NAG Fortran Library Routine Document

D03PJF/D03PJA

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

D03PJF/D03PJA integrates a system of linear or nonlinear parabolic partial differential equations (PDEs), in one space variable with scope for coupled ordinary differential equations (ODEs). The spatial discretisation is performed using a Chebyshev C^0 collocation method, and the method of lines is employed to reduce the PDEs to a system of ODEs. The resulting system is solved using a backward differentiation formula (BDF) method or a Theta method (switching between Newton's method and functional iteration).

D03PJA is a version of D03PJF that has additional parameters in order to make it safe for use in multithreaded applications (see Section 5 below).

2 Specifications

2.1 Specification for D03PJF

```
 SUBROUTINE D03PJF(NPDE, M, TS, TOUT, PDEDEF, BNDARY, U, NBKPTS, XBKPTS,
1                  NPOLY, NPTS, X, NCODE, ODEDEF, NXI, XI, NEQN, UVINIT,
2                  RTOL, ATOL, ITOL, NORM, LAOPT, ALGOPT, W, NW, IW, NIW,
3                  ITASK, ITRACE, IND, IFAIL)
 INTEGER           NPDE, M, NBKPTS, NPOLY, NPTS, NCODE, NXI, NEQN, ITOL,
1                  NW, IW(NIW), NIW, ITASK, ITRACE, IND, IFAIL
 real              TS, TOUT, U(NEQN), XBKPTS(NBKPTS), X(NPTS), XI(*),
1                  RTOL(*), ATOL(*), ALGOPT(30), W(NW)
 CHARACTER*1       NORM, LAOPT
 EXTERNAL          PDEDEF, BNDARY, ODEDEF, UVINIT
```

2.2 Specification for D03PJA

```
 SUBROUTINE D03PJA(NPDE, M, TS, TOUT, PDEDEF, BNDARY, U, NBKPTS, XBKPTS,
1                  NPOLY, NPTS, X, NCODE, ODEDEF, NXI, XI, NEQN, UVINIT,
2                  RTOL, ATOL, ITOL, NORM, LAOPT, ALGOPT, W, NW, IW, NIW,
3                  ITASK, ITRACE, IND, IUSER, RUSER, CWSAV, LWSAV, IWSAV,
4                  RWSAV, IFAIL)
 INTEGER           NPDE, M, NBKPTS, NPOLY, NPTS, NCODE, NXI, NEQN, ITOL,
1                  NW, IW(NIW), NIW, ITASK, ITRACE, IND, IUSER(*),
2                  IWSAV(505), IFAIL
 real              TS, TOUT, U(NEQN), XBKPTS(NBKPTS), X(NPTS), XI(*),
1                  RTOL(*), ATOL(*), ALGOPT(30), W(NW), RUSER(*),
2                  RWSAV(1100)
 LOGICAL           LWSAV(100)
 CHARACTER*1       NORM, LAOPT
 CHARACTER*80      CWSAV(10)
 EXTERNAL          PDEDEF, BNDARY, ODEDEF, UVINIT
```

3 Description

D03PJF/D03PJA integrates the system of parabolic-elliptic equations and coupled ODEs

$$\sum_{j=1}^{\text{NPDE}} P_{i,j}\frac{\partial U_j}{\partial t} + Q_i = x^{-m}\frac{\partial}{\partial x}(x^m R_i), \quad i = 1,2,\ldots,\text{NPDE}, \quad a \le x \le b,\ t \ge t_0, \tag{1}$$

$$F_i(t, V, \dot{V}, \xi, U^*, U_x^*, \mathrm{R}^*, U_t^*, U_{xt}^*) = 0, \quad i = 1, 2, \ldots, \mathrm{NCODE}, \tag{2}$$

where (1) defines the PDE part and (2) generalizes the coupled ODE part of the problem.

In (1), $P_{i,j}$ and R_i depend on x, t, U, U_x, and V; Q_i depends on x, t, U, U_x, V and **linearly** on $\dot{V}$. The vector U is the set of PDE solution values

$$U(x,t) = [U_1(x,t), \ldots, U_{\mathrm{NPDE}}(x,t)]^T,$$

and the vector U_x is the partial derivative with respect to x. Note that $P_{i,j}$, Q_i and R_i must not depend on $(\partial U)/(\partial t)$. The vector V is the set of ODE solution values

$$V(t) = [V_1(t), \ldots, V_{\mathrm{NCODE}}(t)]^T,$$

and $\dot{V}$ denotes its derivative with respect to time.

In (2), ξ represents a vector of n_ξ spatial coupling points at which the ODEs are coupled to the PDEs. These points may or may not be equal to some of the PDE spatial mesh points. U^*, U_x^*, R^*, U_t^* and U_{xt}^* are the functions U, U_x, R, U_t and U_{xt} evaluated at these coupling points. Each F_i may only depend linearly on time derivatives. Hence the equation (2) may be written more precisely as

$$F = G - A\dot{V} - B\begin{pmatrix} U_t^* \\ U_{xt}^* \end{pmatrix}, \tag{3}$$

where $\mathrm{F} = [F_1, \ldots, F_{\mathrm{NCODE}}]^{\mathrm{T}}$, G is a vector of length NCODE, A is an NCODE by NCODE matrix, B is an NCODE by $(n_\xi \times \mathrm{NPDE})$ matrix and the entries in G, A and B may depend on t, ξ, U^*, U_x^* and V. In practice the user needs only to supply a vector of information to define the ODEs and not the matrices A and B. (See Section 5 for the specification of the user-supplied procedure ODEDEF.)

The integration in time is from t_0 to t_{out}, over the space interval $a \le x \le b$, where $a = x_1$ and $b = x_{\mathrm{NBKPTS}}$ are the leftmost and rightmost of a user-defined set of break-points $x_1, x_2, \ldots, x_{\mathrm{NBKPTS}}$. The co-ordinate system in space is defined by the value of m; $m = 0$ for Cartesian co-ordinates, $m = 1$ for cylindrical polar co-ordinates and $m = 2$ for spherical polar co-ordinates.

The PDE system which is defined by the functions $P_{i,j}$, Q_i and R_i must be specified in a subroutine PDEDEF supplied by the user.

The initial values of the functions $U(x,t)$ and $V(t)$ must be given at $t = t_0$. These values are calculated in a user-supplied subroutine, UVINIT.

The functions R_i which may be thought of as fluxes, are also used in the definition of the boundary conditions. The boundary conditions must have the form

$$\beta_i(x,t)R_i(x,t,U,U_x,V) = \gamma_i(x,t,U,U_x,V,\dot{V}), \quad i = 1, 2, \ldots, \mathrm{NPDE}, \tag{4}$$

where $x = a$ or $x = b$. The functions γ_i may only depend **linearly** on $\dot{V}$.

The boundary conditions must be specified in a subroutine BNDARY provided by the user.

The algebraic-differential equation system which is defined by the functions F_i must be specified in a subroutine ODEDEF supplied by the user. The user must also specify the coupling points ξ in the array XI. Thus, the problem is subject to the following restrictions:

(i) in (1), $\dot{V}_j(t)$, for $j = 1, 2, \ldots, \mathrm{NCODE}$, may only appear **linearly** in the functions Q_i, for $i = 1, 2, \ldots, \mathrm{NPDE}$, with a similar restriction for γ;

(ii) $P_{i,j}$ and the flux R_i must not depend on any time derivatives;

(iii) $t_0 < t_{\mathrm{out}}$, so that integration is in the forward direction;

(iv) the evaluation of the functions $P_{i,j}$, Q_i and R_i is done at both the break-points and internally selected points for each element in turn, that is $P_{i,j}$, Q_i and R_i are evaluated twice at each break-point. Any discontinuities in these functions **must** therefore be at one or more of the mesh points;

(v) at least one of the functions $P_{i,j}$ must be non-zero so that there is a time derivative present in the PDE problem;

(vi) if $m > 0$ and $x_1 = 0.0$, which is the left boundary point, then it must be ensured that the PDE solution is bounded at this point. This can be done either by specifying the solution at $x = 0.0$ or by specifying a zero flux there, that is $\beta_i = 1.0$ and $\gamma_i = 0.0$.

