D03RBF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

Note. This routine was introduced into the NAG Fortran Library at Mark 19 and may therefore not be available to all users of the NAG Fortran SMP Library.

1 Purpose

D03RBF integrates a system of linear or nonlinear, time-dependent partial differential equations (PDEs) in two space dimensions on a rectilinear domain. The method of lines is employed to reduce the PDEs to a system of ordinary differential equations (ODEs) which are solved using a backward differentiation formula (BDF) method. The resulting system of nonlinear equations is solved using a modified Newton method and a Bi-CGSTAB iterative linear solver with ILU preconditioning. Local uniform grid refinement is used to improve the accuracy of the solution. D03RBF originates from the VLUGR2 package [1] [2].

2 Specification

```
 SUBROUTINE D03RBF(NPDE, TS, TOUT, DT, TOLS, TOLT, INIDOM, PDEDEF,
1                  BNDARY, PDEIV, MONITR, OPTI, OPTR, RWK, LENRWK,
2                  IWK, LENIWK, LWK, LENLWK, ITRACE, IND, IFAIL)
 INTEGER           NPDE, OPTI(4), LENRWK, IWK(LENIWK), LENIWK,
1                  LENLWK, ITRACE, IND, IFAIL
 real              TS, TOUT, DT(3), TOLS, TOLT, OPTR(3,NPDE),
1                  RWK(LENRWK)
 LOGICAL           LWK(LENLWK)
 EXTERNAL          INIDOM, PDEDEF, BNDARY, PDEIV, MONITR
```

3 Description

D03RBF integrates the system of PDEs:

$$F_j(t,x,y,u,u_t,u_x,u_y,u_{xx},u_{xy},u_{yy})=0,\ \ j=1,2,\ldots,\mathrm{NPDE},\ \ (x,y)\in\Omega,\ \ t_0\le t\le t_{\mathrm{out}},\qquad (1)$$

where Ω is an arbitrary rectilinear domain, i.e., a domain bounded by perpendicular straight lines. If the domain is rectangular then it is recommended that D03RAF is used.

The vector u is the set of solution values

$$u(x,y,t)=[u_1(x,y,t),\ldots,u_{\mathrm{NPDE}}(x,y,t)]^T,$$

and u_t denotes partial differentiation with respect to t, and similarly for u_x etc.

The functions F_j must be supplied by the user in a subroutine PDEDEF. Similarly the initial values of the functions $u(x,y,t)$ for $(x,y)\in\Omega$ must be specified at $t=t_0$ in a subroutine PDEIV.

Note that whilst complete generality is offered by the master equations (1), D03RBF is not appropriate for all PDEs. In particular, hyperbolic systems should not be solved using this routine. Also, at least one component of u_t must appear in the system of PDEs.

The boundary conditions must be supplied by the user in a subroutine BNDARY in the form

$$G_j(t,x,y,u,u_t,u_x,u_y)=0\ \ j=1,2,\ldots,\mathrm{NPDE},\ \ (x,y)\in\partial\Omega,\ \ t_0\le t\le t_{\mathrm{out}}.\qquad (2)$$

The domain is covered by a uniform coarse base grid specified by the user, and nested finer uniform subgrids are subsequently created in regions with high spatial activity. The refinement is controlled using a space monitor which is computed from the current solution and a user-supplied space tolerance TOLS. A number of optional parameters, e.g., the maximum number of grid levels at any time, and some weighting factors, can be specified in the arrays OPTI and OPTR. Further details of the refinement strategy can be found in Section 8.

The system of PDEs and the boundary conditions are discretised in space on each grid using a standard second-order finite difference scheme (centred on the internal domain and one-sided at the boundaries), and the resulting system of ODEs is integrated in time using a second-order, two-step, implicit BDF method with variable step size. The time integration is controlled using a time monitor computed at each grid level from the current solution and a user-supplied time tolerance TOLT, and some further optional user-specified weighting factors held in OPTR (see Section 8 for details). The time monitor is used to compute a new step size, subject to restrictions on the size of the change between steps, and (optional) user-specified maximum and minimum step sizes held in DT. The step size is adjusted so that the remaining integration interval is an integer number times $\triangle t$. In this way a solution is obtained at $t = t_{\mathrm{out}}$.

A modified Newton method is used to solve the nonlinear equations arising from the time integration. The user may specify (in OPTI) the maximum number of Newton iterations to be attempted. A Jacobian matrix is calculated at the beginning of each time step. If the Newton process diverges or the maximum number of iterations is exceeded, a new Jacobian is calculated using the most recent iterates and the Newton process is restarted. If convergence is not achieved after the (optional) user-specified maximum number of new Jacobian evaluations, the time step is retried with $\triangle t = \triangle t/4$. The linear systems arising from the Newton iteration are solved using a Bi-CGSTAB iterative method, in combination with ILU preconditioning. The maximum number of iterations can be specified by the user in OPTI.

In order to define the base grid the user must first specify a virtual uniform rectangular grid which contains the entire base grid. The position of the virtual grid in physical (x, y) space is given by the (x, y) co-ordinates of its boundaries. The number of points n_x and n_y in the x and y directions must also be given, corresponding to the number of columns and rows respectively. This is sufficient to determine precisely the (x, y) co-ordinates of all virtual grid points. Each virtual grid point is then referred to by integer co-ordinates (v_x, v_y), where $(0, 0)$ corresponds to the lower-left corner and $(n_x - 1, n_y - 1)$ corresponds to the upper-right corner. v_x and v_y are also referred to as the virtual column and row indices respectively.

The base grid is then specified with respect to the virtual grid, with each base grid point coinciding with a virtual grid point. Each base grid point must be given an index, starting from 1, and incrementing row-wise from the leftmost point of the lowest row. Also, each base grid row must be numbered consecutively from the lowest row in the grid, so that row 1 contains grid point 1.

As an example, consider the domain consisting of the two separate squares shown in Figure 1. The left-hand diagram shows the virtual grid and its integer co-ordinates (i.e., its column and row indices), and the right-hand diagram shows the base grid point indices and the base row indices (in brackets).

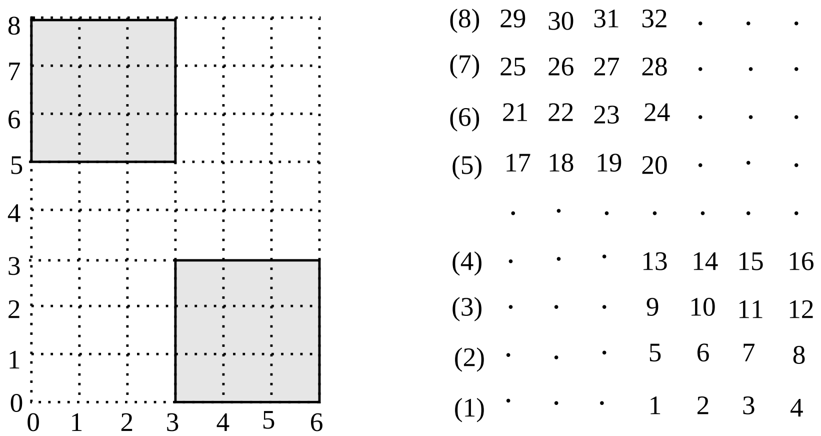

Figure 1

Hence the base grid point with index 6 say is in base row 2, virtual column 4, and virtual row 1, i.e., virtual grid integer co-ordinates (4,1); and the base grid point with index 19 say is in base row 5, virtual column 2, and virtual row 5, i.e., virtual grid integer co-ordinates (2,5).

The base grid must then be defined in the subroutine INIDOM by specifying the number of base grid rows, the number of base grid points, the number of boundaries, the number of boundary points, and the following integer arrays:

LROW contains the base grid indices of the starting points of the base grid rows.

IROW contains the virtual row numbers v_y of the base grid rows.

ICOL contains the virtual column numbers v_x of the base grid points.

LBND contains the grid indices of the boundary edges (without corners) and corner points.

LLBND contains the starting elements of the boundaries and corners in LBND.

Finally, ILBND contains the types of the boundaries and corners, as follows:

Boundaries:

1 – lower boundary

2 – left boundary

3 – upper boundary

4 – right boundary

External corners (90°):

12 – lower-left corner

23 – upper-left corner

34 – upper-right corner

41 – lower-right corner

Internal corners (270°):

21 – lower-left corner

32 – upper-left corner

43 – upper-right corner

14 – lower-right corner

Figure 2 shows the boundary types of a domain with a hole. Notice the logic behind the labelling of the corners: each one includes the types of the two adjacent boundary edges, in a clockwise fashion (outside the domain).

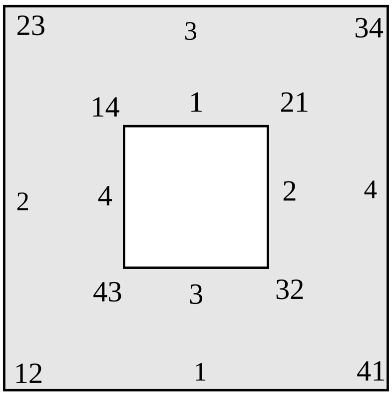

Figure 2

As an example, consider the domain shown in Figure 3. The left-hand diagram shows the physical domain and the right-hand diagram shows the base and virtual grids. The numbers outside the base grid are the indices of the left and rightmost base grid points, and the numbers inside the base grid are the boundary or corner numbers, indicating the order in which the boundaries are stored in LBND.

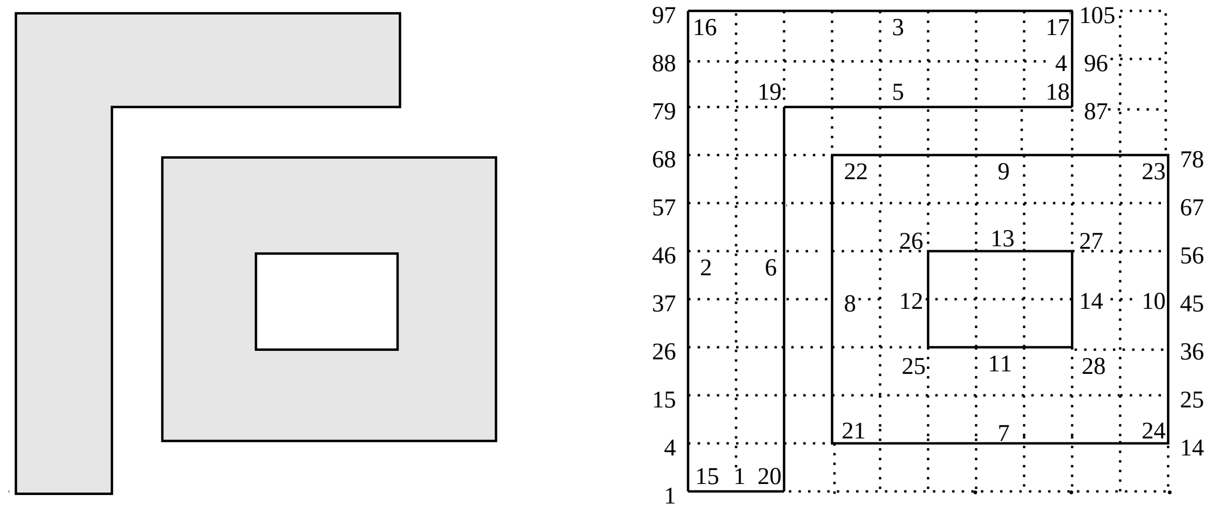

Figure 3

For this example we have

```
NROWS = 11
NPTS = 105
NBNDS = 28
NBPTS = 72

LROW = (1,4,15,26,37,46,57,68,79,88,97)

IROW = (0,1,2,3,4,5,6,7,8,9,10)

ICOL = (0,1,2,
        0,1,2,3,4,5,6,7,8,9,10,
        0,1,2,3,4,5,6,7,8,9,10,
        0,1,2,3,4,5,6,7,8,9,10,
        0,1,2,3,4,5,8,9,10,
        0,1,2,3,4,5,6,7,8,9,10,
        0,1,2,3,4,5,6,7,8,9,10,
        0,1,2,3,4,5,6,7,8,9,10,
        0,1,2,3,4,5,6,7,8,
        0,1,2,3,4,5,6,7,8,
        0,1,2,3,4,5,6,7,8)

LBND = (2,
        4,15,26,37,46,57,68,79,88,
        98,99,100,101,102,103,104,
        96,
        86,85,84,83,82,
        70,59,48,39,28,17,6,
        8,9,10,11,12,13,
        18,29,40,49,60,
        72,73,74,75,76,77,
        67,56,45,36,25,
        33,32,
        42,
        52,53,
        43,
        1,97,105,87,81,3,7,71,78,14,31,51,54,34)
```

```
LLBND = (1,2,11,18,19,24,31,37,42,48,53,55,56,58,59,60,
         61,62,63,64,65,66,67,68,69,70,71,72)

ILBND = (1,2,3,4,1,4,1,2,3,4,3,4,1,2,12,23,34,41,14,41,12,23,34,41,43,14,21,32)
```

This particular domain is used in Example 1 in Section 9.1, and data statements are used to define the above arrays in that example program. In Example 2 a less complicated domain is used, and in that case it is simpler to assign the values of the arrays in do-loops. This also allows flexibility in the number of base grid points.

