HYPACK: A SUBROUTINE FOR A HYPERBOLIC SYSTEM COUPLED WITH A SINGLE ELLIPTIC EQUATION.

by

John Gary
Department of Computer Science
University of Colorado at Boulder
Boulder, Colorado 80309

CU-CS-142-78 August, 1978

The work reported here was supported by NSF Grant #ATM75-17036.

1. Introduction

We will describe a Fortran subroutine package intended for the solution of an initial value problem possibly coupled with an elliptic equation. Examples from fluid mechanics are certain vorticity-stream function models, the Navier-Stokes equations for incompressible (or compressible) flow, and certain anelastic cloud models. An explicit method is used to solve the initial value problem, therefore the method is inefficient for parabolic or heavily dissipative problems. It is better for hyperbolic problems. A rectangular domain in cartesian coordinatnts is assumed. The user can choose a second or fourth order finite difference approximation over the two diminsional retangular domain for both marching and elliptic variables. A third order Runge-Kutta method with step size control is used for the integration of the marching equations in time. Several types of boundary conditions are permitted including periodic, symmetric, Dirichlet, Neumann or mixed. In addition boundary conditions based on characteristic variables for hyperbolic equations can be set up. The code can be run with arrays in central memory or in bulk memory (LCM or ECS). It can also be run with arrays on a disk, although data buffering is not ideal in this case. Spherical or cylindrical geometries cannot be solved with this package.

The elliptic equation is solved using a subroutine package developed by Paul Swartztrauber, Roland Sweet and John Adams at the National Center for Atmospheric Research.

The remainder of this users manual is a description of the type of problem which can be solved, along with a description of user supplied data and subroutines. Much of this information is also contained in comments in the code.

2. Statement of the mathematical problem.

The differential equations for the marching variables are assumed to be in the form

$$\frac{\partial u}{\partial t} = \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} + h$$

$$a(x)\frac{\partial^2 p}{\partial x^2} + b(x)\frac{\partial p}{\partial x} + c(x)p + d(y)\frac{\partial^2 p}{\partial y^2} + e(y)\frac{\partial p}{\partial y} + f(y)p = r$$

Here $u = u(x,y,t)$ is a vector of unknowns $u = (u_1, u_2, \ldots, u_m)$ and the elliptic unknown $p = p(x,y,t)$ is a scaler function. The functions on the right side of the first equation are vector functions of order m

$$f_i = f_i(x,y,t,w) \qquad 1 \le i \le m$$

$$g_i = g_i(x,y,t,w)$$

$$h_i = h_i(x,y,t,w,w_x,w_y,w_{xx},w_{xy},w_{yy},w_{x^4},w_{y^4})$$

Here w is the vector of unknowns $w = (u_1, u_2, \ldots u_m, p)$ and w_x, w_y, etc. denote partial derivatives $w_{xx} \equiv \partial^2 w/\partial x^2$, $w_{x^4} = \partial^4 w/\partial x^4$. The elliptic variable p and its equation may be absent, in which case we have an ordinary initial value problem.

In order to define the equation it would be sufficient to have only the function h; the functions f and g allow an alternate representation in "conservation" form. The use of conservation form will sometimes prevent nonlinear instability which can occur in certain hyperbolic equations. Therefore we allow an alternate representation of equations like the following (we drop the y-direction here). The equation $\partial u/\partial t = u\partial u/\partial x$ can be represented by $f(u) = 1/2\, u^2$ and $h \equiv o$, or $f \equiv o$ and $h(u,u_x) = uu_x$.

The domain must be rectangular in cartesian coordinants. Irregular domains, or spherical or polar coordinants whose domain contains the singularity at the pole, are not permitted. The user must input arrays x_i and y_i which define the mesh used for the finite difference approximation. The mesh spacing need not be uniform, thus $x_2 - x_1 \ne x_3 - x_2$ is allowed. The number of mesh points in each direction(NX and NY where $1 \le i \le NX$, and $1 \le j \le NY$) are input parameters.

