALL D(T,X(NPTS),U(1,NPTS),DVAL(1,1,IBCK),NPDE)                   PDE01910
C                                                                       PDE01920
C  FORM APPROXIMATIONS TO DU/DX AT THE RIGHT BOUNDARY                   PDE01930
C                                                                       PDE01940
      DXI = 1. / (X(NPTS) - X(ILIM))                                    PDE01950
      DO 140 K=1,NPDE                                                   PDE01960
        IF(BETA(K).NE.0.0)  GO TO 130                                   PDE01970
        UX(K) = DXI *(U(K,NPTS) - U(K,ILIM))                            PDE01980
        GO TO 140                                                       PDE01990
  130   UX(K) = (GAMMA(K) - ALPHA(K)*U(K,NPTS))/BETA(K)                 PDE02000
  140 CONTINUE                                                          PDE02010
      DXIL = DXIR                                                       PDE02020
      DXIR = 1                                                          PDE02030
      IF (ICORD.NE.0) DXIR = X(NPTS)**ICORD                             PDE02040
      DXIC=FLOAT(ICORD1) /(X(NPTS)**ICORD1-XAVGR**ICORD1)               PDE02050
C                                                                       PDE02060
C  EVALUATE DUXX AT THE RIGHT BOUNDARY                                  PDE02070
C                                                                       PDE02080
      DO 150 K=1,NPDE                                                   PDE02090
        DO 150 L=1,NPDE                                                 PDE02100
          DVAL(K,L,IBCK)=DXIC*(DVAL(K,L,IBCK)*UX(L)*DXIR                PDE02110
     *    -DVAL(K,L,IFWD)*(U(L,NPTS)-U(L,ILIM))*DXIL)                   PDE02120
  150 CONTINUE                                                          PDE02130
C                                                                       PDE02140
C  EVALUATE RIGHT HAND SIDE OF PDE'S AT THE RIGHT BOUNDARY              PDE02150
C                                                                       PDE02160
      CALL F(T,X(NPTS),U(1,NPTS),UX,DVAL(1,1,IBCK),                     PDE02170
     *   UDOT(1,NPTS),NPDE)                                             PDE02180
C                                                                       PDE02190
C  DECOMPOSE THE (A) MATRIX AND SOLVE FOR THE RIGHT HAND                PDE02200
C  SIDE IF NECESSARY.                                                   PDE02210
C                                                                       PDE02220
      IF (MATRIX.EQ.0) GO TO 155                                        PDE02230
      CALL AMATRX(NPDE,T,X(NPTS),U(1,NPTS))                             PDE02240
      CALL DECOMP(NPDE,NPDE,A,IP)                                       PDE02250
```

```
      CALL SOLVE(NPDE,NPDE,A,UDOT(1,NPTS),IP)                    PDE02260
C                                                                PDE02270
C SET UDOT = 0 FOR KNOWN RIGHT BOUNDARY VALUES                   PDE02280
C                                                                PDE02290
  155 DO 160 K=1,NPDE                                            PDE02300
        IF (BETA(K).EQ.0.0) UDOT(K,NPTS)=0.                      PDE02310
  160  CONTINUE                                                  PDE02320
      RETURN                                                     PDE02330
      END                                                        PDE02340
```

APPENDIX B

```
      SUBROUTINE PDEJAC(N,NO,NPDE,NPTS,ML,T,U,UDOT,EL,FSAVE,          PDJ00010
     *                  YMAX,PW,H)                                    PDJ00020
      IMPLICIT REAL*8(A-H,O-Z)                                        PDJ00030
      DIMENSION U(NPDE,NPTS),UDOT(NPDE,NPTS),FSAVE(NO),               PDJ00040
     *          YMAX(NO),PW(1)                                        PDJ00050
C                                                                     PDJ00060
C PDEJAC IS AN INTERFACE SUBROUTINE WHICH GENERATES THE               PDJ00070
C BLOCK TRIDIAGONAL JACOBIAN MATRIX NEEDED FOR STIFF METHODS          PDJ00080
C WHEN SOLVING PARTIAL DIFFERENTIAL EQUATIONS.  THIS                  PDJ00090
C ROUTINE WAS WRITTEN TO BE COMBINED WITH STIFF INTEGRATORS           PDJ00100
C AND CALCULATE THE JACOBIAN MATRIX IN ONE CALL.                      PDJ00110
C THE VALUES OF -EL(1)*H*JACOBIAN NEEDED FOR STIFF METHODS ARE        PDJ00120
C CALCULATED IN THIS ROUTINE AND STORED IN PW IN THE                  PDJ00130
C APPROPRIATE PLACES.  PW IS STORED FROM THE LOWERMOST                PDJ00140
C BAND UPWARD WITH N LOCATIONS FOR EACH BAND.                         PDJ00150
C                                                                     PDJ00160