The routine D03RYF can be called from INIDOM to obtain a simple graphical representation of the base grid, and to verify the data that the user has specified in INIDOM.

Subgrids are stored internally using the same data structure, and solution information is communicated to the user in the subroutines PDEIV, PDEDEF and BNDARY in arrays according to the grid index on the particular level, e.g., X(i) and Y(i) contain the (x, y) co-ordinates of grid point i, and U(i,j) contains the jth solution component u_j at grid point i.

The grid data and the solutions at all grid levels are stored in the workspace arrays, along with other information needed for a restart (i.e., a continuation call). It is not intended that the user extracts the solution from these arrays, indeed the necessary information regarding these arrays is not provided. The user-supplied monitor routine MONITR should be used to obtain the solution at particular levels and times. MONITR is called at the end of every time step, with the last step being identified via the input argument TLAST. The routine D03RZF should be called from MONITR to obtain grid information at a particular level.

Further details of the underlying algorithm can be found in Section 8 and in [1] and [2] and the references therein.

4 References

[1] Blom J G and Verwer J G (1993) VLUGR2: A vectorized local uniform grid refinement code for PDEs in 2D *Report NM–R9306* CWI, Amsterdam

[2] Blom J G, Trompert R A and Verwer J G (1996) Algorithm 758. VLUGR2: A vectorizable adaptive grid solver for PDEs in 2D *Trans. Math. Software* **22** 302–328

[3] Trompert R A and Verwer J G (1993) Analysis of the implicit Euler local uniform grid refinement method *SIAM J. Sci. Comput.* **14** 259–278

[4] Trompert R A (1993) Local uniform grid refinement and systems of coupled partial differential equations *Appl. Numer. Maths* **12** 331–355

5 Parameters

1: NPDE — INTEGER *Input*

On entry: the number of PDEs in the system.

Constraint: NPDE $\geq$ 1.

2: TS — ***real*** *Input/Output*

On entry: the initial value of the independent variable t.

On exit: the value of t which has been reached. Normally TS = TOUT.

Constraint: TS < TOUT.

3: TOUT — ***real*** *Input*

On entry: the final value of t to which the integration is to be carried out.

4: DT(3) — ***real*** array *Input/Output*

On entry: the initial, minimum and maximum time step sizes respectively. DT(1) specifies the initial time step size to be used on the first entry, i.e., when IND = 0. If DT(1) = 0.0 then the default value DT(1) = 0.01 × (TOUT−TS) is used. On subsequent entries (IND = 1), the value of DT(1) is not referenced.

DT(2) specifies the minimum time step size to be attempted by the integrator. If DT(2) = 0.0 the default value DT(2) = 10.0 × ***machine precision*** is used.

DT(3) specifies the maximum time step size to be attempted by the integrator. If DT(3) = 0.0 the default value DT(3) = TOUT − TS is used.

On exit: DT(1) contains the time step size for the next time step. DT(2) and DT(3) are unchanged or set to their default values if zero on entry.

Constraints: if IND = 1 then DT(1) is unconstrained. Otherwise DT(1) ≥ 0 and if DT(1) > 0.0 then it must satisfy the constraints:

$$10.0 \times \textbf{\textit{machine precision}} \times \max(|\mathrm{TS}|,|\mathrm{TOUT}|) \le \mathrm{DT}(1) \le \mathrm{TOUT} - \mathrm{TS}$$
$$\mathrm{DT}(2) \le \mathrm{DT}(1) \le \mathrm{DT}(3)$$

where the values of DT(2) and DT(3) will have been reset to their default values if zero on entry.

DT(2) and DT(3) must satisfy DT(i) ≥ 0, i = 2,3 and DT(2) ≤ DT(3) for IND = 0 and IND = 1.

5: TOLS — ***real*** *Input*

On entry: the space tolerance used in the grid refinement strategy (σ in equation (4)). See Section 8.2.

Constraint: TOLS > 0.0.

6: TOLT — ***real*** *Input*

On entry: the time tolerance used to determine the time step size (τ in equation (7)). See Section 8.3.

Constraint: TOLT > 0.0.

7: INIDOM — SUBROUTINE, supplied by the user. *External Procedure*

INIDOM must specify the base grid in terms of the data structure described in Section 3. INIDOM is not referenced if, on entry, IND = 1. D03RYF can be called from INIDOM to obtain a simple graphical representation of the base grid, and to verify the data that the user has specified in INIDOM. D03RBF also checks the validity of the data, but the user is strongly advised to call D03RYF to ensure that the base grid is exactly as required.

Note. The boundaries of the base grid should consist of as many points as are necessary to employ second-order space discretization, i.e.,, a boundary enclosing the internal part of the domain must include at least 3 grid points including the corners. If Neumann boundary conditions are to be applied the minimum is 4.

Its specification is:

```
      SUBROUTINE INIDOM(MAXPTS, XMIN, XMAX, YMIN, YMAX, NX, NY, NPTS,
     1                  NROWS, NBNDS, NBPTS, LROW, IROW, ICOL, LLBND,
     2                  ILBND, LBND, IERR)
      INTEGER           MAXPTS, NX, NY, NPTS, NROWS, NBNDS, NBPTS,
     1                  LROW(*), IROW(*), ICOL(*), LLBND(*), ILBND(*),
     2                  LBND(*), IERR
      real              XMIN, XMAX, YMIN, YMAX
```

1: MAXPTS — INTEGER *Input*

On entry: the maximum number of base grid points allowed by the available workspace.

2: XMIN — ***real*** *Output*
3: XMAX — ***real*** *Output*

On exit: the extents of the virtual grid in the x-direction, i.e., the x co-ordinates of the left and right boundaries respectively.

Constraints: XMIN $<$ XMAX and XMAX must be sufficiently distinguishable from XMIN for the precision of the machine being used.

4: YMIN — ***real*** *Output*
5: YMAX — ***real*** *Output*

On exit: the extents of the virtual grid in the y-direction, i.e., the y co-ordinates of the left and right boundaries respectively.

Constraints: YMIN $<$ YMAX and YMAX must be sufficiently distinguishable from YMIN for the precision of the machine being used.

6: NX — INTEGER *Output*
7: NY — INTEGER *Output*

On exit: the number of virtual grid points in the x- and y-direction respectively (including the boundary points).

Constraints: NX and NY ≥ 4.

8: NPTS — INTEGER *Output*

On exit: the total number of points in the base grid. If the required number of points is greater than MAXPTS then INIDOM must be exited immediately with IERR set to -1 to avoid overwriting memory.

Constraints: NPTS $\leq$ NX $\times$ NY and if IERR $\neq -1$ on exit, NPTS $\leq$ MAXPTS.

9: NROWS — INTEGER *Output*

On exit: the total number of rows of the virtual grid that contain base grid points. This is the maximum base row index.

Constraint: $4 \leq$ NROWS $\leq$ NY.

10: NBNDS — INTEGER *Output*

On exit: the total number of physical boundaries and corners in the base grid.

Constraint: NBNDS ≥ 8.

11: NBPTS — INTEGER *Output*

On exit: the total number of boundary points in the base grid.

Constraint: $12 \leq$ NBPTS $<$ NPTS.

12: LROW($*$) — INTEGER array *Output*

On exit: LROW(i) for $i = 1, 2, \ldots$,NROWS must contain the base grid index of the first grid point in base grid row i.

Constraints: $1 \leq$ LROW(i) $\leq$ NPTS for $i = 1, 2, \ldots$,NROWS,
LROW($i-1$) $<$ LROW(i) for $i = 2, 3, \ldots$,NROWS.

13: IROW($*$) — INTEGER array *Output*

On exit: IROW(i) for $i = 1, 2, \ldots$,NROWS must contain the virtual row number v_y that corresponds to base grid row i.

Constraints: $0 \leq$ IROW(i) $\leq$ NY for $i = 1, 2, \ldots$,NROWS,
IROW($i-1$) $<$ IROW(i) for $i = 2, 3, \ldots$,NROWS.

14: ICOL($*$) — INTEGER array *Output*

On exit: ICOL(i) for $i = 1, 2, \ldots$,NPTS must contain the virtual column number v_x that contains base grid point i.

Constraint: $0 \leq$ ICOL(i) $\leq$ NX for $i = 1, 2, \ldots$,NPTS.

15: LLBND(∗) — INTEGER array *Output*

On exit: LLBND(i) for $i = 1, 2, \ldots$,NBNDS must contain the element of LBND corresponding to the start of the ith boundary or corner.

Note. The order of the boundaries and corners in LLBND must be first all the boundaries and then all the corners. The end points of a boundary (i.e., the adjacent corner points) must **not** be included in the list of points on that boundary. Also, if a corner is shared by two pairs of physical boundaries then it has two types and must therefore be treated as two corners.

Constraints: $1 \leq$ LLBND(i) $\leq$ NBPTS for $i = 1, 2, \ldots$,NBNDS,
LLBND($i - 1$) $<$ LLBND(i) for $i = 2, 3, \ldots$,NBNDS.

16: ILBND(∗) — INTEGER array *Output*

On exit: ILBND(i) for $i = 1, 2, \ldots$,NBNDS must contain the type of the ith boundary (or corner), as given in Section 3.

Constraint: ILBND(i) must be equal to one of the following: 1, 2, 3, 4, 12, 23, 34, 41, 21, 32, 43 or 14, for $i = 1, 2, \ldots$,NBNDS.

17: LBND(∗) — INTEGER array *Output*

On exit: LBND(i) for $i = 1, 2, \ldots$,NBPTS must contain the grid index of the ith boundary point. The order of the boundaries is as specified in LLBND, but within this restriction the order of the points in LBND is arbitrary.

Constraint: $1 \leq$ LBND(i) $\leq$ NPTS for $i = 1, 2, \ldots$,NBPTS.

18: IERR — INTEGER *Output*

On exit: if the required number of grid points is larger than MAXPTS, IERR must be set to -1 to force a termination of the integration and an immediate return to the calling program with IFAIL set to 3. Otherwise, IERR should remain unchanged.

INIDOM must be declared as EXTERNAL in the (sub)program from which D03RBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

8: PDEDEF — SUBROUTINE, supplied by the user. *External Procedure*

PDEDEF must evaluate the functions F_j, $j = 1, 2, \ldots$,NPDE, in equation (1) which define the system of PDEs (i.e., the residuals of the resulting ODE system) at all interior points of the domain. Values at points on the boundaries of the domain are ignored and will be overwritten by the subroutine BNDARY. PDEDEF is called for each subgrid in turn.

Its specification is:

```
      SUBROUTINE PDEDEF(NPTS, NPDE, T, X, Y, U, UT, UX, UY, UXX, UXY,
     1                  UYY, RES)
      INTEGER           NPTS, NPDE
      real              T, X(NPTS), Y(NPTS), U(NPTS,NPDE),
     1                  UT(NPTS,NPDE), UX(NPTS,NPDE), UY(NPTS,NPDE),
     2                  UXX(NPTS,NPDE), UXY(NPTS,NPDE), UYY(NPTS,NPDE),
     3                  RES(NPTS,NPDE)
```

1: NPTS — INTEGER *Input*

On entry: the number of grid points in the current grid.

2: NPDE — INTEGER *Input*

On entry: the number of PDEs in the system.

3: T — ***real*** *Input*

On entry: the current value of the independent variable t.

4: X(NPTS) — ***real*** array *Input*

On entry: X(i) contains the x co-ordinate of the ith grid point, for $i = 1, 2, \ldots,$NPTS.

5: Y(NPTS) — ***real*** array *Input*

On entry: Y(i) contains the y co-ordinate of the ith grid point, for $i = 1, 2, \ldots,$NPTS.

6: U(NPTS,NPDE) — ***real*** array *Input*

On entry: U(i,j) contains the value of the jth PDE component at the ith grid point, for $i = 1, 2, \ldots,$NPTS, $j = 1, 2, \ldots,$NPDE.

7: UT(NPTS,NPDE) — ***real*** array *Input*

On entry: UT(i,j) contains the value of $\partial u/\partial t$ for the jth PDE component at the ith grid point, for $i = 1, 2, \ldots,$NPTS, $j = 1, 2, \ldots,$NPDE.

8: UX(NPTS,NPDE) — ***real*** array *Input*

On entry: UX(i,j) contains the value of $\partial u/\partial x$ for the jth PDE component at the ith grid point, for $i = 1, 2, \ldots,$NPTS, $j = 1, 2, \ldots,$NPDE.

9: UY(NPTS,NPDE) — ***real*** array *Input*

On entry: UY(i,j) contains the value of $\partial u/\partial y$ for the jth PDE component at the ith grid point, for $i = 1, 2, \ldots,$NPTS, $j = 1, 2, \ldots,$NPDE.

10: UXX(NPTS,NPDE) — ***real*** array *Input*

On entry: UXX(i,j) contains the value of $\partial^2 u/\partial x^2$ for the jth PDE component at the ith grid point, for $i = 1, 2, \ldots,$NPTS, $j = 1, 2, \ldots,$NPDE.

11: UXY(NPTS,NPDE) — ***real*** array *Input*

On entry: UXY(i,j) contains the value of $\partial^2 u/\partial x \partial y$ for the jth PDE component at the ith grid point, for $i = 1, 2, \ldots,$NPTS, $j = 1, 2, \ldots,$NPDE.