Several types of boundary conditions can be used. The variables may be periodic in x, or y, or both. In the case of a periodic boundary in x with the second order difference scheme an additional point is added at the left and also the right side of the mesh. This is done so that $x_1-x_0 = x_{NX}-x_{NX-1}$ and $x_{NX+1}-x_{NX} = x_2-x_1$ where x_0 and x_{NX+1} are the additional points. The periodicity condition then yields $\underline{W}_0 - \underline{W}_{NX-1}$ for all variables. Using these values at the added points, the finite difference approximation to the derivatives can be computed at x_1 and x_{NX}. The fourth order scheme is treated in the same way. If the periodicity is used it must apply to all the variables u_1, u_2, ..., u_m, p.

A symmetry or skew-symmetry condition can also be used in x, or y, or both x and y. In this case the additional points satisfy the condition $x_1-x_0 = x_2-x_1$ and $x_{NX+1}-x_{NX} = x_{NX}-x_{NX-1}$. The values of the variables at the additional points are defined by $u_0 = u_2$ for symmetry and $u_0 = -u_2$ for skew symmetry. In the case of the elliptic variable p, the symmetry condition is transformed into the Neumann condition $\partial p/\partial n = 0$ where $\partial p/\partial n$ denotes the normal derivative and skew symmetry transforms into $p = 0$ at the boundary. This allows the use of the SEPELI elliptic equation solver. If the symmetry or skew symmetry condition is used one or the other must apply to all variables, each variables is symmetric, or skew symmetric. A Dirichlet type boundary, $u = g$, can be specified for any of the variables. Also a mixed type of condition $\partial u/\partial n+au = g$ can be specified. Of course, the latter is not proper for a hyperbolic equation, but it is reasonable for a parabolic or elliptic equation.

An additional type of boundary condition is intended for hyperbolic equations. To determine these boundary conditions it may be necessary to count the number of "incoming" characteristics. This gives the number of equations, say q, which can be specified at the boundary. Each of these equations usually involves a linear combination of the variables $(u_1,...u_m)$. The finite difference scheme requires an additional m-q equations at the boundary. These can be the differential equations for the m-q "outgoing" characteristic variables. Perhaps we can clarify this by using the example of the wave equation written as a system in one space dimension, namely

$$\frac{\partial u_1}{\partial t} = c\,\frac{\partial u_2}{\partial x} \qquad\qquad 0 \le x \le t$$

$$\frac{\partial u_2}{\partial t} = c\,\frac{\partial u_1}{\partial x}$$

The characteristic variables for this system are $w_1 = u_1+u_2$ and $w_2 = u_1-u_2$, and the characteristic equations are

$$\frac{\partial w_1}{\partial t} - c\,\frac{\partial w_1}{\partial x} = 0$$

$$\frac{\partial w_2}{\partial t} + c\,\frac{\partial w_2}{\partial x} = 0$$

These equations imply that w_2 should be specified at the left boundary, and w_1 at the right, that is

$$w_1(1,t) = g_1(t)$$
$$w_2(o,t) = g_o(t)$$

where $g_o(t)$ and $g_1(t)$ are boundary data. This gives the boundary conditions

$$u_1(1,t) + u_2(1,t) = g_1(t)$$
$$u_1(o,t) - u_2(o,t) = g_o(t)$$

for the original problem. These are not the only boundary conditions which yield a properly posed mathematical problem. For example, we might use

$$u_1(o,t) = g_o(t)$$
$$u_1(1,t) = g_1(t)$$

The mathematical problem is properly posed for the latter boundary conditions, but the finite difference scheme we have used, and all others that we are aware of, are unstable for this boundary condition. It is better to specify the incoming characteristic variables at each boundary. We will indicate how this can be done in the example in section 3.3 below.

3. User supplied input data.

In this section we will describe the input data which the user must supply in the form of arguments to the RKFPDE subroutine. These parameters are also listed in the program comments in the order in which they appear in the argument list. Here we will group the arguments by their functions.

3.1 The following parameters are used to define the differential equation.

NV - the total number of dependent variables including p, that is $(u_1,\ldots u_{NV-1},p)$ or $(u_1,\ldots u_{NV})$ are the dependent variables.

NVP - this input variable is NVP = 1 if the elliptic variable p is present, otherwise NVP = 0.

