D03RAF – 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.

1 Purpose

D03RAF integrates a system of linear or nonlinear, time-dependent partial differential equations (PDEs) in two space dimensions on a rectangular 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. D03RAF originates from the VLUGR2 package [1] [2].

2 Specification

```
 SUBROUTINE D03RAF(NPDE, TS, TOUT, DT, XMIN, XMAX, YMIN, YMAX, NX,
1                  NY, TOLS,TOLT, PDEDEF, BNDARY, PDEIV, MONITR,
2                  OPTI,OPTR, RWK, LENRWK, IWK, LENIWK, LWK,LENLWK,
3                  ITRACE, IND, IFAIL)
 INTEGER           NPDE, NX, NY, OPTI(4), LENRWK, IWK(LENIWK),
1                  LENIWK, LENLWK, ITRACE, IND, IFAIL
 real              TS, TOUT, DT(3), XMIN, XMAX, YMIN, YMAX, TOLS,
1                  TOLT, OPTR(3,NPDE), RWK(LENRWK)
 LOGICAL           LWK(LENLWK)
 EXTERNAL          PDEDEF, BNDARY, PDEIV, MONITR
```

3 Description

D03RAF integrates the system of PDEs:

$$F_j(t, x, y, u, u_t, u_x, u_y, u_{xx}, u_{xy}, u_{yy}) = 0, \quad j = 1, 2, \ldots, \text{NPDE}, \tag{1}$$

for x and y in the rectangular domain $x_{\min} \leq x \leq x_{\max}$, $y_{\min} \leq y \leq y_{\max}$, and time interval $t_0 \leq t \leq t_{\text{out}}$, where the vector u is the set of solution values

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

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 $u(x, y, t)$ must be specified at $t = t_0$ in a subroutine PDEIV.

Note that whilst complete generality is offered by the master equations (1), D03RAF 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

$$G_j(t, x, y, u, u_t, u_x, u_y) = 0 \text{ at } x = x_{\min}, x_{\max}, \; y = y_{\min}, y_{\max}, \text{ for } j = 1, 2, \ldots, \text{NPDE}. \tag{2}$$

The domain is covered by a uniform coarse base grid of size $n_x \times n_y$ 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 $t = t_{\text{out}}$.

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.

The solution at all grid levels is 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 included. 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.

Within the user-specified subroutines PDEIV, PDEDEF, BNDARY and MONITR the data structure is as follows. Each point on a particular grid is given an index (ranging from 1 to the total number of points on the grid) and all coordinate or solution information is stored in arrays according to this index, e.g., X(i) and Y(i) contain the x- and y-coordinate of point i, and U(i,j) contains the jth solution component u_j at point i.

Further details of the underlying algorithm can be found in Section 8 and in [1] [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] Adjerid S and Flaherty J E (1988) A local refinement finite element method for two dimensional parabolic systems *SIAM J. Sci. Statist. Comput.* **9** 792–811

[6] Brown P N, Hindmarsh A C and Petzold L R (1994) Using Krylov methods in the solution of large scale differential-algebraic systems *SIAM J. Sci. Statist. Comput.* **15** 1467–1488

5 Parameters

1: NPDE — INTEGER *Input*

On entry: the number of PDEs in the system.

Constraint: NPDE ≥ 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:

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

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: XMIN — ***real*** *Input*

6: XMAX — ***real*** *Input*

On entry: the extents of the rectangular domain in the x-direction, i.e., the x-coordinates of the left and right boundaries respectively.

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

7: YMIN — ***real*** *Input*

8: YMAX — ***real*** *Input*

On entry: the extents of the rectangular domain in the y-direction, i.e., the y-coordinates of the lower and upper boundaries respectively.

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

9: NX — INTEGER *Input*

On entry: the number of grid points in the x-direction (including the boundary points).

Constraint: NX ≥ 4.

10: NY — INTEGER *Input*

On entry: the number of grid points in the y-direction (including the boundary points).

Constraint: NY ≥ 4.

11: 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.

12: 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.

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

PDEDEF must evaluate the functions F_j, j = 1,2,.., 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-coordinate of the ith grid point, for $i = 1, 2, \ldots$,NPTS.

5: Y(NPTS) — ***real*** array *Input*

On entry: Y(i) contains the y-coordinate 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 D03RAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

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

BNDARY must evaluate the functions G_j, j = 1,2,.., 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, NBPTS, LBND,
1                  RES)
 INTEGER           NPTS, NPDE, NBPTS, 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-coordinate of the ith grid point, for $i = 1, 2, \ldots$,NPTS.

5: Y(NPTS) — ***real*** array *Input*
On entry: Y(i) contains the y-coordinate 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: NBPTS — INTEGER *Input*
On entry: the number of boundary points in the grid.

11: LBND(NBPTS) — INTEGER array *Input*
On entry: LBND(i) contains the grid index for the ith boundary point for $i = 1, 2, \ldots$,NBPTS. Hence the ith boundary point has coordinates X(LBND(i)) and Y(LBND(i)), and the corresponding solution values are U(LBND(i),NPDE), etc.

12: 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 must **not** be altered.

BNDARY must be declared as EXTERNAL in the (sub)program from which D03RAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

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

PDEIV must specify the initial values of the PDE components u at all points in the 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 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-coordinate of the ith grid point, for $i = 1, 2, \ldots$,NPTS.