12: UYY(NPTS,NPDE) — ***real*** array *Input*

On entry: UYY(i,j) contains the value of $\partial^2 u/\partial y^2$ for the jth PDE component at the ith grid point, for $i = 1, 2, \ldots,$NPTS, $j = 1, 2, \ldots,$NPDE.

13: RES(NPTS,NPDE) — ***real*** array *Output*

On exit: RES(i,j) must contain the value of F_j for $j = 1, 2, \ldots,$NPDE, at the ith grid point for $i = 1, 2, \ldots,$NPTS, although the residuals at boundary points will be ignored (and overwritten later on) and so they need not be specified here.

PDEDEF must be declared as EXTERNAL in the (sub)program from which D03RBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

9: BNDARY — SUBROUTINE, supplied by the user. *External Procedure*

BNDARY must evaluate the functions G_j, $j = 1, 2, \ldots,$NPDE, in equation (2) which define the boundary conditions at all boundary points of the domain. Residuals at interior points must **not** be altered by this subroutine.

Its specification is:

```
      SUBROUTINE BNDARY(NPTS, NPDE, T, X, Y, U, UT, UX, UY, NBNDS,
     1                  NBPTS, LLBND, ILBND, LBND, RES)
      INTEGER           NPTS, NPDE, NBNDS, NBPTS, LLBND(NBNDS),
     1                  ILBND(NBNDS), LBND(NBPTS)
      real              T, X(NPTS), Y(NPTS), U(NPTS,NPDE),
     1                  UT(NPTS,NPDE), UX(NPTS,NPDE), UY(NPTS,NPDE),
     2                  RES(NPTS,NPDE)
```

1: NPTS — INTEGER *Input*

On entry: the number of grid points in the current grid.

2: NPDE — INTEGER *Input*

On entry: the number of PDEs in the system.

3: T — ***real*** *Input*

On entry: the current value of the independent variable t.

4: X(NPTS) — ***real*** array *Input*

On entry: X(i) contains the x co-ordinate of the ith grid point, for $i = 1, 2, \ldots,$NPTS.

5: Y(NPTS) — ***real*** array *Input*

On entry: Y(i) contains the y co-ordinate of the ith grid point, for $i = 1, 2, \ldots,$NPTS.

6: U(NPTS,NPDE) — ***real*** array *Input*

On entry: U(i,j) contains the value of the jth PDE component at the ith grid point, for $i = 1, 2, \ldots,$NPTS, $j = 1, 2, \ldots,$NPDE.

7: UT(NPTS,NPDE) — ***real*** array *Input*

On entry: UT(i,j) contains the value of $\partial u/\partial t$ for the jth PDE component at the ith grid point, for $i = 1, 2, \ldots,$NPTS, $j = 1, 2, \ldots,$NPDE.

8: UX(NPTS,NPDE) — ***real*** array *Input*

On entry: UX(i,j) contains the value of $\partial u/\partial x$ for the jth PDE component at the ith grid point, for $i = 1, 2, \ldots,$NPTS, $j = 1, 2, \ldots,$NPDE.

9: UY(NPTS,NPDE) — ***real*** array *Input*

On entry: UY(i,j) contains the value of $\partial u/\partial y$ for the jth PDE component at the ith grid point, for $i = 1, 2, \ldots,$NPTS, $j = 1, 2, \ldots,$NPDE.

10: NBNDS — INTEGER *Input*

On entry: the total number of physical boundaries and corners in the grid.

11: NBPTS — INTEGER *Input*

On entry: the total number of boundary points in the grid.

12: LLBND(NBNDS) — INTEGER array *Input*

On entry: LLBND(i) for $i = 1, 2, \ldots,$NBNDS contains the element of LBND corresponding to the start of the ith boundary (or corner).

13: ILBND(NBNDS) — INTEGER array *Input*

On entry: ILBND(i) for $i = 1, 2, \ldots,$NBNDS contains the type of the ith boundary, as given in Section 3.

14: LBND(NBPTS) — INTEGER array *Input*

On entry: LBND(i) for $i = 1, 2, \ldots,$NBPTS contains the grid index of the ith boundary point, where the order of the boundaries is as specified in LLBND. Hence the ith boundary point has co-ordinates X(LBND(i)) and Y(LBND(i)), and the corresponding solution values are U(LBND(i),j), $j = 1, 2, \ldots,$NPDE.

15: RES(NPTS,NPDE) — ***real*** array *Output*

On exit: RES(LBND(i),j) must contain the value of G_j for $j = 1, 2, \ldots,$NPDE, at the ith boundary point for $i = 1, 2, \ldots,$NBPTS.

Note. Elements of RES corresponding to interior points, i.e., points not included in LBND, must **not** be altered.

BNDARY must be declared as EXTERNAL in the (sub)program from which D03RBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

10: PDEIV — SUBROUTINE, supplied by the user. *External Procedure*

PDEIV must specify the initial values of the PDE components u at all points in the base grid. PDEIV is not referenced if, on entry, IND = 1.

Its specification is:

```
SUBROUTINE PDEIV(NPTS, NPDE, T, X, Y, U)
INTEGER          NPTS, NPDE
real             T, X(NPTS), Y(NPTS), U(NPTS,NPDE)
```

1: NPTS — INTEGER *Input*

On entry: the number of grid points in the base grid.

2: NPDE — INTEGER *Input*

On entry: the number of PDEs in the system.

3: T — ***real*** *Input*

On entry: the (initial) value of the independent variable t.

4: X(NPTS) — ***real*** array *Input*

On entry: X(i) contains the x co-ordinate of the ith grid point, for $i = 1, 2, \ldots$,NPTS.

5: Y(NPTS) — ***real*** array *Input*

On entry: Y(i) contains the y co-ordinate of the ith grid point, for $i = 1, 2, \ldots$,NPTS.

6: U(NPTS,NPDE) — ***real*** array *Output*

On exit: U(i,j) must contain the value of the jth PDE component at the ith grid point, for $i = 1, 2, \ldots$,NPTS, $j = 1, 2, \ldots$,NPDE.

PDEIV must be declared as EXTERNAL in the (sub)program from which D03RBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

11: MONITR — SUBROUTINE, supplied by the user. *External Procedure*

MONITR is called by D03RBF at the end of every successful time step, and may be used to examine or print the solution or perform other tasks such as error calculations, particularly at the final time step, indicated by the parameter TLAST.

The input arguments contain information about the grid and solution at all grid levels used. D03RZF should be called from MONITR in order to extract the number of points and their (x, y) co-ordinates on a particular grid.

MONITR can also be used to force an immediate tidy termination of the solution process and return to the calling program.

Its specification is:

```
      SUBROUTINE MONITR(NPDE, T, DT, DTNEW, TLAST, NLEV, XMIN, YMIN,
     1                  DXB, DYB, LGRID, ISTRUC, LSOL, SOL, IERR)
      INTEGER           NPDE, NLEV, LGRID(*), ISTRUC(*), LSOL(NLEV), IERR
      real              T, DT, DTNEW, XMIN, YMIN, DXB, DYB, SOL(*)
      LOGICAL           TLAST
```

1: NPDE — INTEGER *Input*

On entry: the number of PDEs in the system.

2: T — ***real*** *Input*

On entry: the current value of the independent variable t, i.e., the time at the end of the integration step just completed.

3: DT — ***real*** *Input*

On entry: the current time step size DT, i.e., the time step size used for the integration step just completed.

4: DTNEW — ***real*** *Input*

On entry: the time step size that will be used for the next time step.

5: TLAST — LOGICAL *Input*

On entry: indicates if intermediate or final time step. TLAST = .FALSE. for an intermediate step, TLAST = .TRUE. for the last call to MONITR before returning to the user's program.

6: NLEV — INTEGER *Input*

On entry: the number of grid levels used at time T.

7: XMIN — ***real*** *Input*
8: YMIN — ***real*** *Input*

On entry: the (x, y) co-ordinates of the lower-left corner of the virtual grid.

9: DXB — ***real*** *Input*
10: DYB — ***real*** *Input*

On entry: the sizes of the base grid spacing in the x- and y-direction respectively.

11: LGRID($*$) — INTEGER array *Input*

On entry: LGRID contains pointers to the start of the grid structures in ISTRUC, and must be passed unchanged to D03RZF in order to extract the grid information.

12: ISTRUC($*$) — INTEGER array *Input*

On entry: ISTRUC contains the grid structures for each grid level and must be passed unchanged to D03RZF in order to extract the grid information.

13: LSOL(NLEV) — INTEGER array *Input*

On entry: LSOL(l) contains the pointer to the solution in SOL at grid level l and time T. (LSOL(l) actually contains the array index immediately preceding the start of the solution in SOL. See below.)

14: SOL($*$) — ***real*** array *Input*

On entry: SOL contains the solution u at time T for each grid level l in turn, positioned according to LSOL. More precisely

$$\mathrm{U}(i,j) = \mathrm{SOL}(\mathrm{LSOL}(l) + (j-1) \times n_l + i)$$

represents the jth component of the solution at the ith grid point in the lth level, for $i = 1, \ldots, n_l$, $j = 1, \ldots,$NPDE, $l = 1, \ldots,$NLEV, where n_l is the number of grid points at level l (obtainable by a call to D03RZF).

15: IERR — INTEGER *Output*

On exit: IERR should be set to 1 to force a termination of the integration and an immediate return to the calling program with IFAIL set to 4. IERR should remain unchanged otherwise.

MONITR must be declared as EXTERNAL in the (sub)program from which D03RBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

12: OPTI(4) — INTEGER array *Input*

On entry: OPTI may be set to control various options available in the integrator. If OPTI(1) = 0 then **all** the default options are employed.

If OPTI(1) > 0 then the default value of OPTI(i) for $i = 2, 3, 4$, can be obtained by setting OPTI(i) $= 0$.

OPTI(1) specifies the maximum number of grid levels allowed (including the base grid). OPTI(1) ≥ 0. The default value is OPTI(1) $= 3$.

OPTI(2) specifies the maximum number of Jacobian evaluations allowed during each nonlinear equations solution. OPTI(2) ≥ 0. The default value is OPTI(2) $= 2$.

OPTI(3) specifies the maximum number of Newton iterations in each nonlinear equations solution. OPTI(3) ≥ 0. The default value is OPTI(3) $= 10$.

OPTI(4) specifies the maximum number of iterations in each linear equations solution. OPTI(4) ≥ 0. The default value is OPTI(4) $= 100$.

Constraint: OPTI(1) ≥ 0 and if OPTI(1) > 0 then OPTI(i) ≥ 0 for i = 2,3,4.

13: OPTR(3,NPDE) — ***real*** array *Input*

On entry: OPTR may be used to specify the optional vectors u^{max}, w^s and w^t in the space and time monitors (see Section 8).

If an optional vector is not required then all its components should be set to 1.0.

OPTR(1,j), for $j = 1, 2, \ldots$,NPDE, specifies u_j^{max}, the approximate maximum absolute value of the jth component of u, as used in (4) and (7). OPTR(1,j) > 0.0 for $j = 1, 2, \ldots$,NPDE.

OPTR(2,j), for $j = 1, 2, \ldots$,NPDE, specifies w_j^s, the weighting factors used in the space monitor (see (4)) to indicate the relative importance of the jth component of u on the space monitor. OPTR(2,j) ≥ 0.0 for $j = 1, 2, \ldots$,NPDE.

OPTR(3,j), for $j = 1, 2, \ldots$,NPDE, specifies w_j^t, the weighting factors used in the time monitor (see (6)) to indicate the relative importance of the jth component of u on the time monitor. OPTR(3,j) ≥ 0.0 for $j = 1, 2, \ldots$,NPDE.

Constraint: OPTR(1,j) > 0.0 for $j = 1, 2, \ldots$,NPDE and OPTR(i, j) ≥ 0.0 for $i = 2, 3$ and $j = 1, 2, \ldots$,NPDE.

14: RWK(LENRWK) — ***real*** array *Workspace*

15: LENRWK — INTEGER *Input*

On entry: the dimension of the array RWK as declared in the (sub)program from which D03RBF is called.

The required value of LENRWK can not be determined exactly in advance, but a suggested value is

LENRWK = MAXPTS $\times$ NPDE $\times$ ($5\times l$+18$\times$NPDE+9) + 2 $\times$ MAXPTS,

where l = OPTI(1) if OPTI(1) $\neq 0$ and $l = 3$ otherwise, and MAXPTS is the expected maximum number of grid points at any one level. If during the execution the supplied value is found to be too small then the routine returns with IFAIL = 3 and an estimated required size is printed on the current error message unit (see X04AAF).

Note. The size of LENRWK can not be checked upon initial entry to D03RBF since the number of grid points on the base grid is not known.

16: IWK(LENIWK) — INTEGER array *Input/Output*

On entry: if IND = 0, IWK need not be set. Otherwise IWK must remain unchanged from a previous call to D03RBF.

On exit: the following components of the array IW concern the efficiency of the integration.