C INPUT..                                                             PDJ00170
C     N = NUMBER OF ODE'S, IS EQUAL TO NPDE*NPTS                      PDJ00180
C    NO = NUMBER OF DIFFERENTIAL EQUATIONS INITIALLY.                 PDJ00190
C  NPDE = NUMBER OF PARTIAL DIFFERENTIAL EQUATIONS.                   PDJ00200
C  NPTS = NUMBER OF SPATIAL GRID POINTS.                              PDJ00210
C    ML = BAND WIDTH OF THE LOWER BAND.                               PDJ00220
C     T = CURRENT VALUE OF TIME.                                      PDJ00230
C     U = AN NPDE BY NPTS ARRAY CONTAINING THE COMPUTED               PDJ00240
C         SOLUTION AT TIME T.                                         PDJ00250
C  UDOT = AN NPDE BY NPTS ARRAY CONTAINING THE RIGHT HAND             PDJ00260
C         SIDE OF THE RESULTING SYSTEM OF ODE'S, F(T,X,U)             PDJ00270
C         OBTAINED BY DISCRETIZING THE GIVEN PDE'S.                   PDJ00280
C    EL = METHOD COEFFICIENT.                                         PDJ00290
C FSAVE = VECTOR CONTAINING ORIGINAL VALUE OF THE DERIVATIVE.         PDJ00300
C  YMAX = VECTOR CONTAINING MAXIMUM VALUE OF EACH Y SEEN.             PDJ00310
C     H = THE STEP SIZE TO BE ATTEMPTED BY THE INTEGRATOR.            PDE00320
C                                                                     PDJ00330
C OUTPUT..                                                            PDJ00340
C    PW = A VECTOR CONTAINING THE PARTIAL DERIVATIVES FOR THE         PDJ00350
C         JACOBIAN MATRIX.                                            PDJ00360
C                                                                     PDJ00370
C THE USER MUST INSERT A DIMENSION STATEMENT AND A COMMON             PDJ00380
C STATEMENT OF THE FOLLOWING FORMS#                                   PDJ00390
C                                                                     PDJ00400
C     DIMENSION DVAL(**,**,2),UX(**),UAVG(**),ALPHA(**),              PDJ00410
C    *          BETA(**),GAMMA(**)                                    PDJ00420
C                                                                     PDJ00430
C     COMMON /AMAT/ A(**,**),IP(**),MATRIX                            PDJ00440
C                                                                     PDJ00450
C THE SYMBOLS ** DENOTE THE ACTUAL NUMERICAL VALUE OF NPDE            PDJ00460
C FOR THE PROBLEM BEING SOLVED.                                       PDJ00470
C                                                                     PDJ00480
C COMMON BLOCK AMAT CONTAINS THE (A) MATRIX, THE TRANSFORMATION       PDJ00490
C VECTOR USED IN DECOMPOSING THE MATRIX, AND THE FLAG                 PDJ00500
C SIGNIFYING THE MATRIX IS PRESENT.                                   PDJ00510
C                                                                     PDJ00520
C COMMON BLOCK COORD CONTAINS 0,1, OR 2 DEPENDING ON WHETHER          PDJ00530
C THE PROBLEM IS IN CARTESIAN,CYLINDRICAL, OR SPHERICAL               PDJ00540
```

```
C COORDINATES, RESPECTIVELY.                                          PDJ00550
C                                                                     PDJ00560
C COMMON BLOCK MESH CONTAINS THE USER SPECIFIED SPATIAL               PDJ00570
C GRID POINTS.                                                        PDJ00580
C                                                                     PDJ00590
      COMMON/COORD/ICORD                                              PDJ00600
      COMMON /MESH/ X(1)                                              PDJ00610
C                                                                     PDJ00620
      ANOISE = 2.22044605D-16                                         PDJ00630
      EPSJ = DSQRT(ANOISE)                                            PDJ00640
      N1 = 0                                                          PDJ00650
      NOML = NO * ML                                                  PDJ00660
      NM1 = NO - 1                                                    PDJ00670
      D1= 0.                                                          PDJ00680
      DO 5 I=1,N                                                      PDJ00690
    5 D1= D1+ FSAVE(I)**2                                             PDJ00700
      RO = DABS(H)*DSQRT(D1)*1.D03*ANOISE                             PDJ00710
      J1 = NOML                                                       PDJ00720
C                                                                     PDJ00730
C CALCULATE THE NEEDED PARTIALS WITH RESPECT TO EACH Y.               PDJ00740
C                                                                     PDJ00750
      DO 290 J=1,N                                                    PDJ00760
