NAG Library Routine Document

D03PDF/D03PDA

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

D03PDF/D03PDA integrates a system of linear or nonlinear parabolic partial differential equations (PDEs) in one space variable. The spatial discretization is performed using a Chebyshev C^0 collocation method, and the method of lines is employed to reduce the PDEs to a system of ordinary differential equations (ODEs). The resulting system is solved using a backward differentiation formula method.

D03PDA is a version of D03PDF that has additional arguments in order to make it safe for use in multithreaded applications (see Section 5).

2 Specification

2.1 Specification for D03PDF

```
SUBROUTINE D03PDF (NPDE, M, TS, TOUT, PDEDEF, BNDARY, U, NBKPTS, XBKPTS,    &
                   NPOLY, NPTS, X, UINIT, ACC, RSAVE, LRSAVE, ISAVE,        &
                   LISAVE, ITASK, ITRACE, IND, IFAIL)

INTEGER            NPDE, M, NBKPTS, NPOLY, NPTS, LRSAVE, ISAVE(LISAVE),     &
                   LISAVE, ITASK, ITRACE, IND, IFAIL
REAL (KIND=nag_wp) TS, TOUT, U(NPDE,NPTS), XBKPTS(NBKPTS), X(NPTS),         &
                   ACC, RSAVE(LRSAVE)
EXTERNAL           PDEDEF, BNDARY, UINIT
```

2.2 Specification for D03PDA

```
SUBROUTINE D03PDA (NPDE, M, TS, TOUT, PDEDEF, BNDARY, U, NBKPTS, XBKPTS,    &
                   NPOLY, NPTS, X, UINIT, ACC, RSAVE, LRSAVE, ISAVE,        &
                   LISAVE, ITASK, ITRACE, IND, IUSER, RUSER, CWSAV,         &
                   LWSAV, IWSAV, RWSAV, IFAIL)

INTEGER            NPDE, M, NBKPTS, NPOLY, NPTS, LRSAVE, ISAVE(LISAVE),     &
                   LISAVE, ITASK, ITRACE, IND, IUSER(*), IWSAV(505),        &
                   IFAIL
REAL (KIND=nag_wp) TS, TOUT, U(NPDE,NPTS), XBKPTS(NBKPTS), X(NPTS),         &
                   ACC, RSAVE(LRSAVE), RUSER(*), RWSAV(1100)
LOGICAL            LWSAV(100)
CHARACTER(80)      CWSAV(10)
EXTERNAL           PDEDEF, BNDARY, UINIT
```

3 Description

D03PDF/D03PDA integrates the system of parabolic equations:

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

where $P_{i,j}$, Q_i and R_i depend on x, t, U, U_x and the vector U is the set of solution values

$$U(x,t) = [U_1(x,t),\ldots,U_{\text{NPDE}}(x,t)]^{\text{T}}, \tag{2}$$

and the vector U_x is its partial derivative with respect to x. Note that $P_{i,j}$, Q_i and R_i must not depend on $\dfrac{\partial U}{\partial t}$.

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

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

The initial values of the functions $U(x,t)$ must be given at $t = t_0$, and must be specified in UINIT.

The functions R_i, for $i = 1, 2, \ldots, \text{NPDE}$, which may be thought of as fluxes, are also used in the definition of the boundary conditions for each equation. The boundary conditions must have the form

$$\beta_i(x,t)R_i(x,t,U,U_x) = \gamma_i(x,t,U,U_x), \quad i = 1,2,\ldots,\text{NPDE}, \tag{3}$$

where $x = a$ or $x = b$.

The boundary conditions must be specified in BNDARY. Thus, the problem is subject to the following restrictions:

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

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

(iii) 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 break-points $x_1, x_2, \ldots, x_{\text{NBKPTS}}$;

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

(v) 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 by either specifying the solution at $x = 0.0$ or by specifying a zero flux there, that is $\beta_i = 1.0$ and $\gamma_i = 0.0$. See also Section 9.

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 D03PDF/D03PDA as an element, and on this element, a polynomial and its space and time derivatives are made to satisfy the system of PDEs at $\text{NPOLY} - 1$ spatial points, which are chosen internally by the code and the break-points. In the case of just one element, the break-points are the boundaries. 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 $\text{NPDE} \times ((\text{NBKPTS} - 1) \times \text{NPOLY} + 1)$ ODEs in the time direction; one ODE at each break-point for each PDE component and $(\text{NPOLY} - 1)$ ODEs for each PDE component between each pair of break-points. The system is then integrated forwards in time using a backward differentiation formula method.