IWK(1) contains the number of steps taken in time;

IWK(2) contains the number of rejected time steps;

IWK(2+l) contains the total number of residual evaluations performed (i.e., the number of times PDEDEF was called) at grid level l;

IWK(2+$m+l$) contains the total number of Jacobian evaluations performed at grid level l;

IWK(2+2$\times m+l$) contains the total number of Newton iterations performed at grid level l;

IWK(2+3$\times m+l$) contains the total number of linear solver iterations performed at grid level l;

IWK(2+4$\times m+l$) contains the maximum number of Newton iterations performed at any one time step at grid level l;

IWK(2+5$\times m+l$) contains the maximum number of linear solver iterations performed at any one time step at grid level l;

for $l = 1, 2, \ldots, nl$, where nl is the number of levels used and m = OPTI(1) if OPTI(1) $>$ 0 and m = 3 otherwise.

Note. The total and maximum numbers are cumulative over all calls to D03RBF. If the specified maximum number of Newton or linear solver iterations is exceeded at any stage, then the maximums above are set to the specified maximum plus one.

17: LENIWK — INTEGER *Input*

On entry: the dimension of the array IWK as declared in the (sub)program from which D03RBF is called.

The required value of LENIWK can not be determined exactly in advance, but a suggested value is

LENIWK = MAXPTS $\times$ (14+5$\times m$) + 7 $\times$ m + 2,

where MAXPTS is the expected maximum number of grid points at any one level and m =OPTI(1) if OPTI(1) $>$ 0 and m = 3 otherwise. If during the execution the supplied value is found to be too small then the routine returns with IFAIL = 3 and an estimated required size is printed on the current error message unit (see X04AAF).

Note. The size of LENIWK can not be checked upon initial entry to D03RBF since the number of grid points on the base grid is not known.

18: LWK(LENLWK) — LOGICAL array *Workspace*

19: LENLWK — INTEGER *Input*

On entry: the dimension of the array LWK as declared in the (sub)program from which D03RBF is called.

The required value of LENLWK can not be determined exactly in advance, but a suggested value is

LENLWK = MAXPTS + 1,

where MAXPTS is the expected maximum number of grid points at any one level. If during the execution the supplied value is found to be too small then the routine returns with IFAIL = 3 and an estimated required size is printed on the current error message unit (see X04AAF).

Note. The size of LENLWK can not be checked upon initial entry to D03RBF since the number of grid points on the base grid is not known.

20: ITRACE — INTEGER *Input*

On entry: the level of trace information required from D03RBF. ITRACE may take the value -1, 0, 1, 2, or 3. If ITRACE < -1, then -1 is assumed and similarly if ITRACE > 3, then 3 is assumed. If ITRACE $= -1$, no output is generated. If ITRACE $= 0$, only warning messages are printed, and if ITRACE > 0, then output from the underlying solver is printed on the current advisory message unit (see X04ABF). This output contains details of the time integration, the nonlinear iteration and the linear solver. The advisory messages are given in greater detail as ITRACE increases. Setting ITRACE $= 1$ allows the user to monitor the progress of the integration without possibly excessive information.

21: IND — INTEGER *Input/Output*

On entry: IND must be set to 0 or 1.

IND = 0

starts the integration in time.

IND = 1

continues the integration after an earlier exit from the routine. In this case, only the following parameters may be reset between calls to D03RBF: TOUT, DT(2), DT(3), TOLS, TOLT, OPTI, OPTR, ITRACE and IFAIL.

Constraint: $0 \le$ IND ≤ 1.

On exit: IND = 1.

22: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, NPDE < 1,
or TOUT $\le$ TS,
or TOUT is too close to TS,
or IND = 0 and DT(1) < 0.0,
or DT(i) < 0.0 for $i = 2$ or 3,
or DT(2) $>$ DT(3),
or IND = 0 and
$0.0 <$ DT(1) $< 10 \times$ ***machine precision*** $\times$ max(|TS|,|TOUT|),
or IND = 0 and DT(1) $>$ TOUT $-$ TS,
or IND = 0 and DT(1) $<$ DT(2) or DT(1) $>$ DT(3),
or TOLS or TOLT ≤ 0.0,
or OPTI(1) < 0,
or OPTI(1) > 0 and OPTI(j) < 0 for $j = 2$, 3 or 4,
or OPTR(1,j) ≤ 0.0 for some $j = 1, 2, \ldots$,NPDE,
or OPTR(2,j) < 0.0 for some $j = 1, 2, \ldots$,NPDE,
or OPTR(3,j) < 0.0 for some $j = 1, 2, \ldots$,NPDE,
or IND $\ne$ 0 or 1,

or IND = 1 on initial entry to D03RBF.

IFAIL = 2

The time step size to be attempted is less than the specified minimum size. This may occur following time step failures and subsequent step size reductions caused by one or more of the following:

the requested accuracy could not be achieved, i.e., TOLT is too small,

the maximum number of linear solver iterations, Newton iterations or Jacobian evaluations is too small,

ILU decomposition of the Jacobian matrix could not be performed, possibly due to singularity of the Jacobian.

Setting ITRACE to a higher value may provide further information.

In the latter two cases the user is advised to check their problem formulation in PDEDEF and/or BNDARY, and the initial values in PDEIV if appropriate.

IFAIL = 3

One or more of the workspace arrays is too small for the required number of grid points. At the initial time step this error may result from either the user setting IERR to -1 in INIDOM, or the internal check on the number of grid points following the call to INIDOM. An estimate of the required sizes for the current stage is output, but more space may be required at a later stage.

IFAIL = 4

IERR was set to 1 in the user-supplied subroutine MONITR, forcing control to be passed back to calling program. Integration was successful as far as T = TS.

IFAIL = 5

The integration has been completed but the maximum number of levels specified in OPTI(1) was insufficient at one or more time steps, meaning that the requested space accuracy could not be achieved. To avoid this warning either increase the value of OPTI(1) or decrease the value of TOLS.

IFAIL = 6

One or more of the output arguments of the user-suppled subroutine INIDOM was incorrectly specified, i.e.,

XMIN $\geq$ XMAX,
or XMAX too close to XMIN,
or YMIN $\geq$ YMAX,
or YMAX too close to YMIN,
or NX or NY < 4,
or NROWS < 4,
or NROWS $>$ NY,
or NPTS $>$ NX $\times$ NY,
or NBNDS < 8,
or NBPTS < 12,
or NBPTS $\geq$ NPTS,
or LROW(i) < 1 or LROW(i) $>$ NPTS for some $i = 1, 2, \ldots,$NROWS,
or LROW(i) $\leq$ LROW($i-1$) for some $i = 2, 3, \ldots,$NROWS,
or IROW(i) < 0 or IROW(i) $>$ NY for some $i = 1, 2, \ldots,$NROWS,
or IROW(i) $\leq$ IROW($i-1$) for some $i = 2, 3, \ldots,$NROWS,
or ICOL(i) < 0 or ICOL(i) $>$ NX for some $i = 1, 2, \ldots,$NPTS,
or LLBND(i) < 1 or LLBND(i) $>$ NBPTS for some $i = 1, 2, \ldots,$NBNDS,
or LLBND(i) $\leq$ LLBND($i-1$) for some $i = 2, 3, \ldots,$NBNDS,
or ILBND(i) $\neq$ 1, 2, 3, 4, 12, 23, 34, 41, 21, 32, 43 or 14, for some $i = 1, 2, \ldots,$NBNDS,
or LBND(i) < 1 or LBND(i) $>$ NPTS for some $i = 1, 2, \ldots,$NBPTS.

7 Accuracy

There are three sources of error in the algorithm: space and time discretisation, and interpolation (linear) between grid levels. The space and time discretisation errors are controlled separately using the parameters TOLS and TOLT described in the following section, and the user should test the effects of varying these parameters. Interpolation errors are generally implicitly controlled by the refinement criterion since in areas where interpolation errors are potentially large, the space monitor will also be large. It can be shown that the global spatial accuracy is comparable to that which would be obtained on a uniform grid of the finest grid size. A full error analysis can be found in [3].

8 Further Comments

8.1 Algorithm Outline

The local uniform grid refinement method is summarised as follows

(1) Initialise the course base grid, an initial solution and an initial time step,
(2) Solve the system of PDEs on the current grid with the current time step,
(3) If the required accuracy in space and the maximum number of grid levels have not yet been reached:
   (a) Determine new finer grid at forward time level,
   (b) Get solution values at previous time level(s) on new grid,
   (c) Interpolate internal boundary values from old grid at forward time,
   (d) Get initial values for the Newton process at forward time,
   (e) Goto 2,
(4) Update the coarser grid solution using the finer grid values,
(5) Estimate error in time integration. If time error is acceptable advance time level,
(6) Determine new step size then goto 2 with coarse base as current grid.

8.2 Refinement Strategy

For each grid point i a space monitor μ_i^s is determined by

$$\mu_i^s = \max_{j=1,\mathrm{NPDE}} \{\gamma_j(\mid \triangle x^2 \frac{\partial^2}{\partial x^2} u_j(x_i, y_i, t) \mid + \mid \triangle y^2 \frac{\partial^2}{\partial y^2} u_j(x_i, y_i, t) \mid)\}, \tag{3}$$

where $\triangle x$ and $\triangle y$ are the grid widths in the x and y directions; and x_i, y_i are the (x, y) co-ordinates at grid point i. The parameter γ_j is obtained from

$$\gamma_j = \frac{w_j^s}{u_j^{max}\ \sigma}, \tag{4}$$

where σ is the user-supplied space tolerance; w_j^s is a weighting factor for the relative importance of the jth PDE component on the space monitor; and u_j^{max} is the approximate maximum absolute value of the jth component. A value for σ must be supplied by the user. Values for w_j^s and u_j^{max} must also be supplied but may be set to the values 1.0 if little information about the solution is known.

A new level of refinement is created if

$$\max_i \{\mu_i^s\} > 0.9 \text{ or } 1.0, \tag{5}$$

depending on the grid level at the previous step in order to avoid fluctuations in the number of grid levels between time steps. If (5) is satisfied then all grid points for which $\mu_i^s > 0.25$ are flagged and surrounding cells are quartered in size.

No derefinement takes place as such, since at each time step the solution on the base grid is computed first and new finer grids are then created based on the new solution. Hence derefinement occurs implicitly. See Section 8.1.

8.3 Time Integration

The time integration is controlled using a time monitor calculated at each level l up to the maximum level used, given by

$$\mu_l^t = \sqrt{\frac{1}{\mathrm{N}} \sum_{j=1}^{\mathrm{NPDE}} w_j^t \sum_{i=1}^{\mathrm{NGPTS}(l)} (\frac{\triangle t}{\alpha_{ij}} u_t(x_i, y_i, t))^2} \tag{6}$$

where NGPTS(l) is the total number of points on grid level l; N = NGPTS(l) × NPDE; $\triangle t$ is the current time step; u_t is the time derivative of u which is approximated by first-order finite differences; w_j^t is the time equivalent of the space weighting factor w_j^s; and α_{ij} is given by

$$\alpha_{ij} = \tau(\frac{u_j^{max}}{100} + \mid u(x_i, y_i, t) \mid) \tag{7}$$

where u_j^{max} is as before, and τ is the user-specified time tolerance.

An integration step is rejected and retried at all levels if

$$\max_l \{\mu_l^t\} > 1.0. \tag{8}$$

9 Example

For this routine two examples are presented, in Section 9.1 and Section 9.2. In the example programs distributed to sites, there is a single example program for D03RBF, with a main program:

```
*     D03RBF Example Program Text
*     Mark 18 Release. NAG Copyright 1997.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. External Subroutines ..
      EXTERNAL         EX1, EX2
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D03RBF Example Program Results'
      CALL EX1
      CALL EX2
      STOP
      END
```

The code to solve the two example problems is given in the subroutines EX1 and EX2, in Section 9.1.1 and Section 9.2.1 respectively.

9.1 Example 1

This example is taken from [1] and is the two dimensional Burgers' system

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} + \epsilon\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right),$$

$$\frac{\partial v}{\partial t} = -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} + \epsilon\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right),$$

with $\epsilon = 10^{-3}$ on the domain given in Figure 3. Dirichlet boundary conditions are used on all boundaries using the exact solution

$$u = \frac{3}{4} - \frac{1}{4(1 + \exp\left((-4x + 4y - t)/(32\epsilon)\right))},$$

$$v = \frac{3}{4} + \frac{1}{4(1 + \exp\left((-4x + 4y - t)/(32\epsilon)\right))}.$$

The solution contains a wave front at $y = x + 0.25t$ which propagates in a direction perpendicular to the front with speed $\sqrt{2}/8$.