5: Y(NPTS) — ***real*** array *Input*
On entry: Y(i) contains the y-coordinate 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 D03RAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

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

MONITR is called by D03RAF 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.

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, NGPTS, XPTS,
     1                  YPTS, LSOL, SOL, IERR)
      INTEGER           NPDE, NLEV, NGPTS(NLEV), LSOL(NLEV), IERR
      real              T, DT, DTNEW, XPTS(*), YPTS(*), 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 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: NGPTS(NLEV) — INTEGER array *Input*
On entry: NGPTS(l) contains the number of grid points at level l, for $l = 1, 2, \ldots$,NLEV.

8: XPTS(*) — ***real*** array *Input*
On entry: contains the x-coordinates of the grid points in each level in turn, i.e., X(i), for $i = 1, 2, \ldots$,NGPTS(l), $l = 1, 2, \ldots$,NLEV.

So for level l, X(i) = XPTS($k+i$), where k = NGPTS(1) + NGPTS(2) $+ \cdots +$ NGPTS($l-1$), for $i = 1, 2, \ldots$,NGPTS(l), $l = 1, 2, \ldots$,NLEV.

9: YPTS(*) — ***real*** array *Input*
On entry: contains the y-coordinates of the grid points in each level in turn, i.e., Y(i), for $i = 1, 2, \ldots$,NGPTS(l), $l = 1, 2, \ldots$,NLEV.

So for level l, Y(i) = YPTS($k+i$), where k = NGPTS(1) + NGPTS(2) $+ \cdots +$ NGPTS($l-1$), for $i = 1, 2, \ldots$,NGPTS(l), $l = 1, 2, \ldots$,NLEV.

10: 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.)

11: SOL(*) — ***real*** array *Input*
On entry: SOL contains the solution U(NGPTS(l),NPDE) at time T for each grid level l in turn, positioned according to LSOL i.e., for level l,

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

for $i = 1, \ldots$,NGPTS(l), $j = 1, \ldots$,NPDE, $l = 1, \ldots$,NLEV.

12: IERR — INTEGER *Output*
On exit: IERR should be set to 1 to force a tidy termination 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 D03RAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

17: 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) $\geq$ 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) $\geq$ 0. The default value is OPTI(2) = 2.

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

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

Constraints: if OPTI(1) $\geq$ 0 and OPTI(1) > 0 then OPTI(i) $\geq$ 0, $i = 2, 3, 4$.

18: 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) $\geq$ 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) $\geq$ 0.0 for $j = 1, 2, \ldots$,NPDE.

Constraints:

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

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

20: LENRWK — INTEGER *Input*

On entry: the dimension of the array RWK as declared in the (sub)program from which D03RAF is called.

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

LENRWK = MAXPTS × NPDE × (5×l+18×NPDE+9) + 2 × 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).

Constraint: LENRWK $\geq$ NX × NY × NPDE × (14+18×NPDE) + 2 × NX × NY (the required size for the initial grid).

21: IWK(LENIWK) — INTEGER array *Output*

On entry: if IND = 0, IWK need not be set. Otherwise IWK must remain unchanged from a previous call to D03RAF.

On exit: the following components of the array IWK 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×m+l) contains the total number of Newton iterations performed at grid level l;

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

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

IWK(2+5×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 D03RAF. 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.

22: LENIWK — INTEGER *Input*

On entry: the dimension of the array IWK as declared in the (sub)program from which D03RAF is called.

The required value of LENIWK can not be determined exactly in advance, but a suggested value is LENIWK = MAXPTS × (14+5×m) + 7 × 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).

Constraint: LENIWK $\geq$ 19 × NX × NY + 9 (the required size for the initial grid).

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

24: LENLWK — INTEGER *Input*

On entry: the dimension of the array LWK as declared in the (sub)program from which D03RAF is called.

The required value of LENLWK can not be determined exactly in advanced, 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).

Constraint: LENLWK $\geq$ NX $\times$ NY + 1 (the required size for the initial grid).

25: ITRACE — INTEGER *Input*

On entry: the level of trace information required from D03RAF. 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.

26: 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 D03RAF: TOUT, DT(2), DT(3), TOLS, TOLT, OPTI, OPTR, ITRACE and IFAIL.

Constraint: $0 \leq$ IND ≤ 1.

On exit: IND = 1.

27: 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 $\leq$ 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 = 0.0 and $0.0 <$ DT(1) $< 10 \times$ ***machine precision*** $\times$ max(|TS|,|TOUT|),

or IND = 0.0 and DT(1) > TOUT − TS,
or IND = 0.0 and DT(1) < DT(2) or DT(1) > DT(3),
or XMIN ≥ XMAX,
or XMAX too close to XMIN,
or YMIN ≥ YMAX,
or YMAX too close to YMIN,
or NX or NY < 4,
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 LENRWK, LENIWK or LENLWK too small for initial grid level,
or IND ≠ 0 or 1,
or IND = 1 on initial entry to D03RAF,

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. 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.