4 References

Berzins M (1990) Developments in the NAG Library software for parabolic equations *Scientific Software Systems* (eds 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

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 Arguments

1: NPDE – INTEGER *Input*

On entry: the number of PDEs in the system to be solved.

Constraint: NPDE ≥ 1.

2: M – INTEGER *Input*

On entry: the coordinate system used:

M $= 0$
Indicates Cartesian coordinates.

M $= 1$
Indicates cylindrical polar coordinates.

M $= 2$
Indicates spherical polar coordinates.

Constraint: M $= 0$, 1 or 2.

3: TS – REAL (KIND=nag_wp) *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 (KIND=nag_wp) *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 values of the functions $P_{i,j}$, Q_i and R_i which define the system of PDEs. The functions may depend on x, t, U and U_x and must be evaluated at a set of points.

The specification of PDEDEF for D03PDF is:

```
SUBROUTINE PDEDEF (NPDE, T, X, NPTL, U, UX, P, Q, R, IRES)
INTEGER             NPDE, NPTL, IRES
REAL (KIND=nag_wp) T, X(NPTL), U(NPDE,NPTL), UX(NPDE,NPTL),          &
                    P(NPDE,NPDE,NPTL), Q(NPDE,NPTL),                 &
                    R(NPDE,NPTL)
```

The specification of PDEDEF for D03PDA is:

```
SUBROUTINE PDEDEF (NPDE, T, X, NPTL, U, UX, P, Q, R, IRES, IUSER,    &
                    RUSER)

INTEGER             NPDE, NPTL, IRES, IUSER(*)
REAL (KIND=nag_wp) T, X(NPTL), U(NPDE,NPTL), UX(NPDE,NPTL),          &
                    P(NPDE,NPDE,NPTL), Q(NPDE,NPTL),                 &
                    R(NPDE,NPTL), RUSER(*)
```

1: NPDE – INTEGER *Input*

On entry: the number of PDEs in the system.

2: T – REAL (KIND=nag_wp) *Input*

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

3: X(NPTL) – REAL (KIND=nag_wp) 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 $\mathrm{X}(1) < \mathrm{X}(2) < \cdots < \mathrm{X}(\mathrm{NPTL})$.

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 (KIND=nag_wp) array *Input*

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

6: UX(NPDE, NPTL) – REAL (KIND=nag_wp) array *Input*

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

7: P(NPDE, NPDE, NPTL) – REAL (KIND=nag_wp) array *Output*

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

8: Q(NPDE, NPTL) – REAL (KIND=nag_wp) array *Output*

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

9: R(NPDE, NPTL) – REAL (KIND=nag_wp) array *Output*

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

10: IRES – INTEGER *Input/Output*

On entry: set to -1 or 1.

On exit: should usually remain unchanged. However, you 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)routine 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. You may wish to set IRES $= 3$ when a physically meaningless input or output value has been generated. If you consecutively set IRES $= 3$, then D03PDF/D03PDA returns to the calling subroutine with the error indicator set to IFAIL $= 4$.

Note: *the following are additional arguments for specific use with D03PDA. Users of D03PDF therefore need not read the remainder of this description*.

11: IUSER($*$) – INTEGER array *User Workspace*

12: RUSER($*$) – REAL (KIND=nag_wp) array *User Workspace*

PDEDEF is called with the arguments IUSER and RUSER as supplied to D03PDF/D03PDA. You should use the arrays IUSER and RUSER to supply information to PDEDEF.

PDEDEF must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which D03PDF/D03PDA is called. Arguments 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 (3).

The specification of BNDARY for D03PDF is:

```
SUBROUTINE BNDARY (NPDE, T, U, UX, IBND, BETA, GAMMA, IRES)
INTEGER            NPDE, IBND, IRES
REAL (KIND=nag_wp) T, U(NPDE), UX(NPDE), BETA(NPDE),                &
                   GAMMA(NPDE)
```

The specification of BNDARY for D03PDA is:

```
SUBROUTINE BNDARY (NPDE, T, U, UX, IBND, BETA, GAMMA, IRES,         &
                   IUSER, RUSER)
INTEGER            NPDE, IBND, IRES, IUSER(*)
REAL (KIND=nag_wp) T, U(NPDE), UX(NPDE), BETA(NPDE),                &
                   GAMMA(NPDE), RUSER(*)
```

1: NPDE – INTEGER *Input*

On entry: the number of PDEs in the system.