9.1.1 Program Text

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

```
*
      SUBROUTINE EX1
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          MXLEV, NPDE, NPTS
      PARAMETER        (MXLEV=5,NPDE=2,NPTS=3000)
      INTEGER          LENIWK, LENRWK, LENLWK
      PARAMETER        (LENIWK=NPTS*(5*MXLEV+14)+2+7*MXLEV,
     +                 LENRWK=NPTS*NPDE*(5*MXLEV+9+18*NPDE)+2*NPTS,
     +                 LENLWK=2*NPTS)
*     .. Scalars in Common ..
      INTEGER          IOUT
*     .. Arrays in Common ..
      real             TWANT(2)
*     .. Local Scalars ..
      real             TOLS, TOLT, TOUT, TS
      INTEGER          I, IFAIL, IND, ITRACE, J, MAXLEV
*     .. Local Arrays ..
      real             DT(3), OPTR(3,NPDE), RWK(LENRWK)
      INTEGER          IWK(LENIWK), OPTI(4)
      LOGICAL          LWK(LENLWK)
*     .. External Subroutines ..
      EXTERNAL         BNDRY1, D03RBF, INIDM1, MONIT1, PDEF1, PDEIV1
*     .. Common blocks ..
      COMMON           /OTIME1/TWANT, IOUT
*     .. Save statement ..
      SAVE             /OTIME1/
*     .. Executable Statements ..
      WRITE (NOUT,*)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Example 1'
      WRITE (NOUT,*)
*
      IND = 0
      ITRACE = 0
      TS = 0.0e0
      TWANT(1) = 0.25e0
      TWANT(2) = 1.0e0
      DT(1) = 0.001e0
      DT(2) = 1.0e-7
      DT(3) = 0.0e0
      TOLS = 0.1e0
      TOLT = 0.05e0
      OPTI(1) = 5
      MAXLEV = OPTI(1)
      DO 20 I = 2, 4
         OPTI(I) = 0
   20 CONTINUE
      DO 60 J = 1, NPDE
         DO 40 I = 1, 3
            OPTR(I,J) = 1.0e0
   40    CONTINUE
   60 CONTINUE
```

```
*
*     Call main routine
*
      DO 120 IOUT = 1, 2
         IFAIL = -1
         TOUT = TWANT(IOUT)
         CALL D03RBF(NPDE,TS,TOUT,DT,TOLS,TOLT,INIDM1,PDEF1,BNDRY1,
     +               PDEIV1,MONIT1,OPTI,OPTR,RWK,LENRWK,IWK,LENIWK,LWK,
     +               LENLWK,ITRACE,IND,IFAIL)
*
*        Print statistics
*
         WRITE (NOUT,'(''Statistics:'')')
         WRITE (NOUT,'('' Time = '',F8.4)') TS
         WRITE (NOUT,'('' Total number of accepted timesteps ='', I5)')
     +     IWK(1)
         WRITE (NOUT,'('' Total number of rejected timesteps ='', I5)')
     +     IWK(2)
         WRITE (NOUT,*)
         WRITE (NOUT,
     +     '(''             T o t a l  n u m b e r  o f   '')')
         WRITE (NOUT,
     + '(''          Residual  Jacobian   Newton ''   , ''  Lin sys'')'
     +     )
         WRITE (NOUT,
     + '(''             evals     evals     iters ''   , ''    iters'')'
     +     )
         WRITE (NOUT,'('' At level '')')
         MAXLEV = OPTI(1)
         DO 80 J = 1, MAXLEV
            IF (IWK(J+2).NE.0) WRITE (NOUT,'(I6,4I10)') J, IWK(J+2),
     +          IWK(J+2+MAXLEV), IWK(J+2+2*MAXLEV), IWK(J+2+3*MAXLEV)
*
   80    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,
     +     '(''             M a x i m u m  n u m b e r '', '' o f'')')
         WRITE (NOUT,
     +     '(''              Newton iters    Lin sys iters '')')
         WRITE (NOUT,'('' At level '')')
         DO 100 J = 1, MAXLEV
            IF (IWK(J+2).NE.0) WRITE (NOUT,'(I6,2I14)') J,
     +          IWK(J+2+4*MAXLEV), IWK(J+2+5*MAXLEV)
  100    CONTINUE
         WRITE (NOUT,*)
*
  120 CONTINUE
*
      RETURN
      END
*
      SUBROUTINE INIDM1(MAXPTS,XMIN,XMAX,YMIN,YMAX,NX,NY,NPTS,NROWS,
     +                  NBNDS,NBPTS,LROW,IROW,ICOL,LLBND,ILBND,LBND,
     +                  IERR)
*     .. Parameters ..
      INTEGER           NOUT
      PARAMETER         (NOUT=6)
```

```
*     .. Scalar Arguments ..
      real              XMAX, XMIN, YMAX, YMIN
      INTEGER           IERR, MAXPTS, NBNDS, NBPTS, NPTS, NROWS, NX, NY
*     .. Array Arguments ..
      INTEGER           ICOL(*), ILBND(*), IROW(*), LBND(*), LLBND(*),
     +                  LROW(*)
*     .. Local Scalars ..
      INTEGER           I, IFAIL, J, LENIWK
*     .. Local Arrays ..
      INTEGER           ICOLD(105), ILBNDD(28), IROWD(11), IWK(122),
     +                  LBNDD(72), LLBNDD(28), LROWD(11)
      CHARACTER*33      PGRID(11)
*     .. External Subroutines ..
      EXTERNAL          D03RYF
*     .. Data statements ..
      DATA              ICOLD/0, 1, 2, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
     +                  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 0, 1, 2, 3, 4,
     +                  5, 6, 7, 8, 9, 10, 0, 1, 2, 3, 4, 5, 8, 9, 10,
     +                  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 0, 1, 2, 3, 4,
     +                  5, 6, 7, 8, 9, 10, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
     +                  10, 0, 1, 2, 3, 4, 5, 6, 7, 8, 0, 1, 2, 3, 4, 5,
     +                  6, 7, 8, 0, 1, 2, 3, 4, 5, 6, 7, 8/
      DATA              ILBNDD/1, 2, 3, 4, 1, 4, 1, 2, 3, 4, 3, 4, 1, 2,
     +                  12, 23, 34, 41, 14, 41, 12, 23, 34, 41, 43, 14,
     +                  21, 32/
      DATA              IROWD/0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10/
      DATA              LBNDD/2, 4, 15, 26, 37, 46, 57, 68, 79, 88, 98,
     +                  99, 100, 101, 102, 103, 104, 96, 86, 85, 84, 83,
     +                  82, 70, 59, 48, 39, 28, 17, 6, 8, 9, 10, 11, 12,
     +                  13, 18, 29, 40, 49, 60, 72, 73, 74, 75, 76, 77,
     +                  67, 56, 45, 36, 25, 33, 32, 42, 52, 53, 43, 1,
     +                  97, 105, 87, 81, 3, 7, 71, 78, 14, 31, 51, 54,
     +                  34/
      DATA              LLBNDD/1, 2, 11, 18, 19, 24, 31, 37, 42, 48, 53,
     +                  55, 56, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
     +                  68, 69, 70, 71, 72/
      DATA              LROWD/1, 4, 15, 26, 37, 46, 57, 68, 79, 88, 97/
*     .. Executable Statements ..
      NX = 11
      NY = 11
*
*     Check MAXPTS against rough estimate of NPTS
*
      NPTS = NX*NY
      IF (MAXPTS.LT.NPTS) THEN
         IERR = -1
         RETURN
      END IF
*
      XMIN = 0.0e0
      YMIN = 0.0e0
      XMAX = 1.0e0
      YMAX = 1.0e0
*
      NROWS = 11
      NPTS = 105
      NBNDS = 28
      NBPTS = 72
```

```
*
      DO 20 I = 1, NROWS
         LROW(I) = LROWD(I)
         IROW(I) = IROWD(I)
   20 CONTINUE
*
      DO 40 I = 1, NBNDS
         LLBND(I) = LLBNDD(I)
         ILBND(I) = ILBNDD(I)
   40 CONTINUE
*
      DO 60 I = 1, NBPTS
         LBND(I) = LBNDD(I)
   60 CONTINUE
*
      DO 80 I = 1, NPTS
         ICOL(I) = ICOLD(I)
   80 CONTINUE
*
      WRITE (NOUT,*) 'Base grid:'
      WRITE (NOUT,*)
      LENIWK = 122
      IFAIL = -1
*
      CALL D03RYF(NX,NY,NPTS,NROWS,NBNDS,NBPTS,LROW,IROW,ICOL,LLBND,
     +            ILBND,LBND,IWK,LENIWK,PGRID,IFAIL)
*
      IF (IFAIL.EQ.0) THEN
         WRITE (NOUT,*) ' '
         DO 100 J = 1, NY
            WRITE (NOUT,*) PGRID(J)
            WRITE (NOUT,*) ' '
  100    CONTINUE
         WRITE (NOUT,*) ' '
      END IF
*
      RETURN
      END
*
      SUBROUTINE PDEIV1(NPTS,NPDE,T,X,Y,U)
*     .. Parameters ..
      real              EPS
      PARAMETER         (EPS=1e-3)
*     .. Scalar Arguments ..
      real              T
      INTEGER           NPDE, NPTS
*     .. Array Arguments ..
      real              U(NPTS,NPDE), X(NPTS), Y(NPTS)
*     .. Local Scalars ..
      real              A
      INTEGER           I
*     .. Intrinsic Functions ..
      INTRINSIC         EXP
*     .. Executable Statements ..
      DO 20 I = 1, NPTS
         A = (-4.0e0*X(I)+4.0e0*Y(I)-T)/(32.0e0*EPS)
         IF (A.LE.0.0e0) THEN
            U(I,1) = 0.75e0 - 0.25e0/(1.0e0+EXP(A))
```

```
            U(I,2) = 0.75e0 + 0.25e0/(1.0e0+EXP(A))
         ELSE
            U(I,1) = 0.75e0 - 0.25e0*EXP(-A)/(EXP(-A)+1.0e0)
            U(I,2) = 0.75e0 + 0.25e0*EXP(-A)/(EXP(-A)+1.0e0)
         END IF
   20 CONTINUE
*
      RETURN
      END
*
      SUBROUTINE PDEF1(NPTS,NPDE,T,X,Y,U,UT,UX,UY,UXX,UXY,UYY,RES)
*     .. Parameters ..
      real              EPS
      PARAMETER         (EPS=1e-3)
*     .. Scalar Arguments ..
      real              T
      INTEGER           NPDE, NPTS
*     .. Array Arguments ..
      real              RES(NPTS,NPDE), U(NPTS,NPDE), UT(NPTS,NPDE),
     +                  UX(NPTS,NPDE), UXX(NPTS,NPDE), UXY(NPTS,NPDE),
     +                  UY(NPTS,NPDE), UYY(NPTS,NPDE), X(NPTS), Y(NPTS)
*     .. Local Scalars ..
      INTEGER           I
*     .. Executable Statements ..
      DO 20 I = 1, NPTS
         RES(I,1) = UT(I,1) - (-U(I,1)*UX(I,1)-U(I,2)*UY(I,1)
     +              +EPS*(UXX(I,1)+UYY(I,1)))
         RES(I,2) = UT(I,2) - (-U(I,1)*UX(I,2)-U(I,2)*UY(I,2)
     +              +EPS*(UXX(I,2)+UYY(I,2)))
   20 CONTINUE

      RETURN
      END
      SUBROUTINE BNDRY1(NPTS,NPDE,T,X,Y,U,UT,UX,UY,NBNDS,NBPTS,LLBND,
     +                  ILBND,LBND,RES)
*     .. Parameters ..
      real              EPS
      PARAMETER         (EPS=1e-3)
*     .. Scalar Arguments ..
      real              T
      INTEGER           NBNDS, NBPTS, NPDE, NPTS
*     .. Array Arguments ..
      real              RES(NPTS,NPDE), U(NPTS,NPDE), UT(NPTS,NPDE),
     +                  UX(NPTS,NPDE), UY(NPTS,NPDE), X(NPTS), Y(NPTS)
      INTEGER           ILBND(NBNDS), LBND(NBPTS), LLBND(NBNDS)
*     .. Local Scalars ..
      real              A
      INTEGER           I, K
*     .. Intrinsic Functions ..
      INTRINSIC         EXP
*     .. Executable Statements ..
      DO 20 K = LLBND(1), NBPTS
         I = LBND(K)
         A = (-4.0e0*X(I)+4.0e0*Y(I)-T)/(32.0e0*EPS)
         IF (A.LE.0.0e0) THEN
            RES(I,1) = U(I,1) - (0.75e0-0.25e0/(1.0e0+EXP(A)))
            RES(I,2) = U(I,2) - (0.75e0+0.25e0/(1.0e0+EXP(A)))
         ELSE
```

```
            RES(I,1) = U(I,1) - (0.75e0-0.25e0*EXP(-A)/(EXP(-A)+1.0e0))
            RES(I,2) = U(I,2) - (0.75e0+0.25e0*EXP(-A)/(EXP(-A)+1.0e0))
         END IF
   20 CONTINUE
*
      RETURN
      END
*
      SUBROUTINE MONIT1(NPDE,T,DT,DTNEW,TLAST,NLEV,XMIN,YMIN,DXB,DYB,
     +                  LGRID,ISTRUC,LSOL,SOL,IERR)
*     .. Parameters ..
      INTEGER           MAXPTS, NOUT
      PARAMETER         (MAXPTS=2500,NOUT=6)
*     .. Scalar Arguments ..
      real              DT, DTNEW, DXB, DYB, T, XMIN, YMIN
      INTEGER           IERR, NLEV, NPDE
      LOGICAL           TLAST
*     .. Array Arguments ..
      real              SOL(*)
      INTEGER           ISTRUC(*), LGRID(*), LSOL(NLEV)
*     .. Scalars in Common ..
      INTEGER           IOUT
*     .. Arrays in Common ..
      real              TWANT(2)
*     .. Local Scalars ..
      INTEGER           IFAIL, IPSOL, IPT, LEVEL, NPTS
*     .. Local Arrays ..
      real              UEX(105,2), X(MAXPTS), Y(MAXPTS)
*     .. External Subroutines ..
      EXTERNAL          D03RZF, PDEIV1
*     .. Common blocks ..
      COMMON            /OTIME1/TWANT, IOUT
*     .. Save statement ..
      SAVE              /OTIME1/
*     .. Executable Statements ..
*
      IFAIL = -1
      IF (TLAST) THEN
         DO 40 LEVEL = 1, NLEV
            IPSOL = LSOL(LEVEL)
*
*           Get grid information
*
            CALL D03RZF(LEVEL,NLEV,XMIN,YMIN,DXB,DYB,LGRID,ISTRUC,NPTS,
     +                  X,Y,MAXPTS,IFAIL)
            IF (IFAIL.NE.0) THEN
               IERR = 1
               RETURN
            END IF
*
            IF (IOUT.EQ.2 .AND. LEVEL.EQ.1) THEN
*
*              Get exact solution
*
               CALL PDEIV1(NPTS,NPDE,T,X,Y,UEX)
               WRITE (NOUT,*)
               WRITE (NOUT,
     +'('' Solution at every 2nd grid point '', ''in level 1 at time '',
```

```
     + F8.4,'':'')') T
                 WRITE (NOUT,*)
                 WRITE (NOUT,
     +'(7X,''x'',10X,''y'',8X,''approx u'',5X,''exact u'',4X,
     + ''approx v'',4X,''exact v'')')
                 WRITE (NOUT,*)
                 IPSOL = LSOL(LEVEL)
                 DO 20 IPT = 1, NPTS, 2
                    WRITE (NOUT,'(6(1X,D11.4))') X(IPT), Y(IPT),
     +              SOL(IPSOL+IPT), UEX(IPT,1), SOL(IPSOL+NPTS+IPT),
     +              UEX(IPT,2)
   20            CONTINUE
                 WRITE (NOUT,*)
              END IF
*
   40    CONTINUE
      END IF
*
      RETURN
      END
```

9.1.2 Program Data

None.