The parabolic equations are approximated by a system of ODEs in time for the values of U_i at the mesh points. This ODE system is obtained by approximating the PDE solution between each pair of break-points by a Chebyshev polynomial of degree NPOLY. The interval between each pair of break-points is treated by D03PJF/D03PJA as an element, and on this element, a polynomial and its space and time derivatives are made to satisfy the system of PDEs at NPOLY $-$ 1 spatial points, which are chosen internally by the code and the break-points. The user-defined break-points and the internally selected points together define the mesh. The smallest value that NPOLY can take is one, in which case, the solution is approximated by piecewise linear polynomials between consecutive break-points and the method is similar to an ordinary finite element method.

In total there are $(\text{NBKPTS} - 1) \times \text{NPOLY} + 1$ mesh points in the spatial direction, and $\text{NPDE} \times ((\text{NBKPTS} - 1) \times \text{NPOLY} + 1) + \text{NCODE}$ ODEs in the time direction; one ODE at each break-point for each PDE component, NPOLY $-$ 1 ODEs for each PDE component between each pair of break-points, and NCODE coupled ODEs. The system is then integrated forwards in time using a Backward Differentiation Formula (BDF) method or a Theta method.

4 References

Berzins M (1990) Developments in the NAG Library software for parabolic equations *Scientific Software Systems* (ed J C Mason and M G Cox) 59–72 Chapman and Hall

Berzins M and Dew P M (1991) Algorithm 690: Chebyshev polynomial software for elliptic-parabolic systems of PDEs *ACM Trans. Math. Software* **17** 178–206

Berzins M, Dew P M and Furzeland R M (1988) Software tools for time-dependent equations in simulation and optimisation of large systems *Proc. IMA Conf. Simulation and Optimization* (ed A J Osiadcz) 35–50 Clarendon Press, Oxford

Berzins M and Furzeland R M (1992) An adaptive theta method for the solution of stiff and nonstiff differential equations *Appl. Numer. Math.* **9** 1–19

Zaturska N B, Drazin P G and Banks W H H (1988) On the flow of a viscous fluid driven along a channel by a suction at porous walls *Fluid Dynamics Research* **4**

5 Parameters

1: NPDE – INTEGER *Input*

On entry: the number of PDEs to be solved.

Constraint: NPDE ≥ 1.

2: M – INTEGER *Input*

On entry: the co-ordinate system used:

M $= 0$

Indicates Cartesian co-ordinates.

M $= 1$

Indicates cylindrical polar co-ordinates.

M $= 2$

Indicates spherical polar co-ordinates.

Constraint: $0 \leq \text{M} \leq 2$.

3: TS – ***real*** *Input/Output*

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

On exit: the value of t corresponding to the solution values in U. Normally TS = TOUT.

Constraint: TS $<$ TOUT.

4: TOUT – ***real*** *Input*

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

5: PDEDEF – SUBROUTINE, supplied by the user. *External Procedure*

PDEDEF must compute the functions $P_{i,j}$, Q_i and R_i which define the system of PDEs. The functions may depend on x, t, U, U_x and V; Q_i may depend linearly on $\dot{\mathrm{V}}$. The functions must be evaluated at a set of points.

The specification of PDEDEF for D03PJF is:

```
      SUBROUTINE PDEDEF(NPDE, T, X, NPTL, U, UX, NCODE, V, VDOT, P, Q, R,
     1                  IRES)
      INTEGER           NPDE, NPTL, NCODE, IRES
      real              T, X(NPTL), U(NPDE,NPTL), UX(NPDE,NPTL), V(*),
     1                  VDOT(*), P(NPDE,NPDE,NPTL), Q(NPDE,NPTL),
     2                  R(NPDE,NPTL)
```

The specification of PDEDEF for D03PJA is:

```
      SUBROUTINE PDEDEF(NPDE, T, X, NPTL, U, UX, NCODE, V, VDOT, P, Q, R,
     1                  IRES, IUSER, RUSER)
      INTEGER           NPDE, NPTL, NCODE, IRES, IUSER(*)
      real              T, X(NPTL), U(NPDE,NPTL), UX(NPDE,NPTL), V(*),
     1                  VDOT(*), P(NPDE,NPDE,NPTL), Q(NPDE,NPTL),
     2                  R(NPDE,NPTL), RUSER(*)
```

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.

3: X(NPTL) – ***real*** array *Input*

On entry: contains a set of mesh points at which $P_{i,j}$, Q_i and R_i are to be evaluated. X(1) and X(NPTL) contain successive user-supplied break-points and the elements of the array will satisfy X(1) $<$ X(2) $<$ ... $<$ X(NPTL).

4: NPTL – INTEGER *Input*

On entry: the number of points at which evaluations are required (the value of NPOLY + 1).

5: U(NPDE,NPTL) – ***real*** array *Input*

On entry: U(i,j) contains the value of the component $U_i(x,t)$ where $x = \mathrm{X}(j)$, for $i = 1,2,\ldots,\mathrm{NPDE}$; $j = 1,2,\ldots,\mathrm{NPTL}$.

6: UX(NPDE,NPTL) – ***real*** array *Input*

On entry: UX(i,j) contains the value of the component $(\partial U_i(x,t))/(\partial x)$ where $x = \mathrm{X}(j)$, for $i = 1,2,\ldots,\mathrm{NPDE}$; $j = 1,2,\ldots,\mathrm{NPTL}$.

7: NCODE – INTEGER *Input*

On entry: the number of coupled ODEs in the system.

8: V(∗) – ***real*** array *Input*

On entry: V(i) contains the value of component $V_i(t)$, for $i = 1, 2, \ldots,$ NCODE.

9: VDOT(∗) – ***real*** array *Input*

On entry: VDOT(i) contains the value of component $\dot{V}_i(t)$, for $i = 1, 2, \ldots,$ NCODE.

Note: $\dot{V}_i(t)$, for $i = 1, 2, \ldots,$ NCODE, may only appear linearly in Q_j, for $j = 1, 2, \ldots,$ NPDE.

10: P(NPDE,NPDE,NPTL) – ***real*** array *Output*

On exit: P(i, j, k) must be set to the value of $P_{i,j}(x, t, U, U_x, V)$ where $x =$ X(k), for $i, j = 1, 2, \ldots,$ NPDE; $k = 1, 2, \ldots,$ NPTL.

11: Q(NPDE,NPTL) – ***real*** array *Output*

On exit: Q(i, j) must be set to the value of $Q_i(x, t, U, U_x, V, \dot{V})$ where $x =$ X(j), for $i = 1, 2, \ldots,$ NPDE; $j = 1, 2, \ldots,$ NPTL.

12: R(NPDE,NPTL) – ***real*** array *Output*

On exit: R(i) must be set to the value of $R_i(x, t, U, U_x, V)$ where $x =$ X(i), for $i = 1, 2, \ldots,$ NPDE; $j = 1, 2, \ldots,$ NPTL.

13: IRES – INTEGER *Input/Output*

On entry: set to -1 or 1.

On exit: should usually remain unchanged. However, the user may set IRES to force the integration routine to take certain actions as described below:

IRES $= 2$

Indicates to the integrator that control should be passed back immediately to the calling (sub)program with the error indicator set to IFAIL $= 6$.