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

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

where $\triangle x$ and $\triangle y$ are the grid widths in the x and y directions; and x_i, y_i are the x and 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 value 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

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

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 D03RAF, with a main program:

```
*     D03RAF Example Program Text
*     Mark 19 Revised. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. External Subroutines ..
      EXTERNAL         EX1, EX2
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D03RAF 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 stems from combustion theory and is a model for a single, one-step reaction of a mixture of two chemicals [5]. The PDE for the temperature of the mixture u is

$$\frac{\partial u}{\partial t} = d\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + D(1+\alpha-u)\exp\left(-\frac{\delta}{u}|\right)$$

for $0 \leq x, y \leq 1$ and $t \geq 0$, with initial conditions $u(x,y,0) = 1$ for $0 \leq x, y \leq 1$, and boundary conditions

$$u_x(0,y,t) = 0, u(1,y,t) = 1 \quad \text{for} \quad 0 \leq y \leq 1,$$

$$u_y(x,0,t) = 0, u(x,1,t) = 1 \quad \text{for} \quad 0 \leq x \leq 1.$$

The heat release parameter $\alpha = 1$, the Damkohler number $D = R\exp(\delta)/(\alpha\delta)$, the activation energy $\delta = 20$, the reaction rate $R = 5$, and the diffusion parameter $d = 0.1$.

For small times the temperature gradually increases in a circular region about the origin, and at about $t = 0.24$ 'ignition' occurs causing the temperature to suddenly jump from near unity to $1 + \alpha$, and a reaction front forms and propagates outwards, becoming steeper. Thus during the solution, just one grid level is used up to the ignition point, then two levels, and then three as the reaction front steepens.

9.1.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
      SUBROUTINE EX1
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
```

```
      INTEGER          MXLEV, NPDE, NPTS
      PARAMETER        (MXLEV=3,NPDE=1,NPTS=2000)
      INTEGER          LENIWK, LENRWK, LENLWK
      PARAMETER        (LENIWK=NPTS*(5*MXLEV+14)+2+7*MXLEV,
     +                 LENRWK=NPTS*NPDE*(5*MXLEV+9+18*NPDE)+NPTS*2,
     +                 LENLWK=NPTS+1)
*     .. Scalars in Common ..
      real             ALPHA, D, DELTA, DIFF, REAC
      INTEGER          IOUT
*     .. Arrays in Common ..
      real             TWANT(2)
*     .. Local Scalars ..
      real             TOLS, TOLT, TOUT, TS, XMAX, XMIN, YMAX, YMIN
      INTEGER          I, IFAIL, IND, ITRACE, J, MAXLEV, NX, NY
*     .. Local Arrays ..
      real             DT(3), OPTR(3,NPDE), RWK(LENRWK)
      INTEGER          IWK(LENIWK), OPTI(4)
      LOGICAL          LWK(LENLWK)
*     .. External Subroutines ..
      EXTERNAL         BNDRY1, D03RAF, MONIT1, PDEF1, PDEIV1
*     .. Intrinsic Functions ..
      INTRINSIC        EXP
*     .. Common blocks ..
      COMMON           /OTIME1/TWANT, IOUT
      COMMON           /PARAM1/ALPHA, DELTA, REAC, DIFF, D
*     .. Save statement ..
      SAVE             /OTIME1/, /PARAM1/
*     .. Executable Statements ..
      WRITE (NOUT,*)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Example 1'
      WRITE (NOUT,*)
*
*     Problem Parameters
*
      ALPHA = 1.0e0
      DELTA = 20.0e0
      REAC = 5.0e0
      DIFF = 0.1e0
      D = REAC*EXP(DELTA)/(ALPHA*DELTA)
*
      IND = 0
      ITRACE = 0
      TS = 0.0e0
      DT(1) = 0.1e-2
      DT(2) = 0.0e0
      DT(3) = 0.0e0
      TOUT = 0.24e0
      TWANT(1) = 0.24e0
      TWANT(2) = 0.25e0
      XMIN = 0.0e0
      XMAX = 1.0e0
      YMIN = 0.0e0
      YMAX = 1.0e0
      NX = 21
      NY = 21
      TOLS = 0.5e0
      TOLT = 0.01e0
```

```
      DO 20 I = 1, 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
*
