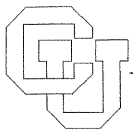
HYPACK: A Software Package for Hyperbolic and Elliptic Equations in Two Space Dimensions

John Gary

CU-CS-076-75 December 1975*



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

* Revised July 1976

Version 1

HYPACK: A software package for hyperbolic and elliptic equations in two space dimensions

by

John Gary
Department of Computer Science
University of Colorado
Boulder, Colorado 80302

TR #CU-CS-076-75

December 1975
Revised July 1976

Contents

1. Introduction. This is a preliminary description of a FORTRAN program package for the solution of a system of hyperbolic equations coupled with a single elliptic equation. The system is restricted to Cartesian coordinates (x,y) on a rectangular domain $(x_a<x<x_b, y_a<y<y_b)$. The equations are allowed to contain parabolic terms (i.e., $\partial^2 u/\partial x^2$ and $\partial^2 u/\partial y^2$), however we assume the diffusion terms are small so that the proper boundary conditions are hyperbolic (except for the single elliptic variable). An arbitrary number of hyperbolic variables (≥ 1) are allowed, however there can be at most one elliptic variable. The elliptic variable may be absent. Periodic boundary conditions are allowed in either the x or y direction, or in both directions. If one variable is periodic, all variables must be periodic. In the non-periodic case the user can set the time derivative of the hyperbolic variables at the boundary in a separate boundary subroutine. This is intended to allow use of the characteristics of the hyperbolic system in the definition of the boundary conditions. This is described later. It is somewhat complicated. However we feel the user must have control of the boundary conditons, this cannot be completely automated in two dimensions. The boundary conditions are a difficult problem for hyperbolic equations. Their treatment is a central part of the package.

The elliptic equation is solved by the package of Adams, Swartztrauber, and Sweet which uses a direct method of second order accuracy combined with deferred corrections to yield a fourth order finite difference method. This allows periodic, Dirichlet or Neuman conditions on each pair of opposite sides.

The spatial derivatives $(U_x, U_y, U_{xx}, U_{xy}, U_{yy})$ in the hyperbolic equations, and on the right side of the elliptic equations, are replaced by finite differences, of either second order (three point) or fourth order (five point) accuracy. In addition the user is allowed to include a diffusion term which is a five point approximation of $-(\varepsilon^x U_{x4} + \varepsilon^y U_{y4})$ where the user supplies the constants ε^x and ε^y.

This spatial discretization produces a semi discrete "method of lines" approximation; that is, a system of ordinary differential equations. This system is solved by either a fourth order Runge-Kutta-Fehlberg [1]

method or a second order leapfrog. Both methods are adapted to handle data contained in LCM (or ECS) rather than central memory. Thus our package can effectively use ECS on the CDC 6000 series or LCM on the CDC 7600. We will eventually adapt the code to handle problems contained in central memory. Eventually we will adapt the package to run problems whose data is disk contained, but this is a more difficult problem.

The output routines presently produce tables of array values, and contour plots.

It is difficult to write a package, especially in two or three dimensions, which can handle a significant percentage of problems without some modification. Even after the restriction to Cartesian coordinates on a rectangle, modifications may be required. Therefore, we have attempted to break the program into subroutine modules corresponding to the functions which the user is likely to want to modify. Development of the proper modules is going to require some experiment. This development is going to be "tuned" for atmospheric models, especially vorticity-stream function models, anelastic cloud models, perhaps some types of ocean models, and perhaps some types of lee wave and eventually primitive equation models.

2. The basic equations. The system of equations which the package is set up to solve is the following.

$$\frac{\partial \underline{u}}{\partial t} = \frac{\partial \underline{f}}{\partial x} + \frac{\partial \underline{g}}{\partial y} + \underline{r}\,(x,y,t,\underline{w},\underline{w}_x,\underline{w}_y,\underline{w}_{xx},\underline{w}_{xy},\underline{w}_{yy}) \tag{1}$$

$$ax(x)\frac{\partial^2\phi}{\partial x^2} + bx(x)\frac{\partial\phi}{\partial x} + cx(x) + ay(y)\frac{\partial^2\phi}{\partial y^2} + by(y)\frac{\partial\phi}{\partial y} + cy(y)\phi = s(x,y,t,\underline{u},\underline{u}_x,\underline{u}_y,\underline{u}_{xx},\underline{u}_{xy},\underline{u}_{yy})$$