      N1 = N1 + 1                                                     PDJ00770
      J1 = J1 + NO                                                    PDJ00780
      YJ = U(J,1)                                                     PDJ00790
      R = EPSJ*YMAX(J)                                                PDJ00800
      R = DMAX1(R,RO)                                                 PDJ00810
      U(J,1) = U(J,1) + R                                             PDJ00820
      D1=-EL*H/R                                                      PDJ00830
      IWORK = IWORK + 1                                               PDJ00840
C                                                                     PDJ00850
C CALCULATE WHICH BLOCK THE PARTIAL IS NEEDED IN.                     PDJ00860
C                                                                     PDJ00870
      JEQ = ((J - 1) / NPDE) + 1                                      PDJ00880
      ILIM = NPTS-1                                                   PDJ00890
      ICORD1 = ICORD + 1                                              PDJ00900
      IF (JEQ.GE.ILIM) GO TO 72                                       PDJ00910
C                                                                     PDJ00920
C CALCULATE UDOT WITHOUT ANY BOUNDARY CONDITIONS.                     PDJ00930
C                                                                     PDJ00940
      IF (JEQ.GT.2.AND.JEQ.LT.ILIM) GO TO 83                          PDJ00950
C                                                                     PDJ00960
C UPDATE BOUNDARY VALUES AT THE LEFT BOUNDARY                         PDJ00970
C                                                                     PDJ00980
      CALL BNDRY(T,X(1),U,ALPHA,BETA,GAMMA,NPDE)                      PDJ00990
      ITEST = 0                                                       PDJ01000
      DO 10 K=1,NPDE                                                  PDJ01010
        IF (BETA(K).NE.0.0)  GO TO 10                                 PDJ01020
        U(K,1) = GAMMA(K)/ALPHA(K)                                    PDJ01030
        ITEST = ITEST + 1                                             PDJ01040
   10 CONTINUE                                                        PDJ01050
      DXI = 1./(X(2)-X(1))                                            PDJ01060
      IF (ITEST.EQ.0) GO TO 20                                        PDJ01070
      IF (ITEST.EQ.NPDE)  GO TO 45                                    PDJ01080
      CALL BNDRY(T,X(1),U,ALPHA,BETA,GAMMA,NPDE)                      PDJ01090
C                                                                     PDJ01100
C EVALUATE 'DIFFUSION COEFFICIENTS', D, AT THE LEFT BOUNDARY          PDJ01110
```

```
C                                                                       PDJ01120
   20 CALL D(T,X(1),U,DVAL,NPDE)                                        PDJ01130
C                                                                       PDJ01140
C FORM APPROXIMATION TO DU/DX AT THE LEFT BOUNDARY                      PDJ01150
C                                                                       PDJ01160
      DO 40 K=1,NPDE                                                    PDJ01170
        IF(BETA(K).NE.0.0)  GO TO 30                                    PDJ01180
        UX(K) = DXI *(U(K,2) - U(K,1))                                  PDJ01190
        GO TO 40                                                        PDJ01200
   30   UX(K) = (GAMMA(K) - ALPHA(K)*U(K,1))/BETA(K)                    PDJ01210
   40 CONTINUE                                                          PDJ01220
C                                                                       PDJ01230
C EVALUATE U-AVERAGE IN THE FIRST INTERVAL                              PDJ01240
C                                                                       PDJ01250
   45 DO 50 K=1,NPDE                                                    PDJ01260
        UAVG(K) = .5*(U(K,2) + U(K,1))                                  PDJ01270
   50 CONTINUE                                                          PDJ01280
C                                                                       PDJ01290
C EVALUATE 'DIFFUSION COEFFICIENTS', D, IN THE FIRST INTERVAL           PDJ01300
C                                                                       PDJ01310
      XAVGR = .5 * ( X(2) + X(1) )                                      PDJ01320
      CALL D(T,XAVGR,UAVG,DVAL(1,1,2),NPDE)                             PDJ01330
      DXIR = DXI                                                        PDJ01340
      DXIL = 1.                                                         PDJ01350