2: T – REAL (KIND=nag_wp) *Input*

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

3: U(NPDE) – REAL (KIND=nag_wp) array *Input*

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

4: UX(NPDE) – REAL (KIND=nag_wp) array *Input*

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

5: IBND – INTEGER *Input*

On entry: specifies which boundary conditions are to be evaluated.

IBND $= 0$
: BNDARY must set up the coefficients of the left-hand boundary, $x = a$.

IBND $\neq 0$
: BNDARY must set up the coefficients of the right-hand boundary, $x = b$.

6: BETA(NPDE) – REAL (KIND=nag_wp) array *Output*

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

7: GAMMA(NPDE) – REAL (KIND=nag_wp) array *Output*

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

8: IRES – INTEGER *Input/Output*

On entry: set to -1 or 1.

On exit: should usually remain unchanged. However, you 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)routine 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. You may wish to set IRES $= 3$ when a physically meaningless input or output value has been generated. If you consecutively set IRES $= 3$, then D03PDF/D03PDA returns to the calling subroutine with the error indicator set to IFAIL $= 4$.

Note: *the following are additional arguments for specific use with D03PDA. Users of D03PDF therefore need not read the remainder of this description*.

9: IUSER$(*)$ – INTEGER array *User Workspace*

10: RUSER$(*)$ – REAL (KIND=nag_wp) array *User Workspace*

BNDARY is called with the arguments IUSER and RUSER as supplied to D03PDF/D03PDA. You should use the arrays IUSER and RUSER to supply information to BNDARY.

BNDARY must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D03PDF/D03PDA is called. Arguments denoted as *Input* must **not** be changed by this procedure.

7: U(NPDE, NPTS) – REAL (KIND=nag_wp) array *Input/Output*

On entry: if IND $= 1$ the value of U must be unchanged from the previous call.

On exit: U(i, j) will contain the computed solution at $t =$ TS.

8: NBKPTS – INTEGER *Input*

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

Constraint: NBKPTS ≥ 2.

9: XBKPTS(NBKPTS) – REAL (KIND=nag_wp) 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 $=$ (NBKPTS $- 1) \times$ NPOLY $+ 1$.

12: X(NPTS) – REAL (KIND=nag_wp) array *Output*

On exit: the mesh points chosen by D03PDF/D03PDA in the spatial direction. The values of X will satisfy $\mathrm{X}(1) < \mathrm{X}(2) < \cdots < \mathrm{X}(\mathrm{NPTS})$.

13: UINIT – SUBROUTINE, supplied by the user. *External Procedure*

UINIT must compute the initial values of the PDE components $U_i(x_j, t_0)$, for $i = 1, 2, \ldots, \mathrm{NPDE}$ and $j = 1, 2, \ldots, \mathrm{NPTS}$.

The specification of UINIT for D03PDF is:

```
SUBROUTINE UINIT (NPDE, NPTS, X, U)
INTEGER             NPDE, NPTS
REAL (KIND=nag_wp) X(NPTS), U(NPDE,NPTS)
```