The solution can be specified at the boundary ($\phi = \rho(x,y)$), or a derivative condition can be specified ($\phi_x + \gamma_1\phi = \rho(x,y)$, or $\phi_y + \gamma_2\phi = \rho(x,y)$ where γ_i are constant. The variables are the following.

$$\underline{u} = (u_1,\ldots,u_M)^T \qquad \underline{f} = (f_1,..,f_M)^T$$
$$\underline{w} = (u_1,\ldots,u_M,\phi) \qquad \underline{g} = (g_1,..,g_M)^T$$
$$u_i = u_i(x,y,t) \qquad \underline{r} = (h_1,\ldots,h_M)^T \qquad (3)$$

$$f_i = f_i(x,y,t,\underline{w}) \qquad g_i = g_i(x,y,t,\underline{w})$$
$$r_i = r_i(x,y,t,\underline{w},\underline{w}_x,\underline{w}_y,\underline{w}_{xx},\underline{w}_{yy})$$

The system of hyperbolic equations is converted to a system of ordinary differential equations (a semi-discrete approximation) by the "method of lines." The spatial derivatives are replaced by finite differences using one sided differences near the boundary and making certain modifications for the boundary conditions. To illustrate the method consider the single equation in one space dimension

i.e. $u_t = -u_x$
$u(x,o) = \sin(x)$
$u(o,t) = -\sin(t)$
$o \le x \le 1$
$o \le t$

whose solution is $u(x,t) = \sin(x-t)$. Suppose the mesh is $x_i = (i-1)\Delta x$ where $1<i<NX$. Using second order difference approximations in the interior and first order at the boundary, the ODE system is

$$\frac{dU_i}{dt} = -\frac{U_{i+1}-U_{i-1}}{2\Delta x} \qquad 1 \le i \le NX-1$$

$$\frac{dU_i}{dt} = -\frac{U_i-U_{i-1}}{\Delta x} \qquad i = NX$$

$$\frac{dU_o}{dt} = -\cos(t)$$

Note that the time derivative of U_0 at the inflow boundary is obtained by differentiation of the boundary condition $U_0(t) = -\sin(t)$. The time derivative at the outflow is obtained from a one sided difference. At an outflow boundary, that is one where the characteristic lies on the inside of the domain, no boundary condition should be imposed. This package will compute the time derivative at both boundaries using one sided difference approximations. The user must then supply a routine called HPBDY to correct the time derivative of the "inflow" variables. Generally the variables are not characteristic variables which means that we cannot speak of "inflow" and "outflow" variables. We will discuss this complication later.

It is convenient to have a differential equation for each variable, therefore we differentiate the boundary condition although this is not absolutely necessary [2]. Later we will supply other ODE solvers such as the variable order Adams integrator due to Shampine [3]. We will provide additional details in the next section.

3. The use of the package. The user must supply subroutines to evaluate the functions f,g,r and s, and the coefficients ax(x), ...ay(y),...cy(y). Not all of these functions need be called - "flags" are supplied to suppress the subroutine calls for the unused functions. In this case, dummy functions must be supplied in order to load the program package.

The mesh. The mesh must be a rectangle with equally spaced mesh points in each direction. The user supplies the input parameters XA,XB,YA,YB,NXPTS, and NYPTS. The mesh points are

$$X_i = XA + (i-1)*(XB-XA)/(NXPTS-1)$$
$$Y_j = YA + (j-1)*(YB-YA)/(NYPTS-1)$$

The user supplies parameters MHPEQN and MELEQN which determine the number of hyperbolic variables (M in equations (3) above) and MELEQN which indicates the presence of an elliptic variable. The restrictions are $0 \le MELEQN \le 1$ and $1 \le MHPEQN \le NWQ$. For the definition of NWQ see the section on "array specification" below.