      IF (ICORD.EQ.0) GO TO 55                                          PDJ01360
      DXIL = X(1)**ICORD                                                PDJ01370
      DXIR = XAVGR**ICORD * DXI                                         PDJ01380
   55 DXIC = FLOAT(ICORD1) / (XAVGR**ICORD1 - X(1)**ICORD1)             PDJ01390
C                                                                       PDJ01400
C EVALUATE DUXX AT THE LEFT BOUNDARY                                    PDJ01410
C                                                                       PDJ01420
      DO 60 K=1,NPDE                                                    PDJ01430
        DO 60 L=1,NPDE                                                  PDJ01440
          DVAL(K,L,1)=DXIC*(DVAL(K,L,2)*(U(L,2)-U(L,1))*DXIR-           PDJ01450
     *       DVAL(K,L,1)*UX(L)*DXIL)                                    PDJ01460
   60 CONTINUE                                                          PDJ01470
      IF (ITEST.EQ.NPDE) GO TO 65                                       PDJ01480
C                                                                       PDJ01490
C EVALUATE RIGHT HAND SIDE OF PDE'S AT THE LEFT BOUNDARY                PDJ01500
C                                                                       PDJ01510
      CALL F(T,X(1),U,UX,DVAL,UDOT,NPDE)                                PDJ01520
C                                                                       PDJ01530
C DECOMPOSE THE (A) MATRIX IF NECESSARY                                 PDJ01540
C                                                                       PDJ01550
      IF (MATRIX.EQ.0) GO TO 65                                         PDJ01560
      CALL AMATRX(NPDE,T,X(1),U)                                        PDJ01570
      CALL DECOMP(NPDE,NPDE,A,IP)                                       PDJ01580
      CALL SOLVE(NPDE,NPDE,A,UDOT(1,1),IP)                              PDJ01590
C                                                                       PDJ01600
C SET UDOT = 0 FOR KNOWN LEFT BOUNDARY VALUES                           PDJ01610
C                                                                       PDJ01620
   65 DO 70 K=1,NPDE                                                    PDJ01630
        IF (BETA(K).EQ.0.0) UDOT(K,1)=0.0                               PDJ01640
   70 CONTINUE                                                          PDJ01650
C                                                                       PDJ01660
C UPDATE BOUNDARY VALUES AT THE RIGHT BOUNDARY                          PDJ01670
C                                                                       PDJ01680
```

```
   72 CALL BNDRY(T,X(NPTS),U(1,NPTS),ALPHA,BETA,GAMMA,NPDE)          PDJ01690
      ITEST = 0                                                      PDJ01700
      DO 75 K=1,NPDE                                                 PDJ01710
        IF(BETA(K).NE.0.0)  GO TO 75                                 PDJ01720
        U(K,NPTS) = GAMMA(K)/ALPHA(K)                                PDJ01730
        ITEST = ITEST + 1                                            PDJ01740
   75 CONTINUE                                                       PDJ01750
   77 IF (JEQ.EQ.1) GO TO 79                                         PDJ01760
      IF (JEQ.EQ.2) GO TO 80                                         PDJ01770
      IF (JEQ.EQ.ILIM) GO TO 81                                      PDJ01780
C                                                                    PDJ01790
C CALCULATE ONE BLOCK BEFORE THE RIGHT BOUNDARY.                     PDJ01800
C                                                                    PDJ01810
      IBEG = ILIM                                                    PDJ01820
      IEND = ILIM                                                    PDJ01830
      I1 = IBEG - 1                                                  PDJ01840
      GO TO 87                                                       PDJ01850
C                                                                    PDJ01860
C CALCULATE ONE BLOCK AFTER THE LEFT BOUNDARY.                       PDJ01870
C                                                                    PDJ01880
   79 IBEG = 2                                                       PDJ01890
      IEND = 2                                                       PDJ01900
      I1 = 1                                                         PDJ01910
      GO TO 84                                                       PDJ01920
C                                                                    PDJ01930
C CALCULATE TWO BLOCKS AFTER THE LEFT BOUNDARY.                      PDJ01940
C                                                                    PDJ01950