      DO 120 IOUT = 1, 2
         IFAIL = -1
         TOUT = TWANT(IOUT)
         CALL D03RAF(NPDE,TS,TOUT,DT,XMIN,XMAX,YMIN,YMAX,NX,NY,TOLS,
     +              TOLT,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 = 3
         DO 80 J = 1, MAXLEV
            IF (IWK(J+2).NE.0) WRITE (NOUT,'(I8,4I10)') 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,'(I8,2I14)') J,
     +          IWK(J+2+4*MAXLEV), IWK(J+2+5*MAXLEV)
  100    CONTINUE
         WRITE (NOUT,*)
*
  120 CONTINUE
*
      RETURN
      END
*
```

```
      SUBROUTINE PDEIV1(NPTS,NPDE,T,X,Y,U)
*     .. Scalar Arguments ..
      real              T
      INTEGER           NPDE, NPTS
*     .. Array Arguments ..
      real              U(NPTS,NPDE), X(NPTS), Y(NPTS)
*     .. Local Scalars ..
      INTEGER           I
*     .. Executable Statements ..
*
      DO 20 I = 1, NPTS
         U(I,1) = 1.0e0
   20 CONTINUE
*
      RETURN
      END
*
      SUBROUTINE PDEF1(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              ALPHA, D, DELTA, DIFF, REAC
*     .. Local Scalars ..
      INTEGER           I
*     .. Intrinsic Functions ..
      INTRINSIC         EXP
*     .. Common blocks ..
      COMMON            /PARAM1/ALPHA, DELTA, REAC, DIFF, D
*     .. Save statement ..
      SAVE              /PARAM1/
*     .. Executable Statements ..
      DO 20 I = 1, NPTS
         RES(I,1) = UT(I,1) - DIFF*(UXX(I,1)+UYY(I,1)) -
     +              D*(1.0e0+ALPHA-U(I,1))*EXP(-DELTA/U(I,1))
   20 CONTINUE
*
      RETURN
      END
*
      SUBROUTINE BNDRY1(NPTS,NPDE,T,X,Y,U,UT,UX,UY,NBPTS,LBND,RES)
*     .. Scalar Arguments ..
      real              T
      INTEGER           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           LBND(NBPTS)
*     .. Local Scalars ..
      real              TOL
      INTEGER           I, J
```

```
*     .. External Functions ..
      real              X02AJF
      EXTERNAL          X02AJF
*     .. Intrinsic Functions ..
      INTRINSIC         ABS
*     .. Executable Statements ..
*
      TOL = 10.e0*X02AJF()
*
      DO 20 I = 1, NBPTS
         J = LBND(I)
         IF (ABS(X(J)).LE.TOL) THEN
            RES(J,1) = UX(J,1)
         ELSE IF (ABS(X(J)-1.0e0).LE.TOL) THEN
            RES(J,1) = U(J,1) - 1.0e0
         ELSE IF (ABS(Y(J)).LE.TOL) THEN
            RES(J,1) = UY(J,1)
         ELSE IF (ABS(Y(J)-1.0e0).LE.TOL) THEN
            RES(J,1) = U(J,1) - 1.0e0
         END IF
   20 CONTINUE
*
      RETURN
      END
*
      SUBROUTINE MONIT1(NPDE,T,DT,DTNEW,TLAST,NLEV,NGPTS,XPTS,YPTS,LSOL,
     +                  SOL,IERR)
*     .. Parameters ..
      INTEGER           NOUT
      PARAMETER         (NOUT=6)
*     .. Scalar Arguments ..
      real              DT, DTNEW, T
      INTEGER           IERR, NLEV, NPDE
      LOGICAL           TLAST
*     .. Array Arguments ..
      real              SOL(*), XPTS(*), YPTS(*)
      INTEGER           LSOL(NLEV), NGPTS(NLEV)
*     .. Scalars in Common ..
      INTEGER           IOUT
*     .. Arrays in Common ..
      real              TWANT(2)
*     .. Local Scalars ..
      INTEGER           I, IPSOL, IPT, LEVEL, NPTS
*     .. Common blocks ..
      COMMON            /OTIME1/TWANT, IOUT
*     .. Save statement ..
      SAVE              /OTIME1/
*     .. Executable Statements ..
*
      IF (TLAST) THEN
*
*        Print solution
*
         IF (IOUT.EQ.2) THEN
            WRITE (NOUT,
     +'('' Solution at every 4th grid point '',    ''in level 1 at time
     +'', F8.4,'':'')') T
```

```
            WRITE (NOUT,*)
            WRITE (NOUT,'(7X,''x'',10X,''y'',8X,''approx u'')')
            WRITE (NOUT,*)
            LEVEL = 1
            NPTS = NGPTS(LEVEL)
            IPSOL = LSOL(LEVEL)
            IPT = 1
            DO 20 I = 1, NPTS, 4
               WRITE (NOUT,'(3(1X,D11.4))') XPTS(IPT+I-1),
     +           YPTS(IPT+I-1), SOL(IPSOL+I)
   20       CONTINUE
            WRITE (NOUT,*)
         END IF
      END IF
*
      RETURN
      END
*
```