IRES $= 3$

Indicates to the integrator that the current time step should be abandoned and a smaller time step used instead. The user may wish to set IRES $= 3$ when a physically meaningless input or output value has been generated. If the user consecutively sets IRES $= 3$, then D03PJF/D03PJA returns to the calling (sub)program with the error indicator set to IFAIL $= 4$.

Note: *the following are additional parameters for specific use of PDEDEF with D03PJA. Users of D03PJF therefore need not read the remainder of this description.*

14: IUSER(∗) – INTEGER array *User Workspace*
15: RUSER(∗) – ***real*** array *User Workspace*

PDEDEF is called from D03PJA with the parameters IUSER and RUSER as supplied to D03PJA. You are free to use the arrays IUSER and RUSER to supply information to PDEDEF.

PDEDEF must be declared as EXTERNAL in the (sub)program from which D03PJF/D03PJA is called. Parameters denoted as *Input* must **not** be changed by this procedure.

6: BNDARY – SUBROUTINE, supplied by the user. *External Procedure*

BNDARY must compute the functions β_i and γ_i which define the boundary conditions as in equation (4).

The specification of BNDARY for D03PJF is:

```
SUBROUTINE BNDARY(NPDE, T, U, UX, NCODE, V, VDOT, IBND, BETA, GAMMA,
1                 IRES)
INTEGER           NPDE, NCODE, IBND, IRES
real              T, U(NPDE), UX(NPDE), V(*), VDOT(*), BETA(NPDE),
1                 GAMMA(NPDE)
```

The specification of BNDARY for D03PJA is:

```
SUBROUTINE BNDARY(NPDE, T, U, UX, NCODE, V, VDOT, IBND, BETA, GAMMA,
1                 IRES, IUSER, RUSER)
INTEGER           NPDE, NCODE, IBND, IRES, IUSER(*)
real              T, U(NPDE), UX(NPDE), V(*), VDOT(*), BETA(NPDE),
1                 GAMMA(NPDE), RUSER(*)
```

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.

3: U(NPDE) – ***real*** array *Input*

On entry: U(i) contains the value of the component $U_i(x,t)$ at the boundary specified by IBND, for $i = 1,2,\ldots,$NPDE.

4: UX(NPDE) – ***real*** array *Input*

On entry: UX(i) contains the value of the component $(\partial U_i(x,t))/(\partial x)$ at the boundary specified by IBND, for $i = 1,2,\ldots,$NPDE.

5: NCODE – INTEGER *Input*

On entry: the number of coupled ODEs in the system.

6: V(*) – ***real*** array *Input*

On entry: V(i) contains the value of component $V_i(t)$, for $i = 1,2,\ldots,$NCODE.

7: VDOT(*) – ***real*** array *Input*

On entry: VDOT(i) contains the value of component $\dot{V}_i(t)$, for $i = 1,2,\ldots,$NCODE.

Note: $\dot{V}_i(t)$, for $i = 1,2,\ldots,$NCODE, may only appear linearly in Q_j, for $j = 1,2,\ldots,$NPDE.

8: IBND – INTEGER *Input*

On entry: specifies which boundary conditions are to be evaluated. If IBND $= 0$, then BNDARY must set up the coefficients of the left-hand boundary, $x = a$. If IBND $\neq 0$, then BNDARY must set up the coefficients of the right-hand boundary, $x = b$.

9: BETA(NPDE) – ***real*** array *Output*

On exit: BETA(i) must be set to the value of $\beta_i(x,t)$ at the boundary specified by IBND, for $i = 1,2,\ldots,$NPDE.

10: GAMMA(NPDE) – ***real*** array *Output*

On exit: GAMMA(i) must be set to the value of $\gamma_i(x,t,U,U_x,V,\dot{V})$ at the boundary specified by IBND, for $i = 1,2,\ldots,$NPDE.

11: IRES – INTEGER *Input/Output*

On entry: set to -1 or 1.

On exit: should usually remain unchanged. However, the user may set IRES to force the integration routine to take certain actions as described below:

IRES $= 2$

Indicates to the integrator that control should be passed back immediately to the calling (sub)program with the error indicator set to IFAIL $= 6$.

IRES $= 3$

Indicates to the integrator that the current time step should be abandoned and a smaller time step used instead. The user may wish to set IRES $= 3$ when a physically meaningless input or output value has been generated. If the user consecutively sets IRES $= 3$, then D03PJF/D03PJA returns to the calling (sub)program with the error indicator set to IFAIL $= 4$.

Note: *the following are additional parameters for specific use of BNDARY with D03PJA. Users of D03PJF therefore need not read the remainder of this description.*

12: IUSER(*) – INTEGER array *User Workspace*
13: RUSER(*) – ***real*** array *User Workspace*

BNDARY is called from D03PJA with the parameters IUSER and RUSER as supplied to D03PJA. You are free to use the arrays IUSER and RUSER to supply information to BNDARY.

BNDARY must be declared as EXTERNAL in the (sub)program from which D03PJF/D03PJA is called. Parameters denoted as *Input* must **not** be changed by this procedure.

7: U(NEQN) – ***real*** array *Output*

On exit: the computed solution $U_i(x_j,t)$, for $i = 1,2,\ldots,$NPDE; $j = 1,2,\ldots,$NPTS and $V_k(t)$, for $k = 1,2,\ldots,$NCODE, evaluated at $t =$ S, as follows:

U(NPDE $\times (j-1) + i$) contain $U_i(x_j,t)$, for $i = 1,2,\ldots,$NPDE; $j = 1,2,\ldots,$NPTS and U(NPTS $\times$ NPDE $+ i$) contain $V_i(t)$, for $i = 1,2,\ldots,$NCODE.

8: NBKPTS – INTEGER *Input*

On entry: the number of break-points in the interval $[a,b]$.

Constraint: NBKPTS ≥ 2.

9: XBKPTS(NBKPTS) – ***real*** array *Input*

On entry: the values of the break-points in the space direction. XBKPTS(1) must specify the left-hand boundary, a, and XBKPTS(NBKPTS) must specify the right-hand boundary, b.

Constraint: XBKPTS(1) $<$ XBKPTS(2) $< \cdots <$ XBKPTS(NBKPTS).

10: NPOLY – INTEGER *Input*

On entry: the degree of the Chebyshev polynomial to be used in approximating the PDE solution between each pair of break-points.

Constraint: $1 \leq$ NPOLY ≤ 49.

11: NPTS – INTEGER *Input*

On entry: the number of mesh points in the interval $[a, b]$.

Constraint: NPTS $= (\text{NBKPTS} - 1) \times \text{NPOLY} + 1$.

12: X(NPTS) – ***real*** array *Output*

On exit: the mesh points chosen by D03PJF/D03PJA in the spatial direction. The values of X will satisfy $\text{X}(1) < \text{X}(2) < \ldots < \text{X}(\text{NPTS})$.

13: NCODE – INTEGER *Input*

On entry: the number of coupled ODE components.

Constraint: NCODE ≥ 0.

14: ODEDEF – SUBROUTINE, supplied by the user. *External Procedure*

ODEDEF must evaluate the functions F, which define the system of ODEs, as given in (3). If the user wishes to compute the solution of a system of PDEs only (NCODE $= 0$), ODEDEF must be the dummy routine D03PCK for D03PJF or D53PCK for D03PJA. (D03PCK and D53PCK are included in the NAG Fortran Library; however, their names may be implementation-dependent: see the Users' Note for your implementation for details.)