   80 IBEG = 2                                                       PDJ01960
      IEND = 3                                                       PDJ01970
      I1 = 1                                                         PDJ01980
      GO TO 84                                                       PDJ01990
C                                                                    PDJ02000
C CALCULATE TWO BLOCKS BEFORE THE RIGHT BOUNDARY.                    PDJ02010
C                                                                    PDJ02020
   81 IBEG = NPTS - 2                                                PDJ02030
      IEND = ILIM                                                    PDJ02040
      I1 = IBEG-1                                                    PDJ02050
      GO TO 87                                                       PDJ02060
C                                                                    PDJ02070
C CALCULATE THREE BLOCKS WITHOUT BOUNDARY CONDITIONS.                PDJ02080
C                                                                    PDJ02090
   83 IBEG = JEQ - 1                                                 PDJ02100
      IEND = JEQ + 1                                                 PDJ02110
      I1 = IBEG-1                                                    PDJ02120
C                                                                    PDJ02130
C MAIN LOOP TO FORM ODE'S AT THE INTERIOR GRID POINTS                PDJ02140
C                                                                    PDJ02150
   87 XAVGR = .5 * (X(IBEG) + X(IBEG-1))                             PDJ02160
      DXIR = 1.                                                      PDJ02170
      IF (ICORD.NE.0) DXIR = XAVGR**ICORD                            PDJ02180
      DXIR = DXIR / (X(IBEG) - X(IBEG-1))                            PDJ02190
      DO 88 K=1,NPDE                                                 PDJ02200
        UAVG(K) = .5 * (U(K,IBEG) + U(K,IBEG-1))                     PDJ02210
   88 CONTINUE                                                       PDJ02220
      CALL D(T,XAVGR,UAVG,DVAL(1,1,2),NPDE)                          PDJ02230
C                                                                    PDJ02240
C SET ZEROES ONE BLOCK BEFORE.                                       PDJ02250
```

```
C                                                                       PDJ02260
      DO 93 I=1,NPDE                                                    PDJ02270
   93 UDOT(I,IBEG-1) = 0.                                               PDJ02280
   84 IBCK = 1                                                          PDJ02290
      IFWD = 2                                                          PDJ02300
      I2 = IEND+1                                                       PDJ02310
      IF (JEQ.GE.ILIM) GO TO 97                                         PDJ02320
C                                                                       PDJ02330
C SET ZEROES ONE BLOCK AFTER.                                           PDJ02340
C                                                                       PDJ02350
      DO 95 I=1,NPDE                                                    PDJ02360
   95 UDOT(I,IEND+1) = 0.                                               PDJ02370
   97 DO 100 I=IBEG,IEND                                                PDJ02380
        ITEMP = IBCK                                                    PDJ02390
        IBCK = IFWD                                                     PDJ02400
        IFWD = ITEMP                                                    PDJ02410
        XAVGL = XAVGR                                                   PDJ02420
        XAVGR = .5 * (X(I+1) + X(I))                                    PDJ02430
        DXI = 1./(X(I+1)-X(I-1))                                        PDJ02440
        DXIL = DXIR                                                     PDJ02450
        DXIR = 1.                                                       PDJ02460
        IF (ICORD.NE.0) DXIR =XAVGR**ICORD                              PDJ02470
        DXIR = DXIR / (X(I+1)-X(I))                                     PDJ02480
        CXIC = FLOAT(ICORD1) / (XAVGR**ICORD1 - XAVGL**ICORD1)          PDJ02490
C                                                                       PDJ02500
C EVALUATE DU/DX AND U-AVERAGE AT THE I-TH GRID POINT                   PDJ02510
C                                                                       PDJ02520
        DO 85 K=1,NPDE                                                  PDJ02530
          UX(K) = DXI*(U(K,I+1) - U(K,I-1))                             PDJ02540
          UAVG(K) = .5*(U(K,I+1) + U(K,I))                              PDJ02550