The user supplies parameters NHXBDY and NHYBDY which determine the type of the boundary condition. If NHXBDY=0 then the boundary condition along the sides x=XA and x=XB are periodic, that is $u_m(x,y) = u_m(x+P,y)$ for all variables, where P = XB-XA is the period.
In the case NHXBDY=1, the user supplies a subroutine to set the boundary conditions for the hyperbolic variables. This is discussed in the section below on "boundary conditions". The meaning of NHYBDY is similar. The parameters NEXBDY and NEYBDY are used to set boundary conditions for the elliptic variable ϕ as discussed below. If the boundary conditions are periodic (NHXBDY=0) then the values U(1,J) and U(NXPTS,J) are equal. Thus the code computes the solution for U(I,J) with $1 \le I \le NX$ where

$$NX = \begin{cases} NXPTS & \text{if } NHXBDY \neq 0 \\ NXPTS-1 & \text{if } NHXBDY = 0 \end{cases}$$

User supplied coefficient routines. The subroutines to compute the functions f,g,r and s are written in the following format. The names are

f - FN
g - GN
r - RN
s - SN

The calling sequences are

FN(J,Y,T,U,F,NWX,NWQ)
GN(J,Y,T,U,F,NWX,NWQ)
RN(J,Y,T,U,UX,UY,UXX, UXY,UYY,F,NWX,NWQ)
SN(J,Y,T,U,UX,UY,UXX, UXY,UYY,G,NWX,NWQ)

The input arguments U,UX,UY,UXX,UXY,UYY and output argument F are variable dimensioned arrays and must appear in a DIMENSION statement in these subroutines with the dimension (NWX,NWQ). The array U(I,M) contains the variables (hyperbolic and elliptic) along one line of the mesh. Along the line the index is J and the ordinate is Y. The value of the time is T. The routine FN must take the input parameters (J,Y,T,U,NWX,NWQ) and compute the value of the function F(I,M) which is the function f in equation (1). For the input arrays U,UX,...UYY the range of the subscript M is $1 \le M \le MEQN$

where MEQN = MHPEQN + MELEQN is the total number of variables. These arrays are all indexed by (I,M) where $1 \le I \le NX$. The first partial derivative with respect to x of each unknown is computed by the package using a finite difference formula along the entire line with the given value of J and Y. The result is placed in the array UX and is then available within the RN and SN subroutines. A similar statement holds for the arrays UY,UXX,UXY, and UYY. One sided differences are used near the boundary. For the output F(I,M) we have $1 \le I \le NX$ and $1 \le M \le MHPEQN$. The right side of the Poisson equation for ϕ is a scaler, so the dimension of the result G for SN is G(NWX).

The terms $\underline{f}_x$ and $\underline{g}_y$ in equation (1) are included so that nonlinear terms can be written in conservation form. Otherwise the method of lines integration may be unstable [4]. The functions $\underline{f}$ and $\underline{g}$ are evaluated and then the derivatives $\underline{f}_x$ and $\underline{g}_y$ are approximated by a finite difference of second or fourth order depending on the parameter NORDER.

There is a labeled COMMON block which the user may use in these routines namely,

```
COMMON/FNCOM/NX,NY,XA,XB,YA,YB,DLX,DLY,NCASE
```

Here DLX and DLY are the mesh increments

```
DLX = (XB-XA)/(NXPTS-1)
DLX = (YB-YA)/(NYPTS-1)
```

The parameter NCASE is read in at the start of each run. It is not used by the software package, but is included for the user's convenience in selecting different cases.

As an example, suppose we have the following equation for a single unknown

$$u_t = -uu_x - uu_y + \sin(t) = -.5*((u^2)_x + (u^2)_y) + \sin(t)$$

In this case MHPEQN = 1 and MELEQN = 0. Then the routine might be

```
      SUBROUTINE FN(J,Y,T,U,F,NWX,NWQ)
      COMMON/FNCOM/NX,NY,XA,NB,YA,YB,DLX,DLY,NCASE
      DIMENSION U(NWX,NWQ),F(NWX,NWQ)
      DO 10 I = 1,NX
10    F(I,1) = -.5*U(I,1)**2
      RETURN
      END
```

The same code will do for GN. The subroutine RN is given below.

```
      SUBROUTINE RN(J,Y,T,U,UX,UY,UXX,UYY,F,NWX,NWQ)
      COMMON/FNCOM/NX,NY,XA,XB,YA,YB,DLX,DLY,NCASE
      DIMENSION U(NWX,NWQ),UX(NWX,NWQ),UY(NWX,NWQ),
     X     UXX(NWX,NWQ),UYY(NWX,NWQ)
      SINT = SIN(T)
      DO 10 I=1,NX
10    F(I,1) = SINT
      RETURN
      END
```

The parameters ISFN,ISGN,ISRN are "flags" to indicate that the routines FN, GN and RN are used. If ISFN = 0 the routine FN is not called, although a "dummy" routine with that name must be present in order to load the program package. In the above example ISFN = ISGN = ISRN = 1.