3.2 The mesh domain is defined by the following parameters.

NX - the number of mesh points in the x-direction. The number of intervals is NX -1. If the elliptic equation is present (NVP=1), then NX must satisfy NX $\geq$ 7, otherwise NX $\geq$ 3 for a second order scheme (NORDER =1) and NX $\geq$ 5 for a fourth order scheme (NORDER =2).

NY - the number of points in the y-direction. NY is subject to the same restrictions as NX.

XPTS(NX) - a real input array containing NX values giving the location of the mesh points in the x-direction. These points need not have a constant spacing, however, the conditions XPTS(I) < XPTS(I+1) must be satisfied.

YPTS(NY) - a real input array containing NY values giving the location of the mesh points in the y-direction. It is subject to restrictions similar to XPTS.

3.3 The following arguments define the difference scheme and boundary conditions.

NORDER - an integer input parameter. If NORDER = 1 a second order difference scheme for spatial derivatives based on a three point stencil is used. If NORDER = 2 a fourth order scheme based on a five point stencil is used. Near the boundary one sided finite differences may be required. This is discussed below in the section which describes the numerical method. If NORDER = 1 the fourth derivatives WX4 and WY4 can not be computed since a five point stencil is required.

NBTYP(NV,4) - an integer input array of dimension (NV,4). This array determines the type of boundary used on each side of the rectangular domain. The value of NBTYP(M,NSIDE) gives the boundary condition of the M-TH variable on the side NSIDE where $1 \le M \le NV$ and $1 \le NSIDE \le 4$. Here NSIDE = 1 for the right side, NSIDE = 2 for the top, NSIDE = 3 for the left side and NSIDE = 4 for the bottom side of the rectangular domian. The values of NBTYP range from 1 to 6 with the following meaning.

NBTYP = 1 for periodic boundary. If NBTYP(M,1) = 1 then the condition NBTYP(M,3) = 1 must also hold.

NBTYP = 2 for a symmetric boundary (see the discussion in section 2 for the precise meaning of symmetry). If $2 \le$ NBTYP(M,1) ≤ 3, then the condition $2 \le$ NBTYP(M,3) ≤ 3 must hold for $1 \le M \le NV$.

NBTYP = 3 for a skew symmetric boundary (see the discussion in section 2 and above for NBTYP = 2).

NBTYP = 4 for a Dirichlet type of boundary. If NBTYP(M,1) = 4, then the values of the M-TH variable along the right boundary at X = XPTS(NX) are given by the user supplied function GBDY, that is U(M,NX,J) = GBDY(T,XPTS(NX), YPTS(J), NSIDE,M) where NSIDE = 1 and $1 \le J \le NY$.

NBTYP = 5 for a mixed boundary condition. On the right or left boundary (NSIDE = 1 or 3) this condition is $\partial u/\partial x + a(t,x,y)u = g(t,x,y,)$ where $x = x_{NX}$ or $x = x_1$, and on the top or bottom (NSIDE = 2 or 4) it is $\partial u/\partial y + a(t,x,y)u = g(t,x,y)$ where $y = y_{NY}$ or $y = y_1$. The real functions $a(t,x,y)$ and $g(t,x,y)$ are supplied by the user as

```
FUNCTION ABDY(T,X,Y,NSIDE,M)
FUNCTION GBDY(T,X,Y,NSIDE,M)
```

The same functions are used for all four sides $1 \le NSIDE \le 4$ and all the variables $1 \le M \le NV$ including the elliptic variable if it is present. For the elliptic variable (M = NV) ABDY must not be dependent on X or Y. The values of T, X, Y, NSIDE, M are all input parameters to these functions. The values of X and Y will always be mesh point values XPTS(I) or YPTS(J) for $1 \le I \le NX$ or $1 \le J \le NY$.