   85   CONTINUE                                                        PDJ02560
C                                                                       PDJ02570
C EVALUATE 'DIFFUSION COEFFICIENTS', D, IN THE I-TH INTERVAL            PDJ02580
C                                                                       PDJ02590
      CALL D(T,XAVGR,UAVG,DVAL(1,1,IFWD),NPDE)                          PDJ02600
C                                                                       PDJ02610
C EVALUATE DUXX AT THE I-TH GRID POINT                                  PDJ02620
C                                                                       PDJ02630
        DO 90 K=1,NPDE                                                  PDJ02640
          DO 90 L=1,NPDE                                                PDJ02650
            DVAL(K,L,IBCK)=DXIC*(DVAL(K,L,IFWD)*(U(L,I+1)-              PDJ02660
     *           U(L,I))*DXIR - DVAL(K,L,IBCK)*(U(L,I)                  PDJ02670
     *           -U(L,I-1))*DXIL)                                       PDJ02680
   90   CONTINUE                                                        PDJ02690
C                                                                       PDJ02700
C EVALUATE RIGHT HAND SIDE OF PDE'S AT THE I-TH GRID POINT              PDJ02710
C                                                                       PDJ02720
      CALL F(T,X(I),U(1,I),UX,DVAL(1,1,IBCK),UDOT(1,I),NPDE)            PDJ02730
C                                                                       PDJ02740
C DECOMPOSE THE (A) MATRIX IF NECESSARY                                 PDJ02750
C                                                                       PDJ02760
      IF (MATRIX.EQ.0) GO TO 100                                        PDJ02770
      CALL AMATRX(NPDE,T,X(I),U(1,I))                                   PDJ02780
      CALL DECOMP(NPDE,NPDE,A,IP)                                       PDJ02790
      CALL SOLVE(NPDE,NPDE,A,UDOT(1,I),IP)                              PDJ02800
  100 CONTINUE                                                          PDJ02810
C                                                                       PDJ02820
```

```
C FINISH UPDATING THE RIGHT BOUNDARY IF NECESSARY                       PDJ02830
C                                                                       PDJ02840
      IF (JEQ.LT.ILIM) GO TO 165                                        PDJ02850
  110 IF (ITEST.EQ.0) GO TO 120                                         PDJ02860
      IF(ITEST.EQ.NPDE)   GO TO 155                                     PDJ02870
      CALL BNDRY(T,X(NPTS),U(1,NPTS),ALPHA,BETA,GAMMA,NPDE)             PDJ02880
C                                                                       PDJ02890
C EVALUATE 'DIFFUSION COEFFICIENTS', D, AT THE RIGHT BOUNDARY           PDJ02900
C                                                                       PDJ02910
  120 CALL D(T,X(NPTS),U(1,NPTS),DVAL(1,1,IBCK),NPDE)                   PDJ02920
C                                                                       PDJ02930
C FORM APPROXIMATIONS TO DU/DX AT THE RIGHT BOUNDARY                    PDJ02940
C                                                                       PDJ02950
      DXI = 1. / (X(NPTS) - X(ILIM))                                    PDJ02960
      DO 140 K=1,NPDE                                                   PDJ02970
        IF(BETA(K).NE.0.0)  GO TO 130                                   PDJ02980
        UX(K) = DXI *(U(K,NPTS) - U(K,ILIM))                            PDJ02990
        GO TO 140                                                       PDJ03000
  130   UX(K) = (GAMMA(K) - ALPHA(K)*U(K,NPTS))/BETA(K)                 PDJ03010
  140 CONTINUE                                                          PDJ03020
      DXIL = DXIR                                                       PDJ03030
      DXIR = 1                                                          PDJ03040
      IF (ICORD.NE.0) DXIR = X(NPTS)**ICORD                             PDJ03050
      DXIC=FLOAT(ICORD1) /(X(NPTS)**ICORD1-XAVGR**ICORD1)               PDJ03060
C                                                                       PDJ03070
C EVALUATE DUXX AT THE RIGHT BOUNDARY                                   PDJ03080
C                                                                       PDJ03090
      DO 150 K=1,NPDE                                                   PDJ03100
        DO 150 L=1,NPDE                                                 PDJ03110
          DVAL(K,L,IBCK)=DXIC*(DVAL(K,L,IBCK)*UX(L)*DXIR                PDJ03120
     *     -DVAL(K,L,IFWD)*(U(L,NPTS)-U(L,ILIM))*DXIL)                  PDJ03130