9.1.2 Program Data

None.

9.1.3 Program Results

```
D03RAF Example Program Results


Example 1

Statistics:
Time =   0.2400
Total number of accepted timesteps =   75
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       600        75       150       159

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

Solution at every 4th grid point in level 1 at time   0.2500:

      x          y        approx u

 0.0000E+00  0.0000E+00  0.2000E+01
 0.2000E+00  0.0000E+00  0.2000E+01
 0.4000E+00  0.0000E+00  0.2000E+01
 0.6000E+00  0.0000E+00  0.2000E+01
 0.8000E+00  0.0000E+00  0.1240E+01
 0.1000E+01  0.0000E+00  0.1000E+01
 0.1500E+00  0.5000E-01  0.2000E+01
 0.3500E+00  0.5000E-01  0.2000E+01
```

```
0.5500E+00  0.5000E-01  0.2000E+01
0.7500E+00  0.5000E-01  0.1645E+01
0.9500E+00  0.5000E-01  0.1048E+01
0.1000E+00  0.1000E+00  0.2000E+01
0.3000E+00  0.1000E+00  0.2000E+01
0.5000E+00  0.1000E+00  0.2000E+01
0.7000E+00  0.1000E+00  0.1999E+01
0.9000E+00  0.1000E+00  0.1097E+01
0.5000E-01  0.1500E+00  0.2000E+01
0.2500E+00  0.1500E+00  0.2000E+01
0.4500E+00  0.1500E+00  0.2000E+01
0.6500E+00  0.1500E+00  0.2000E+01
0.8500E+00  0.1500E+00  0.1154E+01
0.0000E+00  0.2000E+00  0.2000E+01
0.2000E+00  0.2000E+00  0.2000E+01
0.4000E+00  0.2000E+00  0.2000E+01
0.6000E+00  0.2000E+00  0.2000E+01
0.8000E+00  0.2000E+00  0.1240E+01
0.1000E+01  0.2000E+00  0.1000E+01
0.1500E+00  0.2500E+00  0.2000E+01
0.3500E+00  0.2500E+00  0.2000E+01
0.5500E+00  0.2500E+00  0.2000E+01
0.7500E+00  0.2500E+00  0.1635E+01
0.9500E+00  0.2500E+00  0.1048E+01
0.1000E+00  0.3000E+00  0.2000E+01
0.3000E+00  0.3000E+00  0.2000E+01
0.5000E+00  0.3000E+00  0.2000E+01
0.7000E+00  0.3000E+00  0.1999E+01
0.9000E+00  0.3000E+00  0.1097E+01
0.5000E-01  0.3500E+00  0.2000E+01
0.2500E+00  0.3500E+00  0.2000E+01
0.4500E+00  0.3500E+00  0.2000E+01
0.6500E+00  0.3500E+00  0.2000E+01
0.8500E+00  0.3500E+00  0.1153E+01
0.0000E+00  0.4000E+00  0.2000E+01
0.2000E+00  0.4000E+00  0.2000E+01
0.4000E+00  0.4000E+00  0.2000E+01
0.6000E+00  0.4000E+00  0.2000E+01
0.8000E+00  0.4000E+00  0.1234E+01
0.1000E+01  0.4000E+00  0.1000E+01
0.1500E+00  0.4500E+00  0.2000E+01
0.3500E+00  0.4500E+00  0.2000E+01
0.5500E+00  0.4500E+00  0.2000E+01
0.7500E+00  0.4500E+00  0.1508E+01
0.9500E+00  0.4500E+00  0.1048E+01
0.1000E+00  0.5000E+00  0.2000E+01
0.3000E+00  0.5000E+00  0.2000E+01
0.5000E+00  0.5000E+00  0.2000E+01
0.7000E+00  0.5000E+00  0.1993E+01
0.9000E+00  0.5000E+00  0.1095E+01
0.5000E-01  0.5500E+00  0.2000E+01
0.2500E+00  0.5500E+00  0.2000E+01
0.4500E+00  0.5500E+00  0.2000E+01
0.6500E+00  0.5500E+00  0.2000E+01
0.8500E+00  0.5500E+00  0.1145E+01
0.0000E+00  0.6000E+00  0.2000E+01
0.2000E+00  0.6000E+00  0.2000E+01
0.4000E+00  0.6000E+00  0.2000E+01
```

```
 0.6000E+00  0.6000E+00  0.2000E+01
 0.8000E+00  0.6000E+00  0.1200E+01
 0.1000E+01  0.6000E+00  0.1000E+01
 0.1500E+00  0.6500E+00  0.2000E+01
 0.3500E+00  0.6500E+00  0.2000E+01
 0.5500E+00  0.6500E+00  0.2000E+01
 0.7500E+00  0.6500E+00  0.1253E+01
 0.9500E+00  0.6500E+00  0.1044E+01
 0.1000E+00  0.7000E+00  0.1999E+01
 0.3000E+00  0.7000E+00  0.1999E+01
 0.5000E+00  0.7000E+00  0.1993E+01
 0.7000E+00  0.7000E+00  0.1279E+01
 0.9000E+00  0.7000E+00  0.1082E+01
 0.5000E-01  0.7500E+00  0.1645E+01
 0.2500E+00  0.7500E+00  0.1635E+01
 0.4500E+00  0.7500E+00  0.1508E+01
 0.6500E+00  0.7500E+00  0.1253E+01
 0.8500E+00  0.7500E+00  0.1109E+01
 0.0000E+00  0.8000E+00  0.1240E+01
 0.2000E+00  0.8000E+00  0.1240E+01
 0.4000E+00  0.8000E+00  0.1234E+01
 0.6000E+00  0.8000E+00  0.1200E+01
 0.8000E+00  0.8000E+00  0.1119E+01
 0.1000E+01  0.8000E+00  0.1000E+01
 0.1500E+00  0.8500E+00  0.1154E+01
 0.3500E+00  0.8500E+00  0.1153E+01
 0.5500E+00  0.8500E+00  0.1145E+01
 0.7500E+00  0.8500E+00  0.1109E+01
 0.9500E+00  0.8500E+00  0.1029E+01
 0.1000E+00  0.9000E+00  0.1097E+01
 0.3000E+00  0.9000E+00  0.1097E+01
 0.5000E+00  0.9000E+00  0.1095E+01
 0.7000E+00  0.9000E+00  0.1082E+01
 0.9000E+00  0.9000E+00  0.1039E+01
 0.5000E-01  0.9500E+00  0.1048E+01
 0.2500E+00  0.9500E+00  0.1048E+01
 0.4500E+00  0.9500E+00  0.1048E+01
 0.6500E+00  0.9500E+00  0.1044E+01
 0.8500E+00  0.9500E+00  0.1029E+01
 0.0000E+00  0.1000E+01  0.1000E+01
 0.2000E+00  0.1000E+01  0.1000E+01
 0.4000E+00  0.1000E+01  0.1000E+01
 0.6000E+00  0.1000E+01  0.1000E+01
 0.8000E+00  0.1000E+01  0.1000E+01
 0.1000E+01  0.1000E+01  0.1000E+01

Statistics:
Time =   0.2500
Total number of accepted timesteps =  180
Total number of rejected timesteps =    1

           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      1468       181       382       391
      2       662        82       170       170
      3       176        22        44        44
```