The specification of ODEDEF for D03PJF is:

```
      SUBROUTINE ODEDEF(NPDE, T, NCODE, V, VDOT, NXI, XI, UCP, UCPX, RCP,
     1                  UCPT, UCPTX, F, IRES)
      INTEGER           NPDE, NCODE, NXI, IRES
      real              T, V(*), VDOT(*), XI(*), UCP(NPDE,*),
     1                  UCPX(NPDE,*), RCP(NPDE,*), UCPT(NPDE,*),
     2                  UCPTX(NPDE,*), F(*)
```

The specification of ODEDEF for D03PJA is:

```
      SUBROUTINE ODEDEF(NPDE, T, NCODE, V, VDOT, NXI, XI, UCP, UCPX, RCP,
     1                  UCPT, UCPTX, F, IRES, IUSER, RUSER)
      INTEGER           NPDE, NCODE, NXI, IRES, IUSER(*)
      real              T, V(*), VDOT(*), XI(*), UCP(NPDE,*),
     1                  UCPX(NPDE,*), RCP(NPDE,*), UCPT(NPDE,*),
     2                  UCPTX(NPDE,*), F(*), RUSER(*)
```

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.

3: NCODE – INTEGER *Input*

On entry: the number of coupled ODEs in the system.

4: V(*) – ***real*** array *Input*

On entry: V(i) contains the value of component $V_i(t)$, for $i = 1, 2, \ldots, \text{NCODE}$.

5: VDOT(*) – ***real*** array *Input*

On entry: VDOT(i) contains the value of component $\dot{V}_i(t)$, for $i = 1, 2, \ldots, \text{NCODE}$.

6: NXI – INTEGER *Input*

On entry: the number of ODE/PDE coupling points.

7: XI(*) – ***real*** array *Input*

On entry: XI(i) contains the ODE/PDE coupling points, ξ_i, $i = 1, 2, \ldots, \text{NXI}$.

8: UCP(NPDE,*) – ***real*** array *Input*

On entry: UCP(i, j) contains the value of $U_i(x, t)$ at the coupling point $x = \xi_j$, for $i = 1, 2, \ldots, \text{NPDE}$; $j = 1, 2, \ldots, \text{NXI}$.

9: UCPX(NPDE,*) – ***real*** array *Input*

On entry: UCPX(i, j) contains the value of $(\partial U_i(x, t))/(\partial x)$ at the coupling point $x = \xi_j$, for $i = 1, 2, \ldots, \text{NPDE}$; $j = 1, 2, \ldots, \text{NXI}$.

10: RCP(NPDE,*) – ***real*** array *Input*

On entry: RCP(i, j) contains the value of the flux R_i at the coupling point $x = \xi_j$, for $i = 1, 2, \ldots, \text{NPDE}$; $j = 1, 2, \ldots, \text{NXI}$.

11: UCPT(NPDE,*) – ***real*** array *Input*

On entry: UCPT(i, j) contains the value of $(\partial U_i)/(\partial t)$ at the coupling point $x = \xi_j$, for $i = 1, 2, \ldots, \text{NPDE}$; $j = 1, 2, \ldots, \text{NXI}$.

12: UCPTX(NPDE,*) – ***real*** array *Input*

On entry: UCPTX(i, j) contains the value of $(\partial^2 U_i)/(\partial x \partial t)$ at the coupling point $x = \xi_j$, for $i = 1, 2, \ldots, \text{NPDE}$; $j = 1, 2, \ldots, \text{NXI}$.

13: F(*) – ***real*** array *Output*

On exit: F(i) must contain the ith component of F, for $i = 1, 2, \ldots, \text{NCODE}$, where F is defined as

$$F = G - A\dot{V} - B\begin{pmatrix} U_t^* \\ U_{xt}^* \end{pmatrix},$$

or

$$F = -A\dot{V} - B\begin{pmatrix} U_t^* \\ U_{xt}^* \end{pmatrix}.$$

The definition of F is determined by the input value of IRES.

14: IRES – INTEGER *Input/Output*

On entry: the form of F that must be returned in the array F. If IRES $= 1$, then the equation (5) above must be used. If IRES $= -1$, then the equation (6) above must be used.

On exit: should usually remain unchanged. However, the user may reset IRES to force the integration routine to take certain actions as described below:

IRES $= 2$

Indicates to the integrator that control should be passed back immediately to the calling (sub)program with the error indicator set to IFAIL $= 6$.

IRES $= 3$

Indicates to the integrator that the current time step should be abandoned and a smaller time step used instead. The user may wish to set IRES $= 3$ when a physically meaningless input or output value has been generated. If the user consecutively sets IRES $= 3$, then D03PJF/D03PJA returns to the calling (sub)program with the error indicator set to IFAIL $= 4$.

Note: *the following are additional parameters for specific use of ODEDEF with D03PJA. Users of D03PJF therefore need not read the remainder of this description.*

15: IUSER(*) – INTEGER array *User Workspace*
16: RUSER(*) – ***real*** array *User Workspace*

ODEDEF is called from D03PJA with the parameters IUSER and RUSER as supplied to D03PJA. You are free to use the arrays IUSER and RUSER to supply information to ODEDEF.

ODEDEF must be declared as EXTERNAL in the (sub)program from which D03PJF/D03PJA is called. Parameters denoted as *Input* must **not** be changed by this procedure.

15: NXI – INTEGER *Input*

On entry: the number of ODE/PDE coupling points.

Constraints:

$\text{NXI} = 0$ if $\text{NCODE} = 0$,
$\text{NXI} \geq 0$ if $\text{NCODE} > 0$.

16: XI(*) – ***real*** array *Input*

Note: the dimension of the array XI must be at least $\max(1, \text{NXI})$.

On entry: $\text{XI}(i)$, $i = 1, 2, \ldots, \text{NXI}$, must be set to the ODE/PDE coupling points.

Constraint: $\text{XBKPTS}(1) \leq \text{XI}(1) < \text{XI}(2) < \cdots < \text{XI}(\text{NXI}) \leq \text{XBKPTS}(\text{NBKPTS})$.

17: NEQN – INTEGER *Input*

On entry: the number of ODEs in the time direction.

Constraint: $\text{NEQN} = \text{NPDE} \times \text{NPTS} + \text{NCODE}$.

18: UVINIT – SUBROUTINE, supplied by the user. *External Procedure*

UVINIT must compute the initial values of the PDE and the ODE components $U_i(x_j, t_0)$, for $i = 1, 2, \ldots, \text{NPDE}$; $j = 1, 2, \ldots, \text{NPTS}$, and $V_k(t_0)$, for $k = 1, 2, \ldots, \text{NCODE}$.

The specification of UVINIT for D03PJF is:

```
SUBROUTINE UVINIT(NPDE, NPTS, X, U, NCODE, V)
INTEGER           NPDE, NPTS, NCODE
real              X(NPTS), U(NPDE,NPTS), V(*)
```

The specification of UVINIT for D03PJA is:

```
SUBROUTINE UVINIT(NPDE, NPTS, X, U, NCODE, V, IUSER, RUSER)
INTEGER           NPDE, NPTS, NCODE, IUSER(*)
real              X(NPTS), U(NPDE,NPTS), V(*), RUSER(*)
```

1: NPDE – INTEGER *Input*

On entry: the number of PDEs in the system.

2: NPTS – INTEGER *Input*

On entry: the number of mesh points in the interval $[a, b]$.

3: X(NPTS) – ***real*** array *Input*

On entry: $\text{x}(i)$, for $i = 1, 2, \ldots, \text{npts}$, contains the current values of the space variable x_i.

4: U(NPDE,NPTS) – ***real*** array *Output*

On exit: U(i,j) contains the value of the component $U_i(x_j,t_0)$, for $i = 1,2,\ldots,$NPDE; $j = 1,2,\ldots,$NPTS.

5: NCODE – INTEGER *Input*

On entry: the number of coupled ODEs in the system.