  150 CONTINUE                                                          PDJ03140
      I2 = NPTS                                                         PDJ03150
C                                                                       PDJ03160
C EVALUATE RIGHT HAND SIDE OF PDE'S AT THE RIGHT BOUNDARY               PDJ03170
C                                                                       PDJ03180
      CALL F(T,X(NPTS),U(1,NPTS),UX,DVAL(1,1,IBCK),                     PDJ03190
     *   UDOT(1,NPTS),NPDE)                                             PDJ03200
C                                                                       PDJ03210
C DECOMPOSE THE (A) MATRIX IF NECESSARY                                 PDJ03220
C                                                                       PDJ03230
      IF (MATRIX.EQ.0) GO TO 155                                        PDJ03240
      CALL AMATRX(NPDE,T,X(NPTS),U(1,NPTS))                             PDJ03250
      CALL DECOMP(NPDE,NPDE,A,IP)                                       PDJ03260
      CALL SOLVE(NPDE,NPDE,A,UDOT(1,NPTS),IP)                           PDJ03270
C                                                                       PDJ03280
C SET UDOT = 0 FOR KNOWN RIGHT BOUNDARY VALUES                          PDJ03290
C                                                                       PDJ03300
  155 DO 160 K=1,NPDE                                                   PDJ03310
        IF (BETA(K).EQ.0.0) UDOT(K,NPTS)=0.                             PDJ03320
  160  CONTINUE                                                         PDJ03330
  165 CONTINUE                                                          PDJ03340
      DO 285 I = I1,I2                                                  PDJ03350
      IBEG = 1                                                          PDJ03360
      IEND = NPDE                                                       PDJ03370
C                                                                       PDJ03380
C CALCULATE THE NEEDED PW'S FOR BANDS.                                  PDJ03390
```

```
C                                                                       PDJ03400
      IF(I.EQ.I2.AND.N1.EQ.1.AND.JEQ.LT.ILIM) GO TO 285                 PDJ03410
      IF(I.EQ.I1.AND.N1.EQ.NPDE.AND.JEQ.GT.2) GO TO 285                 PDJ03420
      IF (I.EQ.I1.AND.JEQ.GT.2) IBEG = N1 + 1                           PDJ03430
      IF (I.EQ.I2.AND.JEQ.LT.ILIM) IEND = N1 - 1                        PDJ03440
      DO 280 L=IBEG,IEND                                                PDJ03450
      K = (I-1)*NPDE+L                                                  PDJ03460
      JSAVE = J1-NM1*K                                                  PDJ03470
      PW(JSAVE) = (UDOT(L,I) - FSAVE(K))*D1                             PDJ03480
      IF (UDOT(L,I).EQ.0.) PW(JSAVE) = 0.                               PDJ03490
  280 CONTINUE                                                          PDJ03500
  285 CONTINUE                                                          PDJ03510
      IF (N1.EQ.NPDE) N1 = 0                                            PDJ03520
  290 U(J,1) = YJ                                                       PDJ03530
      RETURN                                                            PDJ03540
      END                                                               PDJ03550
```

ACKNOWLEDGMENTS

The author wishes to express his sincere thanks and appreciation to his advisory committee: Dr. Richard F. Sincovec, Associate Professor of Computer Science at Kansas State University; Dr. Nasir Ahmed, Associate Professor of Computer Science at Kansas State University; and Dr. Paul S. Fisher, Head of the Department of Computer Science at Kansas State University. A special note of appreciation is given to Dr. Sincovec for providing the basis of this report and for giving freely of his time. Finally special thanks to Teresa Morse, for her many hours of help in preparing and editing this paper.

TWO MODIFICATIONS TO THE SOFTWARE INTERFACE PACKAGE FOR NONLINEAR PARTIAL DIFFERENTIAL EQUATIONS

by

RICHARD LEE MORSE

B.S., Kansas State University, 1971

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1974

ABSTRACT

This paper presents two modifications to a software interface package for nonlinear partial differential equations being jointly developed by Dr. Richard F. Sincovec, Kansas State University and Dr. Niel K. Madsen, Lawrence Livermore Laboratory. The first change discussed is a modification of the PDEONE routine so that the package can handle coupled sets of partial differential equations. The second modification is the addition of the PDEJAC routine that will generate the Jacobian matrix in an efficient manner.