The same equation could be solved with ISFN = ISGN = 0, and ISRN = 1 with the following RN.

```
      SUBROUTINE RN(J,Y,T,U,UX,UY,UXX,UYY,F,NWX,NWQ)
      COMMON /FNCOM/NX,NY,XA,XB,YA,YB,DLX,DLY,NCASE
      DIMENSION U(NWX,NWQ),UX(NWX,NWQ),UY(NWX,NWQ),
     X     UXX(NWX,NWQ),UYY(NWX,NWQ)
      SINT = SIN(T)
      DO 10 I = 1,NX
10    F(I,1) = -U(I,1)*(UX(I,1) + UY(I,1)) + SINT
      RETURN
      END
```

Additional flags are included to control the computation of the finite differences UX,UY,UXX,UXY, and UYY. If the parameters ISRUX,ISRUY, ISRUXX,ISRUYY vanish, then the corresponding finite difference is not computed, instead the derivative array is a "dummy" array. Similar parameters ISSUX,... are used with the SN routine. These flags are included to avoid needless computation of the finite differences.

The hyperbolic boundary conditions. The hyperbolic boundary conditions are set by the user supplied routine HPBDY and the parameters NHXBDY and NHYBDY. If NHXBDY = 0, then a periodic boundary condition is used in the x-direction. The periodicity condition is used to compute the finite difference approximations to the derivatives instead of the one sided differences near the boundaries. In the nonperiodic case the routine HPBDY allows the user to adjust the time derivatives at the boundary. The argument list for HPBDY is

```
HPBDY(JBDY,J,Y,T,U,DTU,NWX,NWQ)
```

This routine is called once along each horizontal line; that is, once for each value of J and Y. The parameters JBDY and Y are redundant parameters included for convenience only. If the horizontal line is a boundary line (J=1 or J=NY) then JBDY=2, otherwise JBDY=1. Thus JBDY can be used in a computed GOTO within the HPBDY routine. The parameters U and DTU are variable dimensioned arrays. The solution variables along the mesh line are contained in the array U(I,M) where $1 \le I \le NX$ and $1 \le M \le MEQN$. The time derivative of these variables at the time T for each mesh point is held in the array DTU(I,M). These time derivatives are computed using one sided difference approximations to approximate spatial derivatives. All these arguments are input parameters. The time derivative can be corrected at the boundary in accordance with the boundary conditions. If JBDY=1, then the boundary points are at I=1 and I=NX. If JBDY=2, then the entire horizontal line consists of boundary points.

To see how the boundary conditions might be converted into a correction on the time derivative consider the following examples. First the single equation

$$u_t = -u_x - u_y$$

with the boundary conditions

$$u(o,y,t) = \sin(y-2t) \quad \text{at } x=XA=0$$

$$u(x,o,t) = \sin(x-2t) \quad \text{at } y=YA=0$$

In this case the HPBDY routine is obtained by differentiation of the above boundary condition. This HPBDY routine might be

```
      SUBROUTINE HPBDY(JBDY,J,Y,T,U,DTU,NWX,NWQ)
      COMMON/FNCOM/NX,NY,XA,XB,YA,YB,DLX,DLY,NCASE
      DIMENSION U(NWX,NWQ),DTU(NWX,NWQ)
      GOTO(10,50),JBDY
C     SET THE LOWER BOUNDARY LINE AT X=0 (JBDY=1)
 10   DTU(1,1)=-2.*COS(Y-2.*T)
      RETURN
C     SET THE LOWER BOUNDARY LINE AT Y=0 (JBDY=2)
 50   IF(J.GT.1)RETURN
      DO 60 I=1,NX
      X=XA+(I-1)*DLX
 60   DTU(I,1)=-2.*COS(X-2.*T)
      RETURN
      END
```

Note that we only correct at the "inflow" points along the left and lower boundary. At the other boundary points the time derivative obtained from one sided differences is used.