9.1.3 Program Results

```
 D03RBF Example Program Results


 Example 1

 Base grid:


  23  3  3  3  3  3  3  3 34 XX XX

   2 .. .. .. .. .. .. ..  4 XX XX

   2 .. 14  1  1  1  1  1 41 XX XX

   2 ..  4 23  3  3  3  3  3  3 34

   2 ..  4  2 .. .. .. .. .. ..  4

   2 ..  4  2 .. 14  1  1 21 ..  4

   2 ..  4  2 ..  4 XX XX  2 ..  4

   2 ..  4  2 .. 43  3  3 32 ..  4

   2 ..  4  2 .. .. .. .. .. ..  4

   2 ..  4 12  1  1  1  1  1  1 41

  12  1 41 XX XX XX XX XX XX XX XX
```

```
Statistics:
 Time =   0.2500
 Total number of accepted timesteps =   14
 Total number of rejected timesteps =    0

           T o t a l  n u m b e r  o f
          Residual  Jacobian    Newton   Lin sys
             evals     evals     iters     iters
 At level
     1       196        14        28        14
     2       196        14        28        22
     3       196        14        28        25
     4       196        14        28        31
     5       141        10        21        29

           M a x i m u m  n u m b e r  o f
            Newton iters    Lin sys iters
 At level
     1            2             1
     2            2             1
     3            2             1
     4            2             2
     5            3             2


 Solution at every 2nd grid point in level 1 at time   1.0000:

      x           y         approx u     exact u    approx v    exact v

  0.0000E+00  0.0000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.2000E+00  0.0000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.1000E+00  0.1000E+00  0.5002E+00  0.5000E+00  0.9998E+00  0.1000E+01
  0.3000E+00  0.1000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.5000E+00  0.1000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.7000E+00  0.1000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.9000E+00  0.1000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.0000E+00  0.2000E+00  0.5005E+00  0.5005E+00  0.9995E+00  0.9995E+00
  0.2000E+00  0.2000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.4000E+00  0.2000E+00  0.5001E+00  0.5000E+00  0.9999E+00  0.1000E+01
  0.6000E+00  0.2000E+00  0.4999E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.8000E+00  0.2000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.1000E+01  0.2000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.1000E+00  0.3000E+00  0.5000E+00  0.5005E+00  0.1000E+01  0.9995E+00
  0.3000E+00  0.3000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.5000E+00  0.3000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.7000E+00  0.3000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.9000E+00  0.3000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.0000E+00  0.4000E+00  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.2000E+00  0.4000E+00  0.5005E+00  0.5005E+00  0.9995E+00  0.9995E+00
  0.4000E+00  0.4000E+00  0.5002E+00  0.5000E+00  0.9998E+00  0.1000E+01
  0.8000E+00  0.4000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.1000E+01  0.4000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.1000E+00  0.5000E+00  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.3000E+00  0.5000E+00  0.5005E+00  0.5005E+00  0.9995E+00  0.9995E+00
  0.5000E+00  0.5000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.7000E+00  0.5000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.9000E+00  0.5000E+00  0.5001E+00  0.5000E+00  0.9999E+00  0.1000E+01
  0.0000E+00  0.6000E+00  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
```

```
  0.2000E+00  0.6000E+00  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.4000E+00  0.6000E+00  0.5000E+00  0.5005E+00  0.1000E+01  0.9995E+00
  0.6000E+00  0.6000E+00  0.4999E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.8000E+00  0.6000E+00  0.4998E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.1000E+01  0.6000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.1000E+00  0.7000E+00  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.3000E+00  0.7000E+00  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.5000E+00  0.7000E+00  0.5005E+00  0.5005E+00  0.9995E+00  0.9995E+00
  0.7000E+00  0.7000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.9000E+00  0.7000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.0000E+00  0.8000E+00  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.2000E+00  0.8000E+00  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.4000E+00  0.8000E+00  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.6000E+00  0.8000E+00  0.5005E+00  0.5005E+00  0.9995E+00  0.9995E+00
  0.8000E+00  0.8000E+00  0.5000E+00  0.5000E+00  0.1000E+01  0.1000E+01
  0.1000E+00  0.9000E+00  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.3000E+00  0.9000E+00  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.5000E+00  0.9000E+00  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.7000E+00  0.9000E+00  0.4999E+00  0.5005E+00  0.1000E+01  0.9995E+00
  0.0000E+00  0.1000E+01  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.2000E+00  0.1000E+01  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.4000E+00  0.1000E+01  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.6000E+00  0.1000E+01  0.7500E+00  0.7500E+00  0.7500E+00  0.7500E+00
  0.8000E+00  0.1000E+01  0.5005E+00  0.5005E+00  0.9995E+00  0.9995E+00

Statistics:
 Time =   1.0000
 Total number of accepted timesteps =   45
 Total number of rejected timesteps =    0

            T o t a l  n u m b e r  o f
         Residual  Jacobian    Newton   Lin sys
            evals     evals     iters     iters
 At level
     1       630        45        90        45
     2       630        45        90        78
     3       630        45        90        87
     4       630        45        90       124
     5       575        41        83       122

            M a x i m u m  n u m b e r  o f
             Newton iters    Lin sys iters
 At level
     1            2             1
     2            2             1
     3            2             1
     4            2             2
     5            3             2
```

9.2 Example 2

This example comes from [1] and concerns brine transport in porous media. It involves an isothermal groundwater flow model consisting of an equation for the continuity for the fluid and a salt transport equation:

$$n\rho\gamma\frac{\partial w}{\partial t}+\nabla.(\rho\underline{q})=0,$$

$$n\rho\frac{\partial w}{\partial t}+\rho\underline{q}.\nabla w+\nabla.(\rho\underline{J}^w)=0,$$

where w is the salt mass fraction; n is the porosity parameter; γ is a salt coefficient; ρ is the density satisfying $\rho=\rho_0\exp(\gamma w)$ where ρ_0 is the reference viscosity of fresh water; and $\underline{q}$ is the fluid velocity given by

$$\underline{q}=-\frac{k}{\mu}(\nabla p-\rho\underline{g}),$$

where k is the permeability coefficient of the porous medium; p is the pressure; $\underline{g}=(0,-g)$ is the acceleration due to gravity; and μ is the viscosity given by

$$\mu=\mu_0 m(w),\quad m(w)=1+1.85w-4w^2,$$

where μ_0 is a reference viscosity. $\underline{J}^w$ is the salt-dispersion flux vector given by

$$\underline{J}^w=-nD\nabla w,$$

where D is the dispersion tensor for the solute defined as

$$nD=(nD_{\mathrm{mol}}+\alpha_T\mid\underline{q}\mid)I+\frac{(\alpha_L-\alpha_T)}{\mid\underline{q}\mid}\underline{q}\underline{q}^T,\quad\mid\underline{q}\mid=\sqrt{\underline{q}^T\underline{q}}.$$

The coefficients D_{mol}, α_T, α_L are the molecular diffusion and the transversal and longitudinal dispersion respectively.

The dependent variables are p and w, and the values for the constants are

$n=0.2,$	$D_{\mathrm{mol}}=0.0,$	$\rho_0=1000,$	$\mu_0=10^{-3},$
$k=10^{-10},$	$\alpha_T=0.002,$	$p_0=10^5,$	$w_0=0.25,$
$g=9.81,$	$\alpha_L=0.01,$	$\gamma=\ln(1.2),$	$q_c=10^{-4}.$

This particular example is a simulation of a laboratory experiment looking at the displacement of fresh water by brine in a vertical column filled with porous media. The problem can be considered two-dimensional since the column is thin in one of its dimensions. High concentration brine is injected through a gate at the bottom of the column giving rise to a fresh-salt water front moving in all directions. There is a region of total impermeability in part of the column, and so the solution domain is non-rectangular, as shown in Figure 4. The inlet gate is situated between $(0.025,0)$ and $(0.05,0)$.

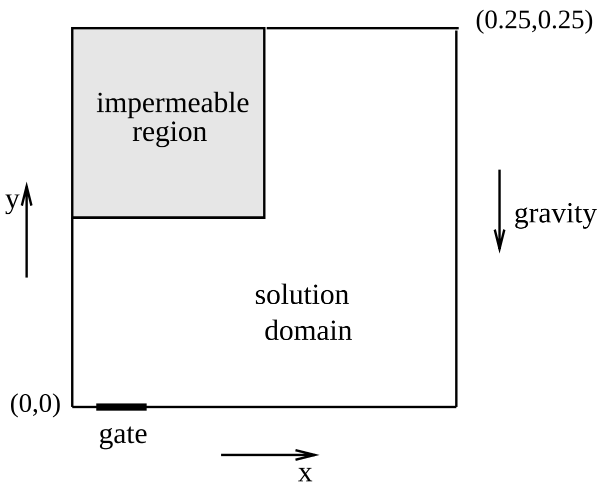

Figure 4

The initial values are

$$p(x,y,0) = p_0 + (0.25 - y)\rho_0 g, \quad w(x,y,0) = 0,$$

and the boundary conditions are

$x = 0$, $0 \leq y \leq 0.125$: $p_x = 0$, $w_x = 0$,

$x = 0.25$, $0 \leq y \leq 0.25$: $p_x = 0$, $w_x = 0$,

$x = 0.125$, $0.125 \leq y \leq 0.25$: $p_x = 0$, $w_x = 0$.

$y = 0$, $0 < x < 0.025$ and $0.05 < x < 0.25$ $-\frac{k}{\mu}(p_y + \rho g) = 0$, $w_y = 0$,

$y = 0$, $0.025 \leq x \leq 0.05$: $-\frac{k}{\mu}(p_y + \rho g) = q_c$, $w = w_0$,

$y = 0.125$, $0 < x < 0.125$: $-\frac{k}{\mu}(p_y + \rho g) = 0$, $w_y = 0$,

$y = 0.25$, $0.125 < x < 0.25$: $p = p_0$, $w_y = 0$,

Note that eventually the solution will reach steady state, with the whole of the domain being filled with brine to a constant concentration, but in the example program shown the solution process is terminated at an earlier stage.