The specification of UINIT for D03PDA is:

```
SUBROUTINE UINIT (NPDE, NPTS, X, U, IUSER, RUSER)
INTEGER             NPDE, NPTS, IUSER(*)
REAL (KIND=nag_wp) X(NPTS), U(NPDE,NPTS), 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 (KIND=nag_wp) array *Input*

On entry: $\mathrm{X}(j)$, contains the values of the jth mesh point, for $j = 1, 2, \ldots, \mathrm{NPTS}$.

4: U(NPDE, NPTS) – REAL (KIND=nag_wp) array *Output*

On exit: $\mathrm{U}(i, j)$ must be set to the initial value $U_i(x_j, t_0)$, for $i = 1, 2, \ldots, \mathrm{NPDE}$ and $j = 1, 2, \ldots, \mathrm{NPTS}$.

Note: *the following are additional arguments for specific use with D03PDA. Users of D03PDF therefore need not read the remainder of this description*.

5: IUSER(∗) – INTEGER array *User Workspace*

6: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

UINIT is called with the arguments IUSER and RUSER as supplied to D03PDF/D03PDA. You should use the arrays IUSER and RUSER to supply information to UINIT.

UINIT must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which D03PDF/D03PDA is called. Arguments denoted as *Input* must **not** be changed by this procedure.

14: ACC – REAL (KIND=nag_wp) *Input*

On entry: a positive quantity for controlling the local error estimate in the time integration. If $E(i, j)$ is the estimated error for U_i at the jth mesh point, the error test is:

$$|E(i, j)| = \mathrm{ACC} \times (1.0 + |\mathrm{U}(i, j)|).$$

Constraint: $\mathrm{ACC} > 0.0$.

15: RSAVE(LRSAVE) – REAL (KIND=nag_wp) array *Communication Array*

If $\mathrm{IND} = 0$, RSAVE need not be set on entry.

If IND = 1, RSAVE must be unchanged from the previous call to the routine because it contains required information about the iteration.

16: LRSAVE – INTEGER *Input*

On entry: the dimension of the array RSAVE as declared in the (sub)program from which D03PDF/D03PDA is called.

Constraint: LRSAVE $\geq 11 \times \text{NPDE} \times \text{NPTS} + 50 + nwkres + lenode$.

17: ISAVE(LISAVE) – INTEGER array *Communication Array*

If IND = 0, ISAVE need not be set on entry.

If IND = 1, ISAVE must be unchanged from the previous call to the routine because it contains required information about the iteration. In particular:

ISAVE(1)
Contains the number of steps taken in time.

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

ISAVE(3)
Contains the number of Jacobian evaluations performed by the time integrator.

ISAVE(4)
Contains the order of the last backward differentiation formula method used.

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

18: LISAVE – INTEGER *Input*

On entry: the dimension of the array ISAVE as declared in the (sub)program from which D03PDF/D03PDA is called.

Constraint: LISAVE $\geq \text{NPDE} \times \text{NPTS} + 24$.

19: ITASK – INTEGER *Input*

On entry: specifies the task to be performed by the ODE integrator.

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 $t = \text{TOUT}$.

Constraint: ITASK = 1, 2 or 3.

20: ITRACE – INTEGER *Input*

On entry: the level of trace information required from D03PDF/D03PDA and the underlying ODE solver. ITRACE may take the value −1, 0, 1, 2 or 3.

ITRACE = −1
No output is generated.

ITRACE $= 0$
: Only warning messages from the PDE solver are printed on the current error message unit (see X04AAF).

ITRACE > 0
: 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.

If ITRACE < -1, then -1 is assumed and similarly if ITRACE > 3, then 3 is assumed.

The advisory messages are given in greater detail as ITRACE increases. You are advised to set ITRACE $= 0$, unless you are experienced with Sub-chapter D02M–N.

21: IND – INTEGER *Input/Output*

On entry: indicates whether this is a continuation call or a new integration.

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 arguments TOUT and IFAIL should be reset between calls to D03PDF/D03PDA.

Constraint: IND $= 0$ or 1.

On exit: IND $= 1$.

22: IFAIL – INTEGER *Input/Output*

Note: *for D03PDA, IFAIL does not occur in this position in the argument list. See the additional arguments described below.*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

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, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

Note: *the following are additional arguments for specific use with D03PDA. Users of D03PDF therefore need not read the remainder of this description.*

23: IUSER($*$) – INTEGER array *User Workspace*

IUSER is not used by D03PDF/D03PDA, but is passed directly to PDEDEF, BNDARY and UINIT and should be used to pass information to these routines.

24: RUSER($*$) – REAL (KIND=nag_wp) array *User Workspace*