NBTYP = 6 for a time derivative boundary condition. This condition is intended for hyperbolic equations. The motivation for its use is given in section 2. If NBTYP(M,1) = 6, then the time derivative of the M-TH variable on the right side will be computed from the differential equation using one sided differences. For example, consider the one dimensional problem $u_t + u_x = 0$. The right boundary at $x = x_{NX}$ is an outflow boundary and thus the value of u at the boundary must be obtained from the interior values thru the differential equation. If a second order spatial discretization is used, a method of lines approximation for interior points is

$$dU_i/dt + (U_{j+1}(t) - U_{j-1}(t))/2\Delta x = 0.$$

At the outflow boundary a one sided first order difference is used

$$dU_{NX}/dt + (U_{NX}(t) - U_{NX-1}(t))/\Delta x = 0$$

A first order approximation is used here for stability. At the left boundary the value $U_1(t)$ is specified by $U_1(t) = g_0(t)$. At this boundary we can use the condition NBTYP(1,3) = 4 (M = NV = 1 for this problem) and supply $g_0(t)$ thru the function GBDY. At the right boundary the condition NBTYP(1,1) = 6 is used. The NBTYP = 6 condition is used in conjunction with a user supplied subroutine called UTBDY. If NBTYP = 6 then the value of the time derivative of each marching variable on the given side is given to the UTBDY routine as an input parameter. The user routine can change the values of these time derivatives before they are used by the Runge-Kutta time integrator. In the case of the present simple example, the time derivative at the right side obtained from one sided differences should be returned unaltered. The routine UTBDY has no function in this case. The boundary condition at the left for this example could be NBTYP(1,3) = 6. In this case the UTBDY routine should return the time derivative of the boundary function $g_0(t)$ ($U_1(t) \equiv g_0(t)$). The calling sequence for UTBDY is

UTBDY(T,X,Y,NSIDE,NV,W,UT)

The input values $(U_1,U_2,\ldots U_{NV-1},P)$ or $(U_1,U_2,\ldots U_{NV})$ are stored in the array W. On input the array UT contains the approximations of the time derivatives $(U'_1,\ldots U'_{NVU-1})$, where NVU = NV - NVP, obtained from the differential equation using one-sided spatial differences. The UTBDY routine can correct these values and return the corrected values as the output in the array UT. In this example, the value UT(1) = dg_0/dt would be returned. (In this example, we have ignored any y-dependence). In this example the use of NBTYP(1,3) = 6 at the left boundary is an alternative to the use of NBTYP(1,3) = 4. Either will do. The use of NBTYP(1,1) =6 on the right (outflow) boundary is quite trivial in this example. The wave equation discussed in section 2 provides a less trivial example. Here the boundary condition at the right boundary is imposed on the characteristic variable

$$U(t,1,NX) + U(t,2,NX) = g_1(t)$$

Here U(t,1,NX) approximates $U_1(t,x_{NX})$ and we have again dropped the y-dependence. From the differential equation we have the time derivative of u_1 and u_2 which we can use to approximate the outflow characteristic u_1-u_2. These time derivatives are supplied to UTBDY in the input array UT. We thus have

$$U'(t,1,NX) - U'(t,2,NX) = UT(1) - UT(2)$$

The inflow characteristic equation can be differentiated to yield

$$U'(t,1,NX) + U'(t,2,NX) = g'_1(t)$$

If these two equations are combined we obtain the following code to correct UT.

```
UT1 = .5*(UT(1)-UT(2)+DG1(T))
UT2 = .5*(UT(2)-UT(1)+DG1(T))
UT(1) = UT1
UT(2) = UT2
```

Here we assume DG1(T) = dg_1/dt. This method introduces the boundary conditions imposed on the characteristic variables (U_1+U_2) and (U_1-U_2) into the differential equation written in terms of U_1 and U_2.

3.4 The following parameters control the time integration

T - on input, this parameter gives the starting value for the time integration. On output it contains the final value of the time, usually this final value is that of TOUT.

TOUT - an input parameter giving the end point of the time integration.

IFLAG - the value IFLAG = 1 on input indicates this is a new run and variables should be initialized. The value IFLAG = 2 on input indicates a continuation of a previous run. In this case various internal variables will not be initialized, instead the old values will be used. For example, the old time step can be used as the initial guess for the next time step, rather than making a new time step estimate independent of past history. If IFLAG = -1 on input, then only a single step will be taken regardless of the value of TOUT. On output the value of IFLAG will be 2 if the TOUT was reached, or will be -2 if IFLAG = -1 on input. If IFLAG > 2 on output then an error was detected by the RKFPDE package. In this case an error message will be printed by the package.