Next we consider an example of a system with two equations in which the unknowns are not characteristic variables. The equations are

$$\frac{\partial u_1}{\partial t} = \frac{\partial u_2}{\partial x} + \frac{\partial u_1}{\partial y}$$

$$\frac{\partial u_2}{\partial t} = \frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} \qquad (5)$$

The boundary conditions are periodic in the y direction, so that NHYBDY=0. We will set the boundary conditions in the x-direction using the HPBDY routine, thus NHXBDY=1. We can place these equations in diagonal or characteristic form by the transformation

$$\phi = U_1 + U_2 \qquad \psi = U_1 - U_2$$

so that the equations are

$$\frac{\partial \phi}{\partial t} = \frac{\partial \phi}{\partial x} + \frac{\partial \phi}{\partial y}$$

$$\frac{\partial \psi}{\partial t} = -\frac{\partial \psi}{\partial x} + \frac{\partial \psi}{\partial y} \qquad (6)$$

If the original system for (u_1, u_2) is written in the form

$$\frac{\partial \underline{u}}{\partial t} = A\frac{\partial \underline{u}}{\partial x} + B\frac{\partial \underline{u}}{\partial y}$$

then ϕ and ψ are the linear combinations of $\underline{U}$ which yield the eigenvectors of A. At x=XA, the inflow characteristic is ψ and the outflow ϕ. Suppose we wish to eliminate reflection. Then we should use the boundary condition ψ=0 at x=XA. We need to translate this back into a condition on the time derivatives of u_1 and u_2. We have the time derivatives of u_1 and u_2 based on the use of one sided difference approximations in the equation (5). These time derivatives are passed to the routine HPBDY in the array DTU. At the boundary we impose the condition ψ=0 or $\frac{d\psi}{dt} = 0$ and use one sided differences in the approximation of $\frac{d\phi}{dt}$. That is,

$$\frac{d\psi}{dt} = \frac{du_1}{dt} - \frac{du_2}{dt} = 0$$

$$\frac{d\phi}{dt} = \frac{du_1}{dt} + \frac{du_2}{dt} = R_1 + R_2 \qquad (7)$$

Here R_1 is the difference approximation to $\partial u_2/\partial x + \partial u_1/\partial y$, and R_2 approximates $\partial u_1/\partial x + \partial u_2/\partial y$. We can solve these equations for the derivatives of u_1 and u_2 to obtain

$$\frac{du_1}{dt} = \frac{1}{2}(R_1 + R_2)$$
$$\frac{du_2}{dt} = \frac{1}{2}(R_1 + R_2)$$

Instead of equation (7) we might consider the following system where one-sided differences are used for the non-characteristic variable u_2.

$$\frac{du_1}{dt} - \frac{du_2}{dt} = 0 \qquad \frac{du_2}{dt} = R_2$$

or

$$\frac{du_1}{dt} = R_2 \qquad \frac{du_2}{dt} = R_2 \tag{8}$$

However, some experiments indicate that it may be better to use characteristic variables in the boundary conditions as in (7) [2]. In this case the HPBDY routine might be

```
      SUBROUTINE HPBDY(JBDY,J,Y,U,DTU,NWX,NWQ)
      COMMON/FNCOM/NX,NY,XA,XB,YA,YB,DLX,DLY,NCASE
      DIMENSION U(NWX,NWQ),DTU(NWX,NWQ)
      A1 = .5*(DTU(1,1) + DTU(1,2)
      DTU(1,1) = A1
      DTU(1,2) = A1
      A1 = .5*(DTU(NX,1) - DTU(NX,2))
      DTU(NX,1) = A1
      DTU(NX,2) = -A1
      RETURN
      END
```

Note that the HPBDY routine is not called with JBDY=2 in this case because the y-boundary condition is periodic. Also note that ϕ is the inflow characteristic at the right boundary x=XB.

Boundary conditions for the elliptic equation. If MELEQN=1, then an elliptic equation is solved for a diagnostic variable. The right side of this equation is computed by the routine SN whose calling sequence is the same as RN and is given above. The elliptic boundary conditions are set by the subroutine ELBDY whose calling sequence is
ELBDY(JBDY,J,Y,T,U,UX,UY,UXX,UXY,UYY,UXA,UXB,UXI,NWX,NWQ)

This routine is called once for each mesh line (i.e., each value of J). Thus it must define the elliptic boundary conditions on only one mesh line per call. The variables JBDY,J,Y,T,U,UX,UY,UXX,UXY,UYY,NWX, AND NWQ have the same meaning as in the routine HPBDY. The boundary conditions are stored in UXA,UXB,UXI in a manner to be described below. The type of boundary is determined by the parameters NEXBDY and NEYBDY. These values are set as follows:

NEXBDY=0: periodic in x

=1: solution specified at x=XA and x=XB.