25: CWSAV(10) – CHARACTER(80) array *Communication Array*

26: LWSAV(100) – LOGICAL array *Communication Array*

27: IWSAV(505) – INTEGER array *Communication Array*

28: RWSAV(1100) – REAL (KIND=nag_wp) array *Communication Array*

29: IFAIL – INTEGER *Input/Output*

Note: see the argument 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 $\leq$ TS,
or TOUT $-$ TS is too small,
or ITASK $\neq 1$, 2 or 3,
or M $\neq 0$, 1 or 2,
or M > 0 and XBKPTS$(1) < 0.0$,
or NPDE < 1,
or NBKPTS < 2,
or NPOLY < 1 or NPOLY > 49,
or NPTS $\neq$ (NBKPTS $- 1$) $\times$ NPOLY $+ 1$,
or ACC ≤ 0.0,
or IND $\neq 0$ or 1,
or break-points XBKPTS(i) are not ordered,
or LRSAVE is too small,
or LISAVE is too small.

IFAIL $= 2$

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

IFAIL $= 3$

In the underlying ODE solver, there were repeated errors or corrector convergence test failures on an attempted step, before completing the requested task. The problem may have a singularity or ACC is too small for the integration to continue. Integration was successful as far as $t =$ TS.

IFAIL $= 4$

In setting up the ODE system, the internal initialization routine was unable to initialize the derivative of the ODE system. This could be due to the fact that IRES was repeatedly set to 3 in at least PDEDEF or BNDARY, when the residual in the underlying ODE solver was being evaluated.

IFAIL $= 5$

In solving the ODE system, a singular Jacobian has been encountered. You should check your problem formulation.

IFAIL $= 6$

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

IFAIL $= 7$

The value of ACC is so small that the routine is unable to start the integration in time.

IFAIL $= 8$

In one of PDEDEF or BNDARY, IRES was set to an invalid value.

IFAIL $= 9$ (D02NNF)

A serious error has occurred in an internal call to the specified routine. Check the problem specification and all arguments 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 ACC is unlikely to produce any change in the computed solution. (Only applies when you are not operating in one step mode, that is when ITASK $\neq 2$.)

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 $= 12$

Not applicable.

IFAIL $= 13$

Not applicable.

IFAIL $= 14$

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

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

D03PDF/D03PDA controls the accuracy of the integration in the time direction but not the accuracy of the approximation in space. The spatial accuracy depends on the degree of the polynomial approximation NPOLY, and on both the number of break-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. You should therefore test the effect of varying the accuracy argument, ACC.

8 Parallelism and Performance

D03PDF/D03PDA is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

D03PDF/D03PDA makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

D03PDF/D03PDA is designed to solve parabolic systems (possibly including elliptic equations) with second-order derivatives in space. The argument specification allows you 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 depends on the complexity of the parabolic system and on the accuracy requested.

10 Example

The problem consists of a fourth-order PDE which can be written as a pair of second-order elliptic-parabolic PDEs for $U_1(x,t)$ and $U_2(x,t)$,

$$0 = \frac{\partial^2 U_1}{\partial x^2} - U_2 \tag{4}$$

$$\frac{\partial U_2}{\partial t} = \frac{\partial^2 U_2}{\partial x^2} + U_2\frac{\partial U_1}{\partial x} - U_1\frac{\partial U_2}{\partial x} \tag{5}$$

where $-1 \leq x \leq 1$ and $t \geq 0$. The boundary conditions are given by

$$\frac{\partial U_1}{\partial x} = 0 \quad \text{and} \quad U_1 = 1 \quad \text{at } x = -1, \quad \text{and}$$

$$\frac{\partial U_1}{\partial x} = 0 \quad \text{and} \quad U_1 = -1 \quad \text{at } x = 1.$$

The initial conditions at $t = 0$ are given by

$$U_1 = -\sin\frac{\pi x}{2} \quad \text{and} \quad U_2 = \frac{\pi^2}{4}\sin\frac{\pi x}{2}.$$

The absence of boundary conditions for $U_2(x,t)$ does not pose any difficulties provided that the derivative flux boundary conditions are assigned to the first PDE (4) which has the correct flux, $\frac{\partial U_1}{\partial x}$. The conditions on $U_1(x,t)$ at the boundaries are assigned to the second PDE by setting $\beta_2 = 0.0$ in equation (3) and placing the Dirichlet boundary conditions on $U_1(x,t)$ in the function γ_2.

10.1 Program Text

the following program illustrates the use of D03PDF. An equivalent program illustrating the use of D03PDA is available with the supplied Library and is also available from the NAG web site.