EPSABS - a real input array of dimension NVU (where NVU = NV-NVP is the number of marching variables). The array sets the absolute error tolerance used by the Runge-Kutta method. This method will choose the time step so that the estimated error in the M-TH variable per step (not per unit step) is bounded by

EPSREL(M) * $||U(M)||$ + EPSABS(M).

Here $1 \le M \le NVU = NV-NVP$.

EPSREL - a real input array of dimension NVU (where NVU = NV-NVP is the number of marching variables). The array sets the relative error tolerance used by the Runge-Kutta method.

3.5 The work arrays. The contents of these arrays should not be altered between a call of RKFPDE and a second call (with IFLAG = 2) which continues the integration started with the first call.

WORK - a real work array of dimension NDWK.

NDWK - an integer input parameter giving the dimension of the array WORK. An approximate minimum value for NDWK when NVP = 1 is (using ECS) 200 + (30+18*NV)*NX + (30+LOG2(NY))*NY + 2*NX*NY.
Without ECS increase this by 4*NV*NX*NY

IWORK - an integer input array of dimension NDIWK.

NDIWK - and integer input parameter giving the dimension of IWORK which must be at least 6*NV+24

3.6 Intermediate output. By setting the parameter NBUG, various intermediate results can be obtained. The parameter NBUG is declared in common block

```
COMMON  /RICOM/  MOUT,NBUG
```

This parameter is set in a DATA statement in the RKFPDE routine. On some systems this DATA statement should be moved to a BLOCK DATA subprogram. The intermediate data obtained from internal print statements for NBUG ≠ 0 was used to debug the HYPACK package. The only printout which is likely to be of much use to most users is obtained when NBUG = 2. This value causes the 'current' time step H = Δt and the normalized estimated error (maximum of estimated error relative to EPSREL(M)*$\|U(M)\|$ + EPSABS(M)) to be printed on each time step attempted.

The parameter MOUT is the device number on which diagnostic messages are printed.

4. User supplied subroutines.

These subroutines define the differential equation and obtain the results of the integration. There are ten of these subroutines.

Even if they are not used, a dummy version may have to be supplied to satisfy the loader. The user supplied subroutines are the following. Examples of their use are given later.

SUBROUTINE F(T,X,Y,NV,W,FM)

This subroutine defines the term $f_m(t,x,y,w)$ where $W = (U_1,\ldots U_{NV-1},P)$. T, x, and y are real input parameters, NV is an integer input, W is a real input array of dimension NV, and FM a real output array of dimension NV. The values f_m are returned as FM(M) where $1 \le M \le NVU$ and NVU = NV-NVP. The values T,X,Y,NV, and the vector W(M), $1 \le M \le NV$ are input. If this f_m term does not appear in the equation it is necessary to set FM(M) = 0 unless the logical array element ISWDR(5) is set to .FALSE. . This array is set by a data statement in the RKFPDE routine. Some computing time can be saved by setting the ISWDR elements .FALSE. for unused terms in the equation.

SUBROUTINE G(T,X,Y,NV,W,GM)

This subroutine defines the term $g_m(t,x,y,w)$ similar to subroutine F described above. If this term is not present in the equation, then GM(M) should be set to zero. In order to avoid calling G at all the internal array element ISWDR(10) can be set .FALSE. within the RKFPDE routine.

SUBROUTINE HU(T,X,Y,NV,W,WX,WY,WXX,WXY,WYY,WX4,WY4,HUM)

This routine returns the values of the term h_m as HUM(M). Here T,X, and Y are real input parameters, NV an integer input, W, WX, WY, WXX, WXY, WYY, WX4, and WY4, are real input arrays of dimension NV and HUM is a real output array of dimension NV. The partial derivatives of $W = (U_1,..U_{NV-1},P)$ are supplied as input. Here $WX = (\partial u_1/\partial x\ \partial u_2/\partial x,\ldots,\partial p/\partial x)$, and similarily for the remaining terms. The fourth derivatives are available only if a five point difference stencil is used (NORDER = 2). The fourth derivative approximations are intended for use in artificial viscosity terms. The element HUM(M) is the M-th component of the term $h(t,x,y,w,w_x,w_y,w_{xx},w_{xy},w_{yy},w_{x^4},w_{w^4})$. If one of the arguments $(w_x,w_y,w_{xx},w_{xy},w_{yy},w_{x^4},w_{y^4})$ is not required to compute h then the corresponding element in the array ISWDR can be set .FALSE. in the RKFPDE routine. These corresponding elements are (2,6,3,9,7,4,8). Resetting the ISWDR array is not necessary but it may reduce computational cost.