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

9.2 Example 2

This example is taken from a multispecies food web model, in which predator-prey relationships in a spatial domain are simulated [6]. In this example there is just one species each of prey and predator, and the two PDEs for the concentrations c_1 and c_2 of the prey and the predator respectively are

$$\frac{\partial c_1}{\partial t} = c_1(b_1 + a_{11}c_1 + a_{12}c_2) + d_1\left(\frac{\partial^2 c_1}{\partial x^2} + \frac{\partial^2 c_1}{\partial y^2}\right),$$

$$0 = c_2(b_2 + a_{21}c_1 + a_{22}c_2) + d_2\left(\frac{\partial^2 c_2}{\partial x^2} + \frac{\partial^2 c_2}{\partial y^2}\right),$$

with $a_{11} = a_{22} = -1$, $a_{12} = -0.5 \times 10^{-6}$, and $a_{21} = 10^4$, and

$$b_1 = 1 + \alpha xy + \beta \sin(4\pi x)\sin(4\pi y),$$

where $\alpha = 50$ and $\beta = 300$, and $b_2 = -b_1$.

The initial conditions are taken to be simple peaked functions which satisfy the boundary conditions and very nearly satisfy the PDEs:

$$c_1 = 10 + (16x(1-x)y(1-y))^2,$$

$$c_2 = b_2 + a_{21}c_1,$$

and the boundary conditions are of Neumann type, i.e., zero normal derivatives everywhere.

During the solution a number of peaks and troughs develop across the domain, and so the number of levels required increases with time. Since the solution varies rapidly in space across the whole of the domain, refinement at intermediate levels tends to occur at all points of the domain.

9.2.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
      SUBROUTINE EX2
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          MXLEV, NPDE, NPTS
      PARAMETER        (MXLEV=4,NPDE=2,NPTS=8000)
      INTEGER          LENIWK, LENRWK, LENLWK
      PARAMETER        (LENIWK=NPTS*(5*MXLEV+14)+2+7*MXLEV,
     +                 LENRWK=NPTS*NPDE*(5*MXLEV+9+18*NPDE)+NPTS*2,
     +                 LENLWK=NPTS+1)
*     .. Scalars in Common ..
      real             ALPHA, BETA, PI
      INTEGER          IOUT
*     .. Arrays in Common ..
      real             TWANT(2)
```

```
*     .. Local Scalars ..
      real             TOLS, TOLT, TOUT, TS, XMAX, XMIN, XX, YMAX, YMIN
      INTEGER          I, IFAIL, IND, ITRACE, J, MAXLEV, NX, NY
*     .. Local Arrays ..
      real             DT(3), OPTR(3,NPDE), RWK(LENRWK)
      INTEGER          IWK(LENIWK), OPTI(4)
      LOGICAL          LWK(LENLWK)
*     .. External Functions ..
      real             X01AAF
      EXTERNAL         X01AAF
*     .. External Subroutines ..
      EXTERNAL         BNDRY2, D03RAF, MONIT2, PDEF2, PDEIV2
*     .. Common blocks ..
      COMMON           /OTIME2/TWANT, IOUT
      COMMON           /PARAM2/ALPHA, BETA, PI
*     .. Save statement ..
      SAVE             /OTIME2/, /PARAM2/
*     .. Executable Statements ..
      WRITE (NOUT,*)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Example 2'
      WRITE (NOUT,*)
*
      XX = 0.0e0
      PI = X01AAF(XX)
      ALPHA = 50.0e0
      BETA = 300.0e0
*
      IND = 0
      ITRACE = 0
      TS = 0.0e0
      TWANT(1) = 0.01e0
      TWANT(2) = 0.025e0
      DT(1) = 0.5e-3
      DT(2) = 1.0e-6
      DT(3) = 0.0e0
      XMIN = 0.0e0
      XMAX = 1.0e0
      YMIN = 0.0e0
      YMAX = 1.0e0
      TOLS = 0.075e0
      TOLT = 0.1e0
      NX = 11
      NY = 11
      OPTI(1) = 4
      DO 20 I = 2, 4
         OPTI(I) = 0
   20 CONTINUE
      OPTR(1,1) = 250.0e0
      OPTR(1,2) = 1.5e6
      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 D03RAF(NPDE,TS,TOUT,DT,XMIN,XMAX,YMIN,YMAX,NX,NY,TOLS,
     +               TOLT,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 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              ALPHA, BETA, PI
```