=2: solution specified at x=XA and a derivative condition at x=XB, namely $\frac{\partial \phi}{\partial x} + \gamma_{XB}\phi = \rho(XB,y)$. The values of $\rho(XB,y_j)$ are returned by ELBDY in the element UBDY(NX). Note $\rho(x_I y)$ is defined only on the boundary.

=3: derivative condition specified at x=XA and x=XB.

=4: derivative condition specified at x=XA and the solution specified at x=XB.

The values of NEYBDY have a similar meaning.

The array UXI has dimension UXI(NWX). The parameter JBDY is redundant just as it is for the HPBDY routine. If 1<J<NY, then JBDY=1. If j=1 or J=NY, then JBDY=2. If JBDY=1, then the values at the boundary point x=XA (either $\phi(XA,Y_j)$ or $\rho(XA,Y_j)$) are placed in UXA. The boundary values at x=XB are placed in UXB. If JBDY=2, then the boundary values along the J=1 or J=NY ($\phi(x_i,y)$ or $\rho(x_i,y)$ where y=YA or y=YB) are placed in the array UXI(I) where $1 \le I \le NX$. The boundary values at the endpoints are placed in UXA and UXB. In the case of derivative boundary conditions these values may be different from those in UXI(1) and UXI(NX). This rather clumsy arrangement is necessary since the data required to compute the right side of the elliptic equation is in LCM which requires this right side to be computed one line at a time.

Options which control the choice of the numerical method. The user can set the order of the finite difference approximations in space. If NORDER=1 second order (3 point) approximations are used. If NORDER=2, fourth order (5 point) approximations are used. It would be rather easy to add a "Numerov" type of fourth order scheme. This might be more effective than the fourth order explicit five point scheme.

At present there are two ODE solvers available in the package. Both are designed to advance the time step by "mesh lines" using mesh data contained in LCM and thus differ slightly from the usual ODE solvers. The parameter NTPODE controls this selection. If NTPODE=1 a Runge-Kutta-Fehlberg ODE solver is used. If NTPODE=2 a second order (in time) fixed time step leapfrog scheme is used. In this case the time step is set by the user in DLTBGN.

The parameters DLTBGN, DLTMIN, and DLTMAX allow the user to set the initial, minimum and maximum values of the time step used in the Runge-Kutta-Fehlberg. The user can set the error tolerance parameters for the Runge-Kutta-Fehlberg, one for each hyperbolic variable. These parameters are the array EPSU(M) where $1 \le M \le$ MHPEQN. The user must input an approximation to the norm of the Mth prognostic variable in the array element SOLNMX(M). This is used for scaling the error tolerance in the Runge-Kutta-Fehlberg

Stabilization of fourth order leapfrog. The leapfrog scheme when used with fourth order spatial differences is unstable for a nonperiodic boundary condition. If the flag NSTBBD is nonzero, then a stabilization term of the form

$$\gamma(u_i^{n+1} - 2u_i^n + u_i^{n-1})$$

is used for i=1,2,NX-1, and NX. The factor STBBDY(M) is used to set γ in each equation. For the simple equation

$$U_t + cU_x = 0$$

this parameter (STBBDY(1)) should be set to c. Choice of the proper parameter for a system may require some experimentation.

Setting initial conditions. A subroutine USOLN must be provided by the user to initialize the hyperbolic (prognostic) variables. The calling sequence is

USOLN(J,Y,T,U,NWX,NWQ)

The value of T for initialization will be TBGN. The routine should supply the value of U along the J^{th} line at y=Y in the array U(I,M) where $1 \le I \le$ NX, $1 \le M \le$ MHPEQN.