9.2.1 Program Text

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

```
*
      SUBROUTINE EX2
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          MXLEV, NPDE, NPTS
      PARAMETER        (MXLEV=4,NPDE=2,NPTS=2000)
      INTEGER          LENIWK, LENRWK, LENLWK
      PARAMETER        (LENIWK=NPTS*(5*MXLEV+14)+2+7*MXLEV,
     +                 LENRWK=NPTS*NPDE*(5*MXLEV+9+18*NPDE)+2*NPTS,
     +                 LENLWK=2*NPTS)
*     .. Scalars in Common ..
      INTEGER          IOUT
*     .. Arrays in Common ..
      real             TWANT(2)
```

```
*     .. Local Scalars ..
      real              TOLS, TOLT, TOUT, TS
      INTEGER           I, IFAIL, IND, ITRACE, J, MAXLEV
*     .. Local Arrays ..
      real              DT(3), OPTR(3,NPDE), RWK(LENRWK)
      INTEGER           IWK(LENIWK), OPTI(4)
      LOGICAL           LWK(LENLWK)
*     .. External Subroutines ..
      EXTERNAL          BNDRY2, D03RBF, INIDM2, MONIT2, PDEF2, PDEIV2
*     .. Common blocks ..
      COMMON            /OTIME2/TWANT, IOUT
*     .. Save statement ..
      SAVE              /OTIME2/
*     .. Executable Statements ..
      WRITE (NOUT,*)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Example 2'
      WRITE (NOUT,*)
*
      IND = 0
      ITRACE = -1
      TS = 0.0e0
      TWANT(1) = 150.0e0
      TWANT(2) = 750.0e0
      DT(1) = 0.01e0
      DT(2) = 1.0e-3
      DT(3) = 5.0e4
      TOLS = 0.1e0
      TOLT = 0.1e0
      OPTI(1) = 4
      MAXLEV = OPTI(1)
      DO 20 I = 2, 4
         OPTI(I) = 0
   20 CONTINUE
      OPTR(1,1) = 1.1e5
      OPTR(1,2) = 0.25e0
      DO 60 J = 1, NPDE
         DO 40 I = 2, 3
            OPTR(I,J) = 1.0e0
   40    CONTINUE
   60 CONTINUE
*
      DO 120 IOUT = 1, 2
         IFAIL = -1
         TOUT = TWANT(IOUT)
         CALL D03RBF(NPDE,TS,TOUT,DT,TOLS,TOLT,INIDM2,PDEF2,BNDRY2,
     +               PDEIV2,MONIT2,OPTI,OPTR,RWK,LENRWK,IWK,LENIWK,LWK,
     +               LENLWK,ITRACE,IND,IFAIL)
*
*        Print statistics
*
         MAXLEV = OPTI(1)
         WRITE (NOUT,'(''Statistics:'')')
         WRITE (NOUT,'('' Time = '',F8.4)') TS
         WRITE (NOUT,'('' Total number of accepted timesteps ='', I5)')
     +     IWK(1)
         WRITE (NOUT,'('' Total number of rejected timesteps ='', I5)')
     +     IWK(2)
```

```
         WRITE (NOUT,*)
         WRITE (NOUT,
     +     '(''            T o t a l  n u m b e r  o f    '')')
         WRITE (NOUT,
     + '(''          Residual  Jacobian    Newton ''   , ''  Lin sys'')'
     +     )
         WRITE (NOUT,
     + '(''             evals     evals     iters ''   , ''    iters'')'
     +     )
         WRITE (NOUT,'('' At level '')')
         MAXLEV = OPTI(1)
         DO 80 J = 1, MAXLEV
            IF (IWK(J+2).NE.0) WRITE (NOUT,'(I6,4I10)') J, IWK(J+2),
     +          IWK(J+2+MAXLEV), IWK(J+2+2*MAXLEV), IWK(J+2+3*MAXLEV)
*
   80    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,
     +     '(''            M a x i m u m  n u m b e r '', '' o f'')')
         WRITE (NOUT,
     +     '(''             Newton iters    Lin sys iters '')')
         WRITE (NOUT,'('' At level '')')
         DO 100 J = 1, MAXLEV
            IF (IWK(J+2).NE.0) WRITE (NOUT,'(I6,2I14)') J,
     +          IWK(J+2+4*MAXLEV), IWK(J+2+5*MAXLEV)
  100    CONTINUE
         WRITE (NOUT,*)
*
  120 CONTINUE
*
      RETURN
      END
*
      SUBROUTINE INIDM2(MAXPTS,XMIN,XMAX,YMIN,YMAX,NX,NY,NPTS,NROWS,
     +                  NBNDS,NBPTS,LROW,IROW,ICOL,LLBND,ILBND,LBND,
     +                  IERR)
*     .. Parameters ..
      INTEGER           NOUT
      PARAMETER         (NOUT=6)
*     .. Scalar Arguments ..
      real              XMAX, XMIN, YMAX, YMIN
      INTEGER           IERR, MAXPTS, NBNDS, NBPTS, NPTS, NROWS, NX, NY
*     .. Array Arguments ..
      INTEGER           ICOL(*), ILBND(*), IROW(*), LBND(*), LLBND(*),
     +                  LROW(*)
*     .. Local Scalars ..
      INTEGER           I, IFAIL, IPT, J, LENIWK, NX1, NY1
*     .. Local Arrays ..
      INTEGER           IWK(122)
      CHARACTER*33      PGRID(11)
*     .. External Subroutines ..
      EXTERNAL          D03RYF
*     .. Executable Statements ..
*
      NX = 11
      NY = 11
*
*     Check MAXPTS against rough estimate of NPTS
```

```
*
      NPTS = NX*NY
      IF (MAXPTS.LT.NPTS) THEN
         IERR = -1
         RETURN
      END IF
      NROWS = NY
*
      XMIN = 0.0e0
      YMIN = 0.0e0
      XMAX = 0.25e0
      YMAX = 0.25e0
*
      NX1 = (NX-1)/2
      NY1 = (NY-1)/2
*
*     Make grid structure for general NX, NY
*
      IPT = 1
      DO 40 J = 1, NY1 + 1
         LROW(J) = IPT
         IROW(J) = J - 1
         DO 20 I = 1, NX
            ICOL(IPT) = I - 1
            IPT = IPT + 1
   20    CONTINUE
   40 CONTINUE
      DO 80 J = NY1 + 2, NY
         LROW(J) = IPT
         IROW(J) = J - 1
         DO 60 I = NX1 + 1, NX
            ICOL(IPT) = I - 1
            IPT = IPT + 1
   60    CONTINUE
   80 CONTINUE
*
      NPTS = IPT - 1
*
*     Boundaries
*
      NBNDS = 12
*
      ILBND(1) = 1
      ILBND(2) = 2
      ILBND(3) = 3
      ILBND(4) = 2
      ILBND(5) = 3
      ILBND(6) = 4
      ILBND(7) = 12
      ILBND(8) = 23
      ILBND(9) = 32
      ILBND(10) = 23
      ILBND(11) = 34
      ILBND(12) = 41
*
      LLBND(1) = 1
      LLBND(2) = LLBND(1) + NX - 2
      LLBND(3) = LLBND(2) + NY1 - 1
```

```
      LLBND(4) = LLBND(3) + NX1 - 1
      LLBND(5) = LLBND(4) + NY1 - 1
      LLBND(6) = LLBND(5) + NX1 - 1
      LLBND(7) = LLBND(6) + NY - 2
      LLBND(8) = LLBND(7) + 1
      LLBND(9) = LLBND(8) + 1
      LLBND(10) = LLBND(9) + 1
      LLBND(11) = LLBND(10) + 1
      LLBND(12) = LLBND(11) + 1
*
      NBPTS = LLBND(12)
*
*     Lower boundary
*
      DO 100 I = 1, NX - 2
         LBND(LLBND(1)+I-1) = I + 1
  100 CONTINUE
*
*     Upper boundaries
*
      DO 120 I = 1, NX1 - 1
         LBND(LLBND(3)+I-1) = NY1*NX + 1 + I
  120 CONTINUE
      DO 140 I = 1, NX1 - 1
         LBND(LLBND(5)+I-1) = NPTS - I
  140 CONTINUE
*
*     Left boundaries
*
      DO 160 J = 1, NY1 - 1
         LBND(LLBND(2)+J-1) = J*NX + 1
  160 CONTINUE
      DO 180 J = 1, NY1 - 1
         LBND(LLBND(4)+J-1) = (NY1+1)*NX + (J-1)*(NX1+1) + 1
  180 CONTINUE
*
*     Right boundary
*
      DO 200 J = 1, NY1
         LBND(LLBND(6)+J-1) = (J+1)*NX
  200 CONTINUE
      I = LLBND(6) + NY1 - 1
      DO 220 J = 1, NY1 - 1
         LBND(I+J) = LBND(I) + J*(NX1+1)
  220 CONTINUE
*
*     Corners
*
      LBND(LLBND(7)) = 1
      LBND(LLBND(8)) = NY1*NX + 1
      LBND(LLBND(9)) = LBND(LLBND(8)) + NX1
      LBND(LLBND(10)) = LBND(LLBND(5)) - NX1 + 1
      LBND(LLBND(11)) = LBND(LLBND(5)) + 1
      LBND(LLBND(12)) = NX
*
      WRITE (NOUT,*) 'Base grid:'
      WRITE (NOUT,*)
      LENIWK = 122
```

```
      IFAIL = -1
*
      CALL D03RYF(NX,NY,NPTS,NROWS,NBNDS,NBPTS,LROW,IROW,ICOL,LLBND,
     +            ILBND,LBND,IWK,LENIWK,PGRID,IFAIL)
*
      IF (IFAIL.EQ.0) THEN
         WRITE (NOUT,*) ' '
         DO 240 J = 1, NY
            WRITE (NOUT,*) PGRID(J)
            WRITE (NOUT,*) ' '
  240    CONTINUE
         WRITE (NOUT,*) ' '
      END IF
*
      RETURN
      END
*
      SUBROUTINE PDEIV2(NPTS,NPDE,T,X,Y,U)
*     .. Scalar Arguments ..
      real              T
      INTEGER           NPDE, NPTS
*     .. Array Arguments ..
      real              U(NPTS,NPDE), X(NPTS), Y(NPTS)
*     .. Scalars in Common ..
      real              AL, AT, DM, G, GAMMA, KAPPA, MU0, N, P0, QC,
     +                  RHO0, W0
*     .. Local Scalars ..
      INTEGER           I
*     .. Intrinsic Functions ..
      INTRINSIC         LOG
*     .. Common blocks ..
      COMMON            /PARAMS/N, KAPPA, GAMMA, MU0, RHO0, P0, W0, G,
     +                  DM, AL, AT, QC
*     .. Save statement ..
      SAVE              /PARAMS/
*     .. Executable Statements ..
      N = 0.2e0
      KAPPA = 1.0e-10
      G = 9.81e0
      DM = 0.0e0
      AT = 0.002e0
      AL = 0.01e0
      RHO0 = 1.0e+3
      P0 = 1.0e+5
      GAMMA = LOG(1.2e0)
      MU0 = 1.0e-3
      W0 = 0.25e0
      QC = 1.0e-4
*
      DO 20 I = 1, NPTS
         U(I,1) = P0 + (0.25e0-Y(I))*RHO0*G
         U(I,2) = 0.0e0
   20 CONTINUE
*
      RETURN
      END
*
      SUBROUTINE PDEF2(NPTS,NPDE,T,X,Y,U,UT,UX,UY,UXX,UXY,UYY,RES)
```

```
*     .. Scalar Arguments ..
      real              T
      INTEGER           NPDE, NPTS
*     .. Array Arguments ..
      real              RES(NPTS,NPDE), U(NPTS,NPDE), UT(NPTS,NPDE),
     +                  UX(NPTS,NPDE), UXX(NPTS,NPDE), UXY(NPTS,NPDE),
     +                  UY(NPTS,NPDE), UYY(NPTS,NPDE), X(NPTS), Y(NPTS)
*     .. Scalars in Common ..
      real              AL, AT, DM, G, GAMMA, KAPPA, MU0, N, P0, QC,
     +                  RHO0, W0
*     .. Local Scalars ..
      real              JW1, JW1X, JW2, JW2Y, KAPMU, KAPMU2, KAPMUX,
     +                  KAPMUY, MU, MUX, MUY, ND11, ND11Q1, ND11Q2,
     +                  ND11X, ND12, ND12Q1, ND12Q2, ND12X, ND12Y, ND22,
     +                  ND22Q1, ND22Q2, ND22Y, PX, PXX, PXY, PY, PYY, Q1,
     +                  Q1L, Q1X, Q1Y, Q2, Q2L, Q2X, Q2Y, QL, RHO, RHOX,
     +                  RHOY, W, WT, WX, WXX, WXY, WY, WYY
      INTEGER           I
*     .. Intrinsic Functions ..
      INTRINSIC         EXP, SQRT
*     .. Common blocks ..
      COMMON            /PARAMS/N, KAPPA, GAMMA, MU0, RHO0, P0, W0, G,
     +                  DM, AL, AT, QC
*     .. Save statement ..
      SAVE              /PARAMS/
*     .. Executable Statements ..
      DO 20 I = 1, NPTS
         PX = UX(I,1)
         PY = UY(I,1)
         W = U(I,2)
         WT = UT(I,2)
         WX = UX(I,2)
         WY = UY(I,2)
         RHO = RHO0*EXP(GAMMA*W)
         RHOX = RHO*(GAMMA*WX)
         RHOY = RHO*(GAMMA*WY)
         MU = MU0*(1+1.85e0*W-4.0e0*W*W)
         MUX = MU0*(1.85e0*WX-8.0e0*W*WX)
         MUY = MU0*(1.85e0*WY-8.0e0*W*WY)
         KAPMU = KAPPA/MU
         KAPMU2 = -KAPMU/MU
         KAPMUX = KAPMU2*MUX
         KAPMUY = KAPMU2*MUY
         Q1 = -KAPMU*PX
         Q2 = -KAPMU*(PY+RHO*G)
         QL = SQRT(Q1*Q1+Q2*Q2)
         IF (QL.EQ.0.0e0) THEN
            Q1L = 0.0e0
            Q2L = 0.0e0
         ELSE
            Q1L = Q1/QL
            Q2L = Q2/QL
         END IF
         ND11 = N*DM + AT*QL + (AL-AT)*Q1*Q1L
         ND12 = (AL-AT)*Q1*Q2L
         ND22 = N*DM + AT*QL + (AL-AT)*Q2*Q2L
         PXX = UXX(I,1)
         PXY = UXY(I,1)
```

```
            PYY = UYY(I,1)
            WXX = UXX(I,2)
            WXY = UXY(I,2)
            WYY = UYY(I,2)
            ND11Q1 = (AT+(AL-AT)*(2-Q1L**2))*Q1L
            ND11Q2 = (AT-(AL-AT)*(Q1L**2))*Q2L
            ND12Q1 = (AL-AT)*Q2L**3
            ND12Q2 = (AL-AT)*Q1L**3
            ND22Q1 = (AT-(AL-AT)*(Q2L**2))*Q1L
            ND22Q2 = (AT+(AL-AT)*(2-Q2L**2))*Q2L
            Q1X = -(KAPMUX*PX+KAPMU*PXX)
            Q1Y = -(KAPMUY*PX+KAPMU*PXY)
            Q2X = -(KAPMUX*(PY+RHO*G)+KAPMU*(PXY+RHOX*G))
            Q2Y = -(KAPMUY*(PY+RHO*G)+KAPMU*(PYY+RHOY*G))
            ND11X = ND11Q1*Q1X + ND11Q2*Q2X
            ND12X = ND12Q1*Q1X + ND12Q2*Q2X
            ND12Y = ND12Q1*Q1Y + ND12Q2*Q2Y
            ND22Y = ND22Q1*Q1Y + ND22Q2*Q2Y
            JW1 = -(ND11*WX+ND12*WY)
            JW2 = -(ND12*WX+ND22*WY)
            JW1X = -(ND11X*WX+ND11*WXX+ND12X*WY+ND12*WXY)
            JW2Y = -(ND12Y*WX+ND12*WXY+ND22Y*WY+ND22*WYY)
*
            RES(I,1) = N*RHO*GAMMA*WT + RHOX*Q1 + RHO*Q1X + RHOY*Q2 +
     +                 RHO*Q2Y
            RES(I,2) = N*RHO*WT + RHO*Q1*WX + RHO*Q2*WY + RHOX*JW1 +
     +                 RHO*JW1X + RHOY*JW2 + RHO*JW2Y
   20 CONTINUE
*
      RETURN
      END
*
      SUBROUTINE BNDRY2(NPTS,NPDE,T,X,Y,U,UT,UX,UY,NBNDS,NBPTS,LLBND,
     +                  ILBND,LBND,RES)
*     .. Scalar Arguments ..
      real              T
      INTEGER           NBNDS, NBPTS, NPDE, NPTS
*     .. Array Arguments ..
      real              RES(NPTS,NPDE), U(NPTS,NPDE), UT(NPTS,NPDE),
     +                  UX(NPTS,NPDE), UY(NPTS,NPDE), X(NPTS), Y(NPTS)
      INTEGER           ILBND(NBNDS), LBND(NBPTS), LLBND(NBNDS)
*     .. Scalars in Common ..
      real              AL, AT, DM, G, GAMMA, KAPPA, MU0, N, P0, QC,
     +                  RHO0, W0
*     .. Local Scalars ..
      real              KAPMU, MU, PY, Q2, RHO, W
      INTEGER           I, J, K
*     .. Intrinsic Functions ..
      INTRINSIC         EXP
*     .. Common blocks ..
      COMMON            /PARAMS/N, KAPPA, GAMMA, MU0, RHO0, P0, W0, G,
     +                  DM, AL, AT, QC
*     .. Save statement ..
      SAVE              /PARAMS/
*     .. Executable Statements ..
      J = 1
*
*     y = 0.0 boundary
```

```
*
      DO 20 K = LLBND(J), LLBND(J+1) - 1
         I = LBND(K)
         IF ((0.LT.X(I) .AND. X(I).LT.0.025e0) .OR. (0.075e0.LT.X(I)
     +       .AND. X(I).LT.0.25e0)) THEN
            PY = UY(I,1)
            W = U(I,2)
            RHO = RHO0*EXP(GAMMA*W)
            MU = MU0*(1+1.85e0*W-4.0e0*W*W)
            KAPMU = KAPPA/MU
            Q2 = -KAPMU*(PY+RHO*G)
            RES(I,1) = Q2
            RES(I,2) = UY(I,2)
         ELSE
            PY = UY(I,1)
            W = U(I,2)
            RHO = RHO0*EXP(GAMMA*W)
            MU = MU0*(1+1.85e0*W-4.0e0*W*W)
            KAPMU = KAPPA/MU
            Q2 = -KAPMU*(PY+RHO*G)
*
            RES(I,1) = Q2 - QC
            RES(I,2) = W - W0
         END IF
   20 CONTINUE
*
      J = 5
*
*     y = 0.25 boundary
*
      DO 40 K = LLBND(J), LLBND(J+1) - 1
         I = LBND(K)
         RES(I,1) = U(I,1) - P0
         RES(I,2) = UY(I,2)
   40 CONTINUE
*
      DO 80 J = 2, 6, 2
*
*        x = 0.0, 0.125, 0.25 boundaries
*
         DO 60 K = LLBND(J), LLBND(J+1) - 1
            I = LBND(K)
            RES(I,1) = UX(I,1)
            RES(I,2) = UX(I,2)
   60    CONTINUE
   80 CONTINUE
*
      J = 3
*
*     y = 0.125 boundary
*
      DO 100 K = LLBND(J), LLBND(J+1) - 1
         I = LBND(K)
         PY = UY(I,1)
         W = U(I,2)
         RHO = RHO0*EXP(GAMMA*W)
         RES(I,1) = UY(I,1) + RHO*G
         RES(I,2) = UY(I,2)
```

```
  100 CONTINUE
*
      DO 140 J = 7, 11
*
*        Corners: dp/dx = dw/dx = 0
*
         DO 120 K = LLBND(J), LLBND(J+1) - 1
            I = LBND(K)
            RES(I,1) = UX(I,1)
            RES(I,2) = UX(I,2)
  120    CONTINUE
  140 CONTINUE
*
      DO 160 K = LLBND(12), NBPTS
         I = LBND(K)
         RES(I,1) = UX(I,1)
         RES(I,2) = UX(I,2)
  160 CONTINUE
*
      RETURN
      END
*
      SUBROUTINE MONIT2(NPDE,T,DT,DTNEW,TLAST,NLEV,XMIN,YMIN,DXB,DYB,
     +                  LGRID,ISTRUC,LSOL,SOL,IERR)
*     .. Parameters ..
      INTEGER           MAXPTS, NOUT
      PARAMETER         (MAXPTS=2000,NOUT=6)
*     .. Scalar Arguments ..
      real              DT, DTNEW, DXB, DYB, T, XMIN, YMIN
      INTEGER           IERR, NLEV, NPDE
      LOGICAL           TLAST
*     .. Array Arguments ..
      real              SOL(*)
      INTEGER           ISTRUC(*), LGRID(*), LSOL(NLEV)
*     .. Scalars in Common ..
      INTEGER           IOUT
*     .. Arrays in Common ..
      real              TWANT(2)
*     .. Local Scalars ..
      INTEGER           IFAIL, IPSOL, IPT, LEVEL, NPTS
*     .. Local Arrays ..
      real              X(MAXPTS), Y(MAXPTS)
*     .. External Subroutines ..
      EXTERNAL          D03RZF
*     .. Common blocks ..
      COMMON            /OTIME2/TWANT, IOUT
*     .. Save statement ..
      SAVE              /OTIME2/
*     .. Executable Statements ..
*
      IFAIL = -1
      IF (TLAST) THEN
         DO 40 LEVEL = 1, NLEV
            IPSOL = LSOL(LEVEL)
*
*           Get grid information
*
            CALL D03RZF(LEVEL,NLEV,XMIN,YMIN,DXB,DYB,LGRID,ISTRUC,NPTS,
```

```
     +                  X,Y,MAXPTS,IFAIL)
            IF (IFAIL.NE.0) THEN
               IERR = 1
               RETURN
            END IF
*
            IF (IOUT.EQ.2 .AND. LEVEL.EQ.1) THEN
               WRITE (NOUT,*)
               WRITE (NOUT,
     +'('' Solution at every 2nd grid point '', ''in level 1 at time '',
     + F8.4,'':'')') T
               WRITE (NOUT,*)
               WRITE (NOUT,'(7X,''x'',10X,''y'',8X,''approx w'')')
               WRITE (NOUT,*)
               IPSOL = LSOL(LEVEL)
               DO 20 IPT = 1, NPTS, 2
                  WRITE (NOUT,'(3(1X,D11.4))') X(IPT), Y(IPT),
     +              SOL(IPSOL+NPTS+IPT)
   20          CONTINUE
               WRITE (NOUT,*)
            END IF
   40    CONTINUE
      END IF
*
      RETURN
      END
```