```
*     .. Local Scalars ..
      real             B2, FP
      INTEGER          I
*     .. Intrinsic Functions ..
      INTRINSIC        SIN
*     .. Common blocks ..
      COMMON           /PARAM2/ALPHA, BETA, PI
*     .. Save statement ..
      SAVE             /PARAM2/
*     .. Executable Statements ..
*
      FP = 4.0e0*PI
*
      DO 20 I = 1, NPTS
         B2 = -1.0e0 - ALPHA*X(I)*Y(I) - BETA*SIN(FP*X(I))*SIN(FP*Y(I))
         U(I,1) = 1.0e1 + (16.0e0*X(I)*(1.0e0-X(I))*Y(I)*(1.0e0-Y(I)))
     +            **2
         U(I,2) = B2 + 1.0e4*U(I,1)
   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             ALPHA, BETA, PI
*     .. Local Scalars ..
      real             B1, B2, FP
      INTEGER          I
*     .. Intrinsic Functions ..
      INTRINSIC        SIN
*     .. Common blocks ..
      COMMON           /PARAM2/ALPHA, BETA, PI
*     .. Save statement ..
      SAVE             /PARAM2/
*     .. Executable Statements ..
      FP = 4.0e0*PI
*
      DO 20 I = 1, NPTS
         B1 = 1.0e0 + ALPHA*X(I)*Y(I) + BETA*SIN(FP*X(I))*SIN(FP*Y(I))
         B2 = -B1
         RES(I,1) = UT(I,1) - (UXX(I,1)+UYY(I,1)) - U(I,1)*(B1-U(I,1)
     +              -0.5e-6*U(I,2))
         RES(I,2) = -0.05e0*(UXX(I,2)+UYY(I,2)) - U(I,2)
     +              *(B2+1.0e4*U(I,1)-U(I,2))
   20 CONTINUE
*
      RETURN
      END
*
```

```
      SUBROUTINE BNDRY2(NPTS,NPDE,T,X,Y,U,UT,UX,UY,NBPTS,LBND,RES)
*     .. Scalar Arguments ..
      real              T
      INTEGER           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           LBND(NBPTS)
*     .. Local Scalars ..
      real              TOL
      INTEGER           I, J
*     .. External Functions ..
      real              X02AJF
      EXTERNAL          X02AJF
*     .. Intrinsic Functions ..
      INTRINSIC         ABS
*     .. Executable Statements ..
*
      TOL = 10.e0*X02AJF()
*
      DO 20 I = 1, NBPTS
         J = LBND(I)
         IF (ABS(X(J)).LE.TOL .OR. ABS(X(J)-1.0e0).LE.TOL) THEN
            RES(J,1) = UX(J,1)
            RES(J,2) = UX(J,2)
         ELSE IF (ABS(Y(J)).LE.TOL .OR. ABS(Y(J)-1.0e0).LE.TOL) THEN
            RES(J,1) = UY(J,1)
            RES(J,2) = UY(J,2)
         END IF
   20 CONTINUE
*
      RETURN
      END
*
      SUBROUTINE MONIT2(NPDE,T,DT,DTNEW,TLAST,NLEV,NGPTS,XPTS,YPTS,LSOL,
     +                  SOL,IERR)
*     .. Parameters ..
      INTEGER           NOUT
      PARAMETER         (NOUT=6)
*     .. Scalar Arguments ..
      real              DT, DTNEW, T
      INTEGER           IERR, NLEV, NPDE
      LOGICAL           TLAST
*     .. Array Arguments ..
      real              SOL(*), XPTS(*), YPTS(*)
      INTEGER           LSOL(NLEV), NGPTS(NLEV)
*     .. Scalars in Common ..
      INTEGER           IOUT
*     .. Arrays in Common ..
      real              TWANT(2)
*     .. Local Scalars ..
      INTEGER           I, IPSOL, IPT, LEVEL, NPTS
*     .. Common blocks ..
      COMMON            /OTIME2/TWANT, IOUT
*     .. Save statement ..
      SAVE              /OTIME2/
*     .. Executable Statements ..
*
```

```
      IF (TLAST) THEN
*
*        Print solution
*
         IF (IOUT.EQ.2) THEN
            WRITE (NOUT,
     +'('' Solution at every 2nd grid point '',    ''in level 1 at time
     +'', F8.4,'':'')') T
            WRITE (NOUT,*)
            WRITE (NOUT,
     +        '(7X,''x'',10X,''y'',9X,''approx c1'',3X,''approx c2'')')
            WRITE (NOUT,*)
            LEVEL = 1
            NPTS = NGPTS(LEVEL)
            IPSOL = LSOL(LEVEL)
            IPT = 1
            DO 20 I = 1, NPTS, 2
               WRITE (NOUT,'(2(1X,D11.4),2X,D11.4,2X,D11.4)')
     +           XPTS(IPT+I-1), YPTS(IPT+I-1), SOL(IPSOL+I),
     +           SOL(IPSOL+NPTS+I)
   20       CONTINUE
            WRITE (NOUT,*)
         END IF
      END IF
*
      RETURN
      END
```

9.2.2 Program Data

None.

9.2.3 Program Results

```
D03RAF Example Program Results


Example 2

Statistics:
Time =   0.0100
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        42
    2       168        12        24        34
    3        70         5        10        16