SUBROUTINE HP(T,X,Y,NV,W,WX,WY,WXX,WXY,WYY,WX4,WY4,HPM)

This routine computes the right side r of the elliptic equation for p. The input parameters (T,X,Y,NV,W,WX,WY,WXX,WXY,WYY,WX4, WY4) have the same meaning as in the HU routine. However, the output parameter HPM is a real scaler in this case. The logical array ISPDR can be used to avoid computing unused arguments of the HP routine is the same way that ISWDR was used for the HU routine. The elements (2,6,3,9,7,4,8) of ISPDR correspond to $(w_x,w_y,w_{xx},w_{xy},w_{yy},w_{x^4},w_{y^4})$. The ISPDR array is also set in the RKFPDE routine.

FUNCTION GBDY(T,X,Y,NSIDE,M)

This real function has real input parameters T, X, and Y and integer input parameters NSIDE and M. It returns the value of the right side g of the boundary condition $u = g$ or $u_n + au = g$ for the M-TH variable on side determined by NSIDE ($1 \leq NSIDE \leq 4$) at the point (X,Y) at time T. The point (X,Y) will be a boundary point on the mesh (XPTS(I),YPTS(J)). Here $1 \leq M \leq NV$. BEWARE. The elliptic equation solver may not work unless the second order finite difference matrix is diagonally dominant.

FUNCTION ABDY(T,X,Y,NSIDE,M)

This real function has the same input as the GBDY function. It returns the value of the coefficient a in the mixed boundary condition described in section 2, namely $\partial u/\partial n + a\ u = g$. It will only be called if the type of boundary condition is NBTYP(M,NSIDE) = 5. For the elliptic variable (M=NV), this coefficient must be independent of the coordinants (X,Y). It may depend on T.

SUBROUTINE UTBDY(T,X,Y,NSIDE,NV,W,UT)

This routine is called if NBTYP(M,NSIDE) = 6 for one the variables. Its use is described in section 2 with further examples in section 6, below. Normally, if NBTYP(M,NSIDE) = 6 for some M, then NBTYP = 6 for all M along that side, however, this is not necessary. The integer input parameters NSIDE and NV give the boundary side and the dimension of W. The variables T, X, and Y are real input parameters, giving the time and the spatial coordinates. W is a real input array containing the variables $(U_1,U_2,\ldots U_{NV-1},P)$ or $(U_1,\ldots U_{NV})$ if NVP = 0. On input the real array UT contains the approximation to the time derivatives $(U_1',\ldots,U'_{NVU})$ where NVU = NV-NVP. The array UT can be corrected as indicated in section 3 to account for characteristic boundary conditions for a hyperbolic equation.

SUBROUTINE UINIT(T,X,Y,NV,U)

This routine is used to supply the initial values of the marching variables $(U_1,\ldots U_{NVU})$ where NVU = NV-NVP. The real input parameters T, X, and Y give the time and spatial coordinants. The integer input NV is the dimension of the array U which is the same as the number of variables. The output array element U(M) gives the value of the M-th initial variable at T, X, and Y. If the elliptic variable is present (NVP=1), then the initial value of P (which is stored in memory as U(NV)) is not given by this routine, but is computed from the elliptic equation. This routine is required because the variables may be stored in bulk memory (ECS or LCM) rather than in a central memory array. Therefore there is no array U of dimension U(NV,NX,NY) which contains the solution in central memory. Hence this subroutine is required to set the initial values in bulk memory.

SUBROUTINE PCOFX(X,A,B,C)

This routine sets the coefficients A(X),B(X),C(X) of the elliptic equation for p, namely

$$A(x)P_{xx} + B(x)P_x + C(x)P + D(y)P_{yy} + E(y)P_y + F(y)P = r.$$

Here X is a real input parameter and A, B, and C are real output parameters.