6: V(∗) – ***real*** array *Input*

On exit: V(i) contains the value of component $V_i(t_0)$, for $i = 1,2,\ldots,$NCODE.

Note: *the following are additional parameters for specific use of UVINIT with D03PJA. Users of D03PJF therefore need not read the remainder of this description.*

7: IUSER(∗) – INTEGER array *User Workspace*
8: RUSER(∗) – ***real*** array *User Workspace*

UVINIT is called from D03PJA with the parameters IUSER and RUSER as supplied to D03PJA. You are free to use the arrays IUSER and RUSER to supply information to UVINIT.

UVINIT must be declared as EXTERNAL in the (sub)program from which D03PJF/D03PJA is called. Parameters denoted as *Input* must **not** be changed by this procedure.

19: RTOL(∗) – ***real*** array *Input*

Note: the dimension of the array RTOL must be at least 1 if ITOL $= 1$ or 2 and at least NEQN if ITOL $= 3$ or 4.

On entry: the relative local error tolerance.

Constraint: RTOL$(i) \geq 0$ for all relevant i.

20: ATOL(∗) – ***real*** array *Input*

Note: the dimension of the array ATOL must be at least 1 if ITOL $= 1$ or 3 and at least NEQN if ITOL $= 2$ or 4.

On entry: the absolute local error tolerance.

Constraint: ATOL$(i) \geq 0$ for all relevant i.

21: ITOL – INTEGER *Input*

On entry: a value to indicate the form of the local error test. ITOL indicates to D03PJF/D03PJA whether to interpret either or both of RTOL or ATOL as a vector or scalar. The error test to be satisfied is $\|e_i/w_i\| < 1.0$, where w_i is defined as follows:

ITOL	RTOL	ATOL	w_i
1	scalar	scalar	RTOL$(1) \times \lvert U_i\rvert +$ ATOL(1)
2	scalar	vector	RTOL$(1) \times \lvert U_i\rvert +$ ATOL(i)
13	vector	scalar	RTOL$(i) \times \lvert U_i\rvert +$ ATOL(1)
14	vector	vector	RTOL$(i) \times \lvert U_i\rvert +$ ATOL(i)

In the above, e_i denotes the estimated local error for the ith component of the coupled PDE/ODE system in time, U(i), for $i = 1,2,\ldots,$NEQN.

The choice of norm used is defined by the parameter NORM, see below.

Constraint: $1 \leq$ ITOL ≤ 4.

22: NORM – CHARACTER*1 *Input*

On entry: the type of norm to be used. Two options are available:

NORM = 'M'

Maximum norm.

NORM = 'A'

Averaged L_2 norm.

If $\mathrm{U}_{\mathrm{norm}}$ denotes the norm of the vector U of length NEQN, then for the averaged L_2 norm

$$\mathrm{U}_{\mathrm{norm}} = \sqrt{\frac{1}{\mathrm{NEQN}} \sum_{i=1}^{\mathrm{NEQN}} (\mathrm{U}(i)/w_i)^2},$$

while for the maximum norm

$$\mathrm{U}_{\mathrm{norm}} = \max_i |\mathrm{U}(i)/w_i|.$$

See the description of the ITOL parameter for the formulation of the weight vector w.

Constraint: NORM = 'M' or 'A'.

23: LAOPT – CHARACTER*1 *Input*

On entry: the type of matrix algebra required. The possible choices are:

LAOPT = 'F'

Full matrix routines to be used.

LAOPT = 'B'

Banded matrix routines to be used.

LAOPT = 'S'

Sparse matrix routines to be used.

Constraint: LAOPT = 'F', 'B' or 'S'.

Note: the user is recommended to use the banded option when no coupled ODEs are present (i.e., NCODE = 0).

24: ALGOPT(30) – ***real*** array *Input*

On entry: ALGOPT may be set to control various options available in the integrator. If the user wishes to employ all the default options, then ALGOPT(1) should be set to 0.0. Default values will also be used for any other elements of ALGOPT set to zero. The permissible values, default values, and meanings are as follows:

ALGOPT(1) selects the ODE integration method to be used. If ALGOPT$(1) = 1.0$, a BDF method is used and if ALGOPT$(1) = 2.0$, a Theta method is used. The default value is ALGOPT$(1) = 1.0$.

If ALGOPT$(1) = 2.0$, then ALGOPT(i), for $i = 2, 3, 4$ are not used.

ALGOPT(2) specifies the maximum order of the BDF integration formula to be used. ALGOPT(2) may be 1.0, 2.0, 3.0, 4.0 or 5.0. The default value is ALGOPT$(2) = 5.0$.

ALGOPT(3) specifies what method is to be used to solve the system of nonlinear equations arising on each step of the BDF method. If ALGOPT$(3) = 1.0$ a modified Newton iteration is used and if ALGOPT$(3) = 2.0$ a functional iteration method is used. If functional iteration is selected and the integrator encounters difficulty, then there is an automatic switch to the modified Newton iteration. The default value is ALGOPT$(3) = 1.0$.

ALGOPT(4) specifies whether or not the Petzold error test is to be employed. The Petzold error test results in extra overhead but is more suitable when algebraic equations are

present, such as $P_{i,j} = 0.0$, for $j = 1, 2, \ldots,$ NPDE for some i or when there is no $\dot{V}_i(t)$ dependence in the coupled ODE system. If ALGOPT(4) $= 1.0$, then the Petzold test is used. If ALGOPT(4) $= 2.0$, then the Petzold test is not used. The default value is ALGOPT(4) $= 1.0$.

If ALGOPT(1) $= 1.0$, then ALGOPT(i), for $i = 5, 6, 7$ are not used.

ALGOPT(5) specifies the value of Theta to be used in the Theta integration method. $0.51 \leq$ ALGOPT(5) ≤ 0.99.

The default value is ALGOPT(5) $= 0.55$.

ALGOPT(6) specifies what method is to be used to solve the system of nonlinear equations arising on each step of the Theta method. If ALGOPT(6) $= 1.0$, a modified Newton iteration is used and if ALGOPT(6) $= 2.0$, a functional iteration method is used. The default value is ALGOPT(6) $= 1.0$.

ALGOPT(7) specifies whether or not the integrator is allowed to switch automatically between modified Newton and functional iteration methods in order to be more efficient. If ALGOPT(7) $= 1.0$, then switching is allowed and if ALGOPT(7) $= 2.0$, then switching is not allowed. The default value is ALGOPT(7) $= 1.0$.

ALGOPT(11) specifies a point in the time direction, t_{crit}, beyond which integration must not be attempted. The use of t_{crit} is described under the parameter ITASK. If ALGOPT(1) $\neq 0.0$, a value of 0.0 for ALGOPT(11), say, should be specified even if ITASK subsequently specifies that t_{crit} will not be used.

ALGOPT(12) specifies the minimum absolute step size to be allowed in the time integration. If this option is not required, ALGOPT(12) should be set to 0.0.

ALGOPT(13) specifies the maximum absolute step size to be allowed in the time integration. If this option is not required, ALGOPT(13) should be set to 0.0.

ALGOPT(14) specifies the initial step size to be attempted by the integrator. If ALGOPT(14) $= 0.0$, then the initial step size is calculated internally.

ALGOPT(15) specifies the maximum number of steps to be attempted by the integrator in any one call. If ALGOPT(15) $= 0.0$, then no limit is imposed.

ALGOPT(23) specifies what method is to be used to solve the nonlinear equations at the initial point to initialise the values of U, U_t, V and $\dot{V}$. If ALGOPT(23) $= 1.0$, a modified Newton iteration is used and if ALGOPT(23) $= 2.0$, functional iteration is used. The default value is ALGOPT(23) $= 1.0$.