9.2.2 Program Data

None.

9.2.3 Program Results

```
 D03RBF Example Program Results



 Example 2

 Base grid:


  XX XX XX XX XX 23  3  3  3  3 34

  XX XX XX XX XX  2 .. .. .. ..  4

  XX XX XX XX XX  2 .. .. .. ..  4

  XX XX XX XX XX  2 .. .. .. ..  4

  XX XX XX XX XX  2 .. .. .. ..  4

  23  3  3  3  3 32 .. .. .. ..  4

   2 .. .. .. .. .. .. .. .. ..  4

   2 .. .. .. .. .. .. .. .. ..  4
```

```
  2 .. .. .. .. .. .. .. .. .. ..  4

  2 .. .. .. .. .. .. .. .. .. ..  4

 12  1  1  1  1  1  1  1  1  1 41


Statistics:
 Time = 150.0000
 Total number of accepted timesteps =  111
 Total number of rejected timesteps =    0

            T o t a l  n u m b e r  o f
         Residual  Jacobian    Newton   Lin sys
            evals     evals     iters     iters
 At level
     1      1554       111       222       492
     2      1554       111       222       412
     3      1555       111       223       484
     4      1555       111       223       625

            M a x i m u m  n u m b e r  o f
             Newton iters    Lin sys iters
 At level
     1            2             7
     2            2             7
     3            3             6
     4            3             9


 Solution at every 2nd grid point in level 1 at time 750.0000:

       x           y        approx w

  0.0000E+00  0.0000E+00  0.2496E+00
  0.5000E-01  0.0000E+00  0.2500E+00
  0.1000E+00  0.0000E+00  0.2496E+00
  0.1500E+00  0.0000E+00  0.2370E+00
  0.2000E+00  0.0000E+00  0.1615E+00
  0.2500E+00  0.0000E+00  0.1068E-01
  0.2500E-01  0.2500E-01  0.2492E+00
  0.7500E-01  0.2500E-01  0.2496E+00
  0.1250E+00  0.2500E-01  0.2436E+00
  0.1750E+00  0.2500E-01  0.1977E+00
  0.2250E+00  0.2500E-01  0.6179E-01
  0.0000E+00  0.5000E-01  0.2377E+00
  0.5000E-01  0.5000E-01  0.2452E+00
  0.1000E+00  0.5000E-01  0.2408E+00
  0.1500E+00  0.5000E-01  0.2073E+00
  0.2000E+00  0.5000E-01  0.1038E+00
  0.2500E+00  0.5000E-01  0.2914E-02
  0.2500E-01  0.7500E-01  0.2087E+00
  0.7500E-01  0.7500E-01  0.2155E+00
  0.1250E+00  0.7500E-01  0.1942E+00
  0.1750E+00  0.7500E-01  0.1251E+00
  0.2250E+00  0.7500E-01  0.1905E-01
  0.0000E+00  0.1000E+00  0.1125E+00
  0.5000E-01  0.1000E+00  0.1207E+00
```

```
  0.1000E+00  0.1000E+00  0.1185E+00
  0.1500E+00  0.1000E+00  0.1198E+00
  0.2000E+00  0.1000E+00  0.4098E-01
  0.2500E+00  0.1000E+00  0.5599E-03
  0.2500E-01  0.1250E+00  0.1888E-01
  0.7500E-01  0.1250E+00  0.2361E-01
  0.1250E+00  0.1250E+00  0.5862E-01
  0.1750E+00  0.1250E+00  0.5284E-01
  0.2250E+00  0.1250E+00  0.3616E-02
  0.1250E+00  0.1500E+00  0.4117E-01
  0.1750E+00  0.1500E+00  0.2753E-01
  0.2250E+00  0.1500E+00  0.1248E-02
  0.1250E+00  0.1750E+00  0.2045E-01
  0.1750E+00  0.1750E+00  0.1121E-01
  0.2250E+00  0.1750E+00  0.3885E-03
  0.1250E+00  0.2000E+00  0.8343E-02
  0.1750E+00  0.2000E+00  0.3956E-02
  0.2250E+00  0.2000E+00  0.1171E-03
  0.1250E+00  0.2250E+00  0.3053E-02
  0.1750E+00  0.2250E+00  0.1306E-02
  0.2250E+00  0.2250E+00  0.4249E-04
  0.1250E+00  0.2500E+00  0.1489E-02
  0.1750E+00  0.2500E+00  0.6204E-03
  0.2250E+00  0.2500E+00  0.1627E-04

Statistics:
 Time = 750.0000
 Total number of accepted timesteps =  164
 Total number of rejected timesteps =    0

            T o t a l  n u m b e r  o f
         Residual  Jacobian    Newton   Lin sys
            evals     evals     iters     iters
 At level
     1      2296       164       328       854
     2      2296       164       328       881
     3      2297       164       329       970
     4      1695       121       243       705

            M a x i m u m  n u m b e r  o f
             Newton iters    Lin sys iters
 At level
     1             2              7
     2             2             11
     3             3              9
     4             3              9
```