BEWARE: The elliptic equation solver may not work unless the second order finite difference matrix is diagonally dominant.

SUBROUTINE PCOFY(Y,D,E,F)

This routine sets the coefficients D(Y), E(Y), F(Y) of the elliptic equation for p, namely

$$A(x)P_{xx} + B(x)P_x + C(x)P + D(y)P_{yy} + E(y)P_y + F(y)P = r$$

Here Y is a real input parameter and D, E, and F are real output parameters.

5. Obtaining the output from the integration.

As mentioned in the discussion of the UINIT routine, there may be no central memory array which contains the solution, instead the solution is stored in bulk memory (ECS, LCM, or disk). Therefore the solution can not be returned to the user as an array argument to the RKFPDE routine. Instead each variable in bulk memory can be written into a two dimensional central memory array containing NX*NY values. To obtain these values the user must call the GETW subroutine as described below.

SUBROUTINE GETW(NV,NX,NY,M,W,NXD,NYD,WORK,NDWK)

The integer input variables NV, NX, NY, and M give the number of variables, the number of mesh points in X and Y, and indicate that the M-th variable is to be output. The NX*NY values of the M-th variable are written into the real output array W which has dimension W(NXD,NYD). The integer input variables NXD and NYD define these dimensions for the GETW routine. The work array WORK and its dimension NDWK must be the same as supplied to the RKFPDE routine. If bulk memory is not used, then the solution is stored in the array WORK along with intermediate results.

6. Examples of the use of this package.

First we will consider a simple hyperbolic equation on the unit square $0 \le x,y, \le 1$.

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = f$$

We assume the values of u are specified for all $t \geq 0$ along the left and bottom sides of the unit square. If the initial conditions are $u(0,x,y) = \cos(x+y)$ and the boundary conditions along the bottom and left side are $u(t,x,y) = \cos(x+y-2t)$, then the solution of differential equation is $u(t,x,y) = \cos(x+y-2t)$. To obtain a solution for $0 \leq t \leq 1$, the input parameters might be

```
T = 0
TOUT = 1
IFLAG = 1
NX = 8
NY = 8
NV = 1
NVP = 0
EPSREL(1) = 0.001
EPSABS(1) = 0.001
NBTYP(1,NSIDE) = (6,6,6,6)                  1 ≤ NSIDE ≤ 4
NORDER = 2
XPTS(I) = (I-1)/7.
XPTS(J) = (J-1)/7.
```

The user supplied functions might be the following.

```
SUBROUTINE F(T,X,Y,NV,W,FM)
REAL W(NV), FM(NV)
FM(1) = -W(1)
RETURN
END
SUBROUTINE G(T,X,Y,NV,W,GM)
REAL W(NV), FM(NV)
GM(1) = -W(1)
RETURN
END
SUBROUTINE HU(T,X,Y,NV,W,WX,WY,WXX,WXY,WYY,WX4,WY4,HUM)
REAL W(NV), WX(NV), WY(NV), WXX(NV), WXY(NV), WYY(NV), WX4(NV),
     WY4(NV), HUM(NV)
HUM(1) = 0
RETURN
END
```

```
SUBROUTINE UTBDY(T,X,Y,NSIDE,NV,W,UT)
REAL W(NV), UT(NV)
IF (NSIDE .GE. 3) UT(1) = 2.*SIN(X+Y-2.*T)
RETURN
END
SUBROUTINE UINIT(T,X,Y,NV,U)
REAL U(NV)
U(1) = COS(X+Y-2*T)
RETURN
END
```

The routines GBDY, ABDY, HP, PCOFX, AND PCOFY are just dummy routines in this case. We could also use the boundary type NBTYP(1,NSIDE) = (6,6,4,4). In this case the GBDY and UTBDY routines would be the following. The UTBDY routine is only called on the outflow boundaries in this case, and it simply returns the UT value unaltered.

```
FUNCTION GBDY(T,X,Y,NSIDE,M)
GBDY = COS(X+Y-2.*T)
RETURN
END
SUBROUTINE UTBDY(T,X,Y,NSIDE,NV,W,UT)
REAL W(NV), UT(NV)
UT(1) = UT(1)
RETURN
END
```