ALGOPT(29) and ALGOPT(30) are used only for the sparse matrix algebra option, LAOPT = 'S'.

ALGOPT(29) governs the choice of pivots during the decomposition of the first Jacobian matrix. It should lie in the range $0.0 <$ ALGOPT(29) < 1.0, with smaller values biasing the algorithm towards maintaining sparsity at the expense of numerical stability. If ALGOPT(29) lies outside this range then the default value is used. If the routines regard the Jacobian matrix as numerically singular then increasing ALGOPT(29) towards 1.0 may help, but at the cost of increased fill-in. The default value is ALGOPT(29) $= 0.1$.

ALGOPT(30) is used as a relative pivot threshold during subsequent Jacobian decompositions (see ALGOPT(29)) below which an internal error is invoked. If ALGOPT(30) is greater than 1.0 no check is made on the pivot size, and this may be a necessary option if the Jacobian is found to be numerically singular (see ALGOPT(29)). The default value is ALGOPT(30) $= 0.0001$.

25: W(NW) – ***real*** array *Workspace*

26: NW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which D03PJF/D03PJA is called. Its size depends on the type of matrix algebra selected:

LAOPT = 'F',

$$\text{NW} \geq \text{NEQN} \times \text{NEQN} + \text{NEQN} + \text{NWKRES} + \text{LENODE},$$

LAOPT = 'B',

$$\text{NW} \geq (3 \times \text{MLU} + 1) \times \text{NEQN} + \text{NWKRES} + \text{LENODE},$$

LAOPT = 'S',

$$\text{NW} \geq 4 \times \text{NEQN} + 11 \times \text{NEQN}/2 + 1 + \text{NWKRES} + \text{LENODE}.$$

Where MLU = the lower or upper half bandwidths, and $\text{MLU} = (\text{NPOLY} + 1) \times \text{NPDE} - 1$, for PDE problems only, and, $\text{MLU} = \text{NEQN} - 1$, for coupled PDE/ODE problems.

$$\text{NWKRES} = 3 \times (\text{NPOLY} + 1)^2 + (\text{NPOLY} + 1) \times [\text{NPDE}^2 + 6 \times \text{NPDE} + \text{NBKPTS} + 1] + 8 \times \text{NPDE} + \text{NXI} \times (5 \times \text{NPDE} + 1) + \text{NCODE} + 3,$$

when $\text{NCODE} > 0$, and $\text{NXI} > 0$.

$$\text{NWKRES} = 3 \times (\text{NPOLY} + 1)^2 + (\text{NPOLY} + 1) \times [\text{NPDE}^2 + 6 \times \text{NPDE} + \text{NBKPTS} + 1] + 13 \times \text{NPDE} + \text{NCODE} + 4,$$

when $\text{NCODE} > 0$, and $\text{NXI} = 0$.

$$\text{NWKRES} = 3 \times (\text{NPOLY} + 1)^2 + (\text{NPOLY} + 1) \times [\text{NPDE}^2 + 6 \times \text{NPDE} + \text{NBKPTS} + 1] + 13 \times \text{NPDE} + 5,$$

when $\text{NCODE} = 0$.

$\text{LENODE} = (6 + \text{int}(\text{ALGOPT}(2))) \times \text{NEQN} + 50$, when the BDF method is used and,

$\text{LENODE} = 9 \times \text{NEQN} + 50$, when a Theta method is used.

Note: when using the sparse option, the value of NW may be too small when supplied to the integrator. An estimate of the minimum size of NW is printed on the current error message unit if $\text{ITRACE} > 0$ and the routine returns with $\text{IFAIL} = 15$.

27: IW(NIW) – INTEGER array *Output*

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

IW(1) contains the number of steps taken in time.

IW(2) contains the number of residual evaluations of the resulting ODE system used. One such evaluation involves evaluating the PDE functions at all the mesh points, as well as one evaluation of the functions in the boundary conditions.

IW(3) contains the number of Jacobian evaluations performed by the time integrator.

IW(4) contains the order of the ODE method last used in the time integration.

IW(5) contains the number of Newton iterations performed by the time integrator. Each iteration involves residual evaluation of the resulting ODE system followed by a back-substitution using the LU decomposition of the Jacobian matrix.

28: NIW – INTEGER *Input*

On entry: the dimension of the array IW. Its size depends on the type of matrix algebra selected:

29: ITASK – INTEGER *Input*

On entry: specifies the task to be performed by the ODE integrator. The permitted values of ITASK and their meanings are detailed below:

ITASK = 1

Normal computation of output values U at $t = \text{TOUT}$.

ITASK = 2

One step and return.

ITASK = 3

Stop at first internal integration point at or beyond $t =$ TOUT.

ITASK = 4

Normal computation of output values U at $t =$ TOUT but without overshooting $t = t_{\text{crit}}$ where t_{crit} is described under the parameter ALGOPT.

ITASK = 5

Take one step in the time direction and return, without passing t_{crit}, where t_{crit} is described under the parameter ALGOPT.

Constraint: $1 \leq \text{ITASK} \leq 5$.

30: ITRACE – INTEGER *Input*

On entry: the level of trace information required from D03PJF/D03PJA and the underlying ODE solver. 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 from the PDE solver are printed on the current error message unit (see X04AAF). If ITRACE > 0, then output from the underlying ODE solver is printed on the current advisory message unit (see X04ABF). This output contains details of Jacobian entries, the nonlinear iteration and the time integration during the computation of the ODE system. The advisory messages are given in greater detail as ITRACE increases. Users are advised to set ITRACE $= 0$, unless they are experienced with Chapter D02M/N.

31: IND – INTEGER *Input/Output*

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

IND = 0

Starts or restarts the integration in time.

IND = 1

Continues the integration after an earlier exit from the routine. In this case, only the parameters TOUT and IFAIL should be reset between calls to D03PJF/D03PJA.

Constraint: $0 \leq \text{IND} \leq 1$.

On exit: IND = 1.

32: IFAIL – INTEGER *Input/Output*

Note: *for D03PJA, IFAIL does not occur in this position in the parameter list. See the additional parameters described below.*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

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

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, for users not familiar with this parameter the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

Note: *the following are additional parameters for specific use with D03PJA. Users of D03PJF therefore need not read the remainder of this section.*

32: IUSER(*) – INTEGER array *User Workspace*

Note: the first dimension of the array IUSER must be at least 1.

IUSER is not used by D03PJA, but is passed directly to the external procedures PDEDEF, BNDARY, ODEDEF and UVINIT and may be used to pass information to these routines.

33: RUSER(*) – ***real*** array *User Workspace*

Note: the first dimension of the array RUSER must be at least 1.

RUSER is not used by D03PJA, but is passed directly to the external procedures PDEDEF, BNDARY, ODEDEF and UVINIT and may be used to pass information to these routines.

34: CWSAV(10) – CHARACTER*80 array *Workspace*

35: LWSAV(100) – LOGICAL array *Workspace*

36: IWSAV(505) – INTEGER array *Workspace*

37: RWSAV(1100) – ***real*** array *Workspace*

38: IFAIL – INTEGER *Input/Output*

Note: see the parameter description for IFAIL above.