For test purposes it is convenient to solve a system whose exact solution is known. In this case the output routine OUTU (which is called by OUTTIM) will compute and print the error in the finite difference solution. The user must write the routine USOLN to compute the exact solution for any time t=T. For example, if MHPEQN=2 and MELEQN=1 and the hyperbolic solution is $u_1(x,y,t) = \sin(x+y-2t)$

$$u_2(x,y,t) = \cos(x+y-2t)$$

and the elliptic solution is

$$\phi(x,y,t) = \sin(x+y-t)$$

then the USOLN routine might be

```
      SUBROUTINE USOLN(J,Y,T,U,NWX,NWQ)
      COMMON/FNCOM/NX,NY,XA,XB,YA,YB,DLX,DLY,NCASE
      DIMENSION U(NWX,NWQ)
      DO 20 I=1,NX
      X=XA+(I-1)*DLX
      U(I,1)=SIN(X+Y-2.*T)
      U(I,2)=COS(X+Y-2.*T)
20    U(I,3)=SIN(X+Y-T)
      RETURN
      END
```

The routine OUTU will compute the error if the parameter NEXACT is set to NEXACT=1. If NEXACT=0 the error is not computed and USOLN will be called only if T=TBGN and IRESTR=0.

Modifications of array dimensions. The LCM data is accessed only in the routines READWL and WRITWL. To increase the size of the mesh the user must modify the declaration of the following variables which are contained in blank COMMON.

W(NWX,NWQ,16), WE(NWEX,NWEY), WEK(NWEK).

In the COMMON declaration the user must replace NWX, NWEX, NWEY, and NWEK by integer constants. The same integer constants must be used to set these variables in the MAIN program. This blank COMMON declaration is also found in the subroutines IODATA, ADVAN, OUTTIM, RELOAD, and SAVEIT. These subroutines can be converted into main programs in different overlays if it is necessary to overlay the package. These variables are transmitted to the other routines as arguments in order to reduce the number of occurrences of this blank COMMON declaration.

In addition to the above three arrays, the following data arrays are found in COMMON block /INPARM/.

EPSU(25), EXDIF(25), EYDIF(25), SOLNMX(25), KOUTIN(25), KOUTSL(25), KOUTNS(25).

These arrays are used for data associated with each variable. Thus we have the restriction MEQN $\leq$ 25. If this restriction is violated, then the user must increase the subscript range for these arrays in all the /INPARM/ declarations.

The control of output. The variables can be output at the end of a specified time step, or at certain specified times. There are three ways in which output can be obtained. First an increment in time, TIMINC, can be input. Then output will be obtained at TBGN + TININC, TBGN + 2.*TININC, TBGN + 3.*TININC, . . . The variables to be printed are determine the array KOUTIN(NWQ). If KOUTIN(M) $\neq$ 0, then the M^{th} variable will be output, where 1 $\leq$ M $\leq$ MEQN. If NUPRNT = 0, then none of the solution arrays will be printed. If NUPLOT = 0 none will be plotted.

If KSLMAX > 0, then variables will be output at the times TSLK(1), TSLK(2), . . . TSLK(KSLMAX) given by the array TSLK. Which variables are output is governed by the array values KOUTSL(M) $\neq$ 0. The value KSLMAX must satisfy KSLMAX $\leq$ 25 unless the COMMON statements are altered.

The input parameters. The input data can be in one of four formats. The first is NAMELIST input which is not standard Fortran and is not available on the NCAR system. The second is a short form of the input in which as many variables as possible are set by default. The variables are read by a standard formatted read, but they are placed on the card following the name, for example, XA=+Ø.00000, XB=1.00000... . We hope this makes it easier for the user to prepare the input. The third form is a long form which includes all input parameters, about 60 in number. This provides the experienced user with full flexibility. The fourth form allows a restart from a "checkpoint" or "save" tape. This form has not yet been debugged.

The input format is described in more detail on the comment cards in the program.

References

[1] W. Enright, R. Bedet, I. Farkas, T. Hull, "Test Results in Initial Value Methods for Non-stiff Ordinary Differential Equations", Tech. Report 68, Dept. Computer Science, University of Toronto (1974).

[2] J. Gary, "Boundary Conditions for the Method of Lines Applied to Hyperbolic Equations", Department of Computer Science Report, University of Colorado, Boulder, Colorado 80302 (1975).

[3] L. Shampine and M. Gordon, "Computer Solution of Ordinary Differential Equations", W.H. Greeman, San Francisco.

[4] J. Gary, "The Method of Lines Applied to a Simple Hyperbolic Equation", Submitted to Journ. Comp. Phys. (1975).