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

Solution at every 2nd grid point in level 1 at time 0.0250:

x	y	approx c1	approx c2
0.0000E+00	0.0000E+00	0.6615E+02	0.6615E+06
0.2000E+00	0.0000E+00	0.5138E+02	0.5137E+06
0.4000E+00	0.0000E+00	0.1274E+02	0.1275E+06
0.6000E+00	0.0000E+00	0.5217E+02	0.5217E+06
0.8000E+00	0.0000E+00	0.1684E+02	0.1684E+06
0.1000E+01	0.0000E+00	0.4618E+01	0.4619E+05
0.1000E+00	0.1000E+00	0.8832E+02	0.8829E+06
0.3000E+00	0.1000E+00	0.1897E+02	0.1898E+06
0.5000E+00	0.1000E+00	0.3109E+02	0.3109E+06
0.7000E+00	0.1000E+00	0.5115E+02	0.5114E+06
0.9000E+00	0.1000E+00	0.6498E+01	0.6526E+05
0.0000E+00	0.2000E+00	0.5138E+02	0.5137E+06
0.2000E+00	0.2000E+00	0.4480E+02	0.4479E+06
0.4000E+00	0.2000E+00	0.1763E+02	0.1764E+06
0.6000E+00	0.2000E+00	0.4849E+02	0.4848E+06
0.8000E+00	0.2000E+00	0.2308E+02	0.2309E+06
0.1000E+01	0.2000E+00	0.1998E+02	0.1998E+06
0.1000E+00	0.3000E+00	0.1897E+02	0.1898E+06
0.3000E+00	0.3000E+00	0.3745E+02	0.3744E+06
0.5000E+00	0.3000E+00	0.2815E+02	0.2815E+06
0.7000E+00	0.3000E+00	0.2379E+02	0.2380E+06
0.9000E+00	0.3000E+00	0.6076E+02	0.6074E+06
0.0000E+00	0.4000E+00	0.1274E+02	0.1275E+06
0.2000E+00	0.4000E+00	0.1763E+02	0.1764E+06
0.4000E+00	0.4000E+00	0.5816E+02	0.5813E+06
0.6000E+00	0.4000E+00	0.1425E+02	0.1428E+06
0.8000E+00	0.4000E+00	0.5783E+02	0.5782E+06
0.1000E+01	0.4000E+00	0.6492E+02	0.6492E+06
0.1000E+00	0.5000E+00	0.3109E+02	0.3109E+06
0.3000E+00	0.5000E+00	0.2815E+02	0.2815E+06
0.5000E+00	0.5000E+00	0.2966E+02	0.2966E+06
0.7000E+00	0.5000E+00	0.3422E+02	0.3422E+06
0.9000E+00	0.5000E+00	0.4004E+02	0.4003E+06
0.0000E+00	0.6000E+00	0.5217E+02	0.5217E+06
0.2000E+00	0.6000E+00	0.4849E+02	0.4848E+06
0.4000E+00	0.6000E+00	0.1425E+02	0.1428E+06
0.6000E+00	0.6000E+00	0.7001E+02	0.6998E+06
0.8000E+00	0.6000E+00	0.2397E+02	0.2398E+06
0.1000E+01	0.6000E+00	0.1981E+02	0.1981E+06
0.1000E+00	0.7000E+00	0.5115E+02	0.5114E+06
0.3000E+00	0.7000E+00	0.2379E+02	0.2380E+06
0.5000E+00	0.7000E+00	0.3422E+02	0.3422E+06
0.7000E+00	0.7000E+00	0.5069E+02	0.5067E+06
0.9000E+00	0.7000E+00	0.3143E+02	0.3145E+06
0.0000E+00	0.8000E+00	0.1684E+02	0.1684E+06
0.2000E+00	0.8000E+00	0.2308E+02	0.2309E+06
0.4000E+00	0.8000E+00	0.5783E+02	0.5781E+06
0.6000E+00	0.8000E+00	0.2397E+02	0.2398E+06
0.8000E+00	0.8000E+00	0.7164E+02	0.7162E+06
0.1000E+01	0.8000E+00	0.8397E+02	0.8397E+06
0.1000E+00	0.9000E+00	0.6498E+01	0.6526E+05
0.3000E+00	0.9000E+00	0.6076E+02	0.6074E+06
0.5000E+00	0.9000E+00	0.4004E+02	0.4003E+06

```
 0.7000E+00  0.9000E+00   0.3143E+02   0.3145E+06
 0.9000E+00  0.9000E+00   0.1403E+03   0.1403E+07
 0.0000E+00  0.1000E+01   0.4618E+01   0.4619E+05
 0.2000E+00  0.1000E+01   0.1998E+02   0.1998E+06
 0.4000E+00  0.1000E+01   0.6492E+02   0.6491E+06
 0.6000E+00  0.1000E+01   0.1980E+02   0.1980E+06
 0.8000E+00  0.1000E+01   0.8397E+02   0.8396E+06
 0.1000E+01  0.1000E+01   0.1075E+03   0.1075E+07

Statistics:
Time =   0.0250
Total number of accepted timesteps =   29
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       406        29        58        87
    2       378        27        54        79
    3       280        20        40        61
    4        98         7        14        27

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