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, TOUT $-$ TS is too small,
or ITASK $\neq 1$, 2, 3, 4 or 5,
or M $\neq 0$, 1 or 2,
or at least one of the coupling point in array XI is outside the interval [XBKPTS(1),-XBKPTS(NBKPTS)],
or NPTS $\neq$ (NBKPTS $- 1$) $\times$ NPOLY $+ 1$,
or NBKPTS < 2,
or NPDE ≤ 0,
or NORM $\neq$ 'A' or 'M',
or ITOL $\neq 1$, 2, 3 or 4,
or NPOLY < 1 or NPOLY > 49,
or NCODE and NXI are incorrectly defined,
or NEQN $\neq$ NPDE $\times$ NPTS $+$ NCODE,
or LAOPT $\neq$ 'F', 'B' or 'S',
or IND $\neq 0$ or 1,
or incorrectly defined user break-points, XBKPTS$(i) \geq$ XBKPTS$(i+1)$, for some $i = 1, 2, \ldots,$ NBKPTS $- 1$,
or NW is too small,
or NIW is too small,
or the ODE integrator has not been correctly defined; check ALGOPT parameter.
or either an element of RTOL or ATOL < 0.0,
or all the elements of RTOL and ATOL are zero.

IFAIL $= 2$

The underlying ODE solver cannot make any further progress, with the values of ATOL and RTOL, across the integration range from the current point $t =$ TS. The components of U contain the computed values at the current point $t =$ TS.

IFAIL $= 3$

In the underlying ODE solver, there were repeated error test failures on an attempted step, before completing the requested task, but the integration was successful as far as $t =$ TS. The problem may have a singularity, or the error requirement may be inappropriate.

IFAIL = 4

In setting up the ODE system, the internal initialisation routine was unable to initialise the derivative of the ODE system. This could be due to the fact that IRES was repeatedly set to 3 in at least one of the user-supplied subroutines PDEDEF, BNDARY or ODEDEF, when the residual in the underlying ODE solver was being evaluated.

IFAIL = 5

In solving the ODE system, a singular Jacobian has been encountered. The user should check his problem formulation.

IFAIL = 6

When evaluating the residual in solving the ODE system, IRES was set to 2 in at least one of the user-supplied subroutines PDEDEF, BNDARY or ODEDEF. Integration was successful as far as $t =$ TS.

IFAIL = 7

The values of ATOL and RTOL are so small that the routine is unable to start the integration in time.

IFAIL = 8

In one of the user-supplied routines, PDEDEF, BNDARY or ODEDEF, IRES was set to an invalid value.

IFAIL = 9

A serious error has occurred in an internal call to D02NNF. Check problem specification and all parameters and array dimensions. Setting ITRACE = 1 may provide more information. If the problem persists, contact NAG.

IFAIL = 10

The required task has been completed, but it is estimated that a small change in ATOL and RTOL is unlikely to produce any change in the computed solution. (Only applies when the user is not operating in one step mode, that is when ITASK $\neq$ 2 or 5.)

IFAIL = 11

An error occurred during Jacobian formulation of the ODE system (a more detailed error description may be directed to the current error message unit).

IFAIL = 12

In solving the ODE system, the maximum number of steps specified in ALGOPT(15) has been taken.

IFAIL = 13

Some error weights w_i became zero during the time integration (see description of ITOL). Pure relative error control (ATOL(i) = 0.0) was requested on a variable (the ith) which has become zero. The integration was succesful as far as $t =$ TS.

IFAIL = 14

The flux function R_i was detected as depending on time derivatives, which is not permissible.

IFAIL = 15

When using the sparse option, the value of NIW or NW was not sufficient (more detailed information may be directed to the current error message unit).

7 Accuracy

The routine controls the accuracy of the integration in the time direction but not the accuracy of the approximation in space. The spatial accuracy depends on both the number of mesh points and on their distribution in space. In the time integration only the local error over a single step is controlled and so the accuracy over a number of steps cannot be guaranteed. The user should therefore test the effect of varying the accuracy parameter ATOL and RTOL.

8 Further Comments

The parameter specification allows the user to include equations with only first-order derivatives in the space direction but there is no guarantee that the method of integration will be satisfactory for such systems. The position and nature of the boundary conditions in particular are critical in defining a stable problem.

The time taken by the routine depends on the complexity of the parabolic system and on the accuracy requested.

9 Example

This problem provides a simple coupled system of one PDE and one ODE.

$$(V_1)^2 \frac{\partial U_1}{\partial t} - xV_1\dot{V}_1 \frac{\partial U_1}{\partial x} = \frac{\partial^2 U_1}{\partial x^2}$$

$$\dot{V}_1 = V_1 U_1 + \frac{\partial U_1}{\partial x} + 1 + t,$$

for $t \in [10^{-4}, 0.1 \times 2^i]$, $\quad i = 1, 2, \ldots, 5,\ x \in [0, 1]$.

The left boundary condition at $x = 0$ is

$$\frac{\partial U_1}{\partial x} = -V_1 \exp t.$$

The right boundary condition at $x = 1$ is

$$U_1 = -V_1\dot{V}_1.$$

The initial conditions at $t = 10^{-4}$ are defined by the exact solution:

$$V_1 = t, \quad \text{and} \quad U_1(x, t) = \exp\{t(1 - x)\} - 1.0, \quad x \in [0, 1],$$

and the coupling point is at $\xi_1 = 1.0$.

9.1 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.

Note: *the following program illustrates the use of D03PJF. An equivalent program illustrating the use of D03PJA is available with the supplied Library and is also available from the NAG web site.*

```
*     D03PJF Example Program Text
*     Mark 16 Revised. NAG Copyright 1993.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NBKPTS, NEL, NPDE, NPOLY, NPTS, NCODE, M, NXI,
     +                 NEQN, NIW, NPL1, NWKRES, LENODE, NW
      PARAMETER        (NBKPTS=11,NEL=NBKPTS-1,NPDE=1,NPOLY=2,
     +                 NPTS=NEL*NPOLY+1,NCODE=1,M=0,NXI=1,
     +                 NEQN=NPDE*NPTS+NCODE,NIW=24,NPL1=NPOLY+1,
     +                 NWKRES=3*NPL1*NPL1+NPL1*
     +                 (NPDE*NPDE+6*NPDE+NBKPTS+1)+8*NPDE+NXI*(5*NPDE+1)
     +                 +NCODE+3,LENODE=11*NEQN+50,
```

```
     +                NW=NEQN*NEQN+NEQN+NWKRES+LENODE)
*     .. Scalars in Common ..
      real             TS
*     .. Local Scalars ..
      real             TOUT
      INTEGER          I, IFAIL, IND, IT, ITASK, ITOL, ITRACE
      LOGICAL          THETA
      CHARACTER        LAOPT, NORM
*     .. Local Arrays ..
      real             ALGOPT(30), ATOL(1), EXY(NBKPTS), RTOL(1),
     +                 U(NEQN), W(NW), X(NPTS), XBKPTS(NBKPTS), XI(1)
      INTEGER          IW(NIW)
*     .. External Subroutines ..
      EXTERNAL         BNDARY, D03PJF, EXACT, ODEDEF, PDEDEF, UVINIT
*     .. Common blocks ..
      COMMON           /TAXIS/TS
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D03PJF Example Program Results'
      ITRACE = 0
      ITOL = 1
      ATOL(1) = 1.0e-4
      RTOL(1) = ATOL(1)
      WRITE (NOUT,99999) NPOLY, NEL
      WRITE (NOUT,99996) ATOL, NPTS
*
*     Set break-points
*
      DO 20 I = 1, NBKPTS
         XBKPTS(I) = (I-1.0e0)/(NBKPTS-1.0e0)
   20 CONTINUE
*
      XI(1) = 1.0e0
      NORM = 'A'
      LAOPT = 'F'
      IND = 0
      ITASK = 1
*
*     Set THETA to .TRUE. if the Theta integrator is required
*
      THETA = .FALSE.
      DO 40 I = 1, 30
         ALGOPT(I) = 0.0e0
   40 CONTINUE
      IF (THETA) THEN
         ALGOPT(1) = 2.0e0
      ELSE
         ALGOPT(1) = 0.0e0
      END IF
*
*     Loop over output value of t
*
      TS = 1.0e-4
      TOUT = 0.0e0
      WRITE (NOUT,99998) XBKPTS(1), XBKPTS(3), XBKPTS(5), XBKPTS(7),
     +  XBKPTS(11)
      DO 60 IT = 1, 5
         TOUT = 0.1e0*(2**IT)
         IFAIL = -1
*
         CALL D03PJF(NPDE,M,TS,TOUT,PDEDEF,BNDARY,U,NBKPTS,XBKPTS,NPOLY,
     +               NPTS,X,NCODE,ODEDEF,NXI,XI,NEQN,UVINIT,RTOL,ATOL,
     +               ITOL,NORM,LAOPT,ALGOPT,W,NW,IW,NIW,ITASK,ITRACE,
     +               IND,IFAIL)
*
*        Check against the exact solution
*
         CALL EXACT(TOUT,NBKPTS,XBKPTS,EXY)
         WRITE (NOUT,99997) TS
         WRITE (NOUT,99994) U(1), U(5), U(9), U(13), U(21), U(22)
         WRITE (NOUT,99993) EXY(1), EXY(3), EXY(5), EXY(7), EXY(11), TS
   60 CONTINUE
```

```
      WRITE (NOUT,99995) IW(1), IW(2), IW(3), IW(5)
      STOP
*
99999 FORMAT (' Degree of Polynomial =',I4,'   No. of elements =',I4,/)
99998 FORMAT ('  X        ',5F9.3,/)
99997 FORMAT (' T = ',F6.3)
99996 FORMAT (//'  Simple coupled PDE using BDF ',/'  Accuracy require',
     +       'ment =',e10.3,' Number of points = ',I4,/)
99995 FORMAT (' Number of integration steps in time = ',I6,/' Number o',
     +       'f function evaluations = ',I6,/' Number of Jacobian eval',
     +       'uations =',I6,/' Number of iterations = ',I6,/)
99994 FORMAT (1X,'App.  sol.  ',F7.3,4F9.3,'  ODE sol. =',F8.3)
99993 FORMAT (1X,'Exact sol.  ',F7.3,4F9.3,'  ODE sol. =',F8.3,/)
      END
*
      SUBROUTINE UVINIT(NPDE,NPTS,X,U,NCODE,V)
*     Routine for PDE initial values (start time is 0.1D-6)
*     .. Scalar Arguments ..
      INTEGER           NCODE, NPDE, NPTS
*     .. Array Arguments ..
      real              U(NPDE,NPTS), V(*), X(NPTS)
*     .. Scalars in Common ..
      real              TS
*     .. Local Scalars ..
      INTEGER           I
*     .. Intrinsic Functions ..
      INTRINSIC         EXP
*     .. Common blocks ..
      COMMON            /TAXIS/TS
*     .. Executable Statements ..
      V(1) = TS
      DO 20 I = 1, NPTS
         U(1,I) = EXP(TS*(1.0e0-X(I))) - 1.0e0
   20 CONTINUE
      RETURN
      END
*
      SUBROUTINE ODEDEF(NPDE,T,NCODE,V,VDOT,NXI,XI,UCP,UCPX,RCP,UCPT,
     +                  UCPTX,F,IRES)
*     .. Scalar Arguments ..
      real              T
      INTEGER           IRES, NCODE, NPDE, NXI
*     .. Array Arguments ..
      real              F(*), RCP(NPDE,*), UCP(NPDE,*), UCPT(NPDE,*),
     +                  UCPTX(NPDE,*), UCPX(NPDE,*), V(*), VDOT(*),
     +                  XI(*)
*     .. Executable Statements ..
      IF (IRES.EQ.1) THEN
         F(1) = VDOT(1) - V(1)*UCP(1,1) - UCPX(1,1) - 1.0e0 - T
      ELSE IF (IRES.EQ.-1) THEN
         F(1) = VDOT(1)
      END IF
      RETURN
      END
*
      SUBROUTINE PDEDEF(NPDE,T,X,NPTL,U,DUDX,NCODE,V,VDOT,P,Q,R,IRES)
*     .. Scalar Arguments ..
      real              T
      INTEGER           IRES, NCODE, NPDE, NPTL
*     .. Array Arguments ..
      real              DUDX(NPDE,NPTL), P(NPDE,NPDE,NPTL),
     +                  Q(NPDE,NPTL), R(NPDE,NPTL), U(NPDE,NPTL), V(*),
     +                  VDOT(*), X(NPTL)
*     .. Local Scalars ..
      INTEGER           I
*     .. Executable Statements ..
      DO 20 I = 1, NPTL
         P(1,1,I) = V(1)*V(1)
         R(1,I) = DUDX(1,I)
         Q(1,I) = -X(I)*DUDX(1,I)*V(1)*VDOT(1)
   20 CONTINUE
```

```
      RETURN
      END
*
      SUBROUTINE BNDARY(NPDE,T,U,UX,NCODE,V,VDOT,IBND,BETA,GAMMA,IRES)
*     .. Scalar Arguments ..
      real              T
      INTEGER           IBND, IRES, NCODE, NPDE
*     .. Array Arguments ..
      real              BETA(NPDE), GAMMA(NPDE), U(NPDE), UX(NPDE),
     +                  V(*), VDOT(*)
*     .. Intrinsic Functions ..
      INTRINSIC         EXP
*     .. Executable Statements ..
      BETA(1) = 1.0e0
      IF (IBND.EQ.0) THEN
         GAMMA(1) = -V(1)*EXP(T)
      ELSE
         GAMMA(1) = -V(1)*VDOT(1)
      END IF
      RETURN
      END
*
      SUBROUTINE EXACT(TIME,NPTS,X,U)
*     Exact solution (for comparison purposes)
*     .. Scalar Arguments ..
      real              TIME
      INTEGER           NPTS
*     .. Array Arguments ..
      real              U(NPTS), X(NPTS)
*     .. Local Scalars ..
      INTEGER           I
*     .. Intrinsic Functions ..
      INTRINSIC         EXP
*     .. Executable Statements ..
      DO 20 I = 1, NPTS
         U(I) = EXP(TIME*(1.0e0-X(I))) - 1.0e0
   20 CONTINUE
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 D03PJF Example Program Results
 Degree of Polynomial =   2   No. of elements =  10



  Simple coupled PDE using BDF
  Accuracy requirement = 0.100E-03 Number of points =   21

  X            0.000    0.200    0.400    0.600    1.000

 T =  0.200
 App.  sol.    0.222    0.174    0.128    0.084    0.001  ODE sol. =   0.200
 Exact sol.    0.221    0.174    0.127    0.083    0.000  ODE sol. =   0.200

 T =  0.400
 App.  sol.    0.492    0.378    0.272    0.174    0.000  ODE sol. =   0.400
 Exact sol.    0.492    0.377    0.271    0.174    0.000  ODE sol. =   0.400

 T =  0.800
 App.  sol.    1.226    0.897    0.617    0.378    0.000  ODE sol. =   0.800
 Exact sol.    1.226    0.896    0.616    0.377    0.000  ODE sol. =   0.800

 T =  1.600
 App.  sol.    3.954    2.597    1.612    0.896   -0.001  ODE sol. =   1.600
```

```
 Exact sol.    3.953    2.597    1.612    0.896    0.000  ODE sol. =   1.600

 T =  3.200
 App.  sol.   23.534   11.932    5.815    2.590   -0.008  ODE sol. =   3.202
 Exact sol.   23.533   11.936    5.821    2.597    0.000  ODE sol. =   3.200

 Number of integration steps in time =     39
 Number of function evaluations =    462
 Number of Jacobian evaluations =    15
 Number of iterations =    121
```