```
!   D03PDF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module d03pdfe_mod

!     D03PDF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
```

```
!     .. Accessibility Statements ..
      Private
      Public                           :: bndary, pdedef, uinit
!     .. Parameters ..
      Integer, Parameter, Public       :: nin = 5, nout = 6, npde = 2
    Contains
      Subroutine uinit(npde,npts,x,u)

!       .. Use Statements ..
        Use nag_library, Only: x01aaf
!       .. Scalar Arguments ..
        Integer, Intent (In)           :: npde, npts
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: u(npde,npts)
        Real (Kind=nag_wp), Intent (In) :: x(npts)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: piby2
        Integer                        :: i
!       .. Intrinsic Procedures ..
        Intrinsic                      :: sin
!       .. Executable Statements ..
        piby2 = 0.5_nag_wp*x01aaf(piby2)
        Do i = 1, npts
          u(1,i) = -sin(piby2*x(i))
          u(2,i) = -piby2*piby2*u(1,i)
        End Do
        Return
      End Subroutine uinit

      Subroutine pdedef(npde,t,x,nptl,u,ux,p,q,r,ires)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: t
        Integer, Intent (Inout)        :: ires
        Integer, Intent (In)           :: npde, nptl
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: p(npde,npde,nptl), q(npde,nptl),   &
                                            r(npde,nptl)
        Real (Kind=nag_wp), Intent (In) :: u(npde,nptl), ux(npde,nptl),        &
                                            x(nptl)
!       .. Local Scalars ..
        Integer                        :: i
!       .. Executable Statements ..
        Do i = 1, nptl
          q(1,i) = u(2,i)
          q(2,i) = u(1,i)*ux(2,i) - ux(1,i)*u(2,i)
          r(1,i) = ux(1,i)
          r(2,i) = ux(2,i)
          p(1,1,i) = 0.0_nag_wp
          p(1,2,i) = 0.0_nag_wp
          p(2,1,i) = 0.0_nag_wp
          p(2,2,i) = 1.0_nag_wp
        End Do
        Return
      End Subroutine pdedef

      Subroutine bndary(npde,t,u,ux,ibnd,beta,gamma,ires)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: t
        Integer, Intent (In)           :: ibnd, npde
        Integer, Intent (Inout)        :: ires
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: beta(npde), gamma(npde)
        Real (Kind=nag_wp), Intent (In) :: u(npde), ux(npde)
!       .. Executable Statements ..
        If (ibnd==0) Then
          beta(1) = 1.0_nag_wp
          gamma(1) = 0.0_nag_wp
          beta(2) = 0.0_nag_wp
          gamma(2) = u(1) - 1.0_nag_wp
```

```
        Else
          beta(1) = 1.0E+0_nag_wp
          gamma(1) = 0.0_nag_wp
          beta(2) = 0.0_nag_wp
          gamma(2) = u(1) + 1.0_nag_wp
        End If
        Return
      End Subroutine bndary
    End Module d03pdfe_mod

    Program d03pdfe

!     D03PDF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d03pdf, d03pyf, nag_wp
      Use d03pdfe_mod, Only: bndary, nin, nout, npde, pdedef, uinit
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: acc, dx, tout, ts
      Integer                          :: i, ifail, ind, intpts, it, itask,    &
                                          itrace, itype, lenode, lisave,       &
                                          lrsave, m, mu, nbkpts, nel, neqn,    &
                                          npl1, npoly, npts, nwkres
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: rsave(:), u(:,:), uout(:,:,:), x(:), &
                                          xbkpts(:), xout(:)
      Integer, Allocatable             :: isave(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: real
!     .. Executable Statements ..
      Write (nout,*) 'D03PDF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) intpts, nbkpts, npoly, itype

      nel = nbkpts - 1
      npts = nel*npoly + 1
      mu = npde*(npoly+1) - 1
      neqn = npde*npts
      lisave = neqn + 24
      npl1 = npoly + 1
      nwkres = 3*npl1*npl1 + npl1*(npde*npde+6*npde+nbkpts+1) + 13*npde + 5
      lenode = (3*mu+1)*neqn
      lrsave = 11*neqn + 50 + nwkres + lenode

      Allocate (u(npde,npts),uout(npde,intpts,itype),rsave(lrsave),x(npts),    &
        xbkpts(nbkpts),xout(intpts),isave(lisave))

      Read (nin,*) xout(1:intpts)
      Read (nin,*) acc
      Read (nin,*) m, itrace

!     Set the break-points

      dx = 2.0_nag_wp/real(nbkpts-1,kind=nag_wp)
      xbkpts(1) = -1.0_nag_wp
      Do i = 2, nbkpts - 1
        xbkpts(i) = xbkpts(i-1) + dx
      End Do
      xbkpts(nbkpts) = 1.0_nag_wp

      ind = 0
      itask = 1
      Read (nin,*) ts, tout

!     Loop over output values of t

      Do it = 1, 5
        tout = 10.0_nag_wp*tout
```

```
!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call d03pdf(npde,m,ts,tout,pdedef,bndary,u,nbkpts,xbkpts,npoly,npts,x, &
          uinit,acc,rsave,lrsave,isave,lisave,itask,itrace,ind,ifail)

        If (it==1) Then
          Write (nout,99999) npoly, nel
          Write (nout,99998) acc, npts
          Write (nout,99997) xout(1:6)
        End If

!       Interpolate at required spatial points

        ifail = 0
        Call d03pyf(npde,u,nbkpts,xbkpts,npoly,npts,xout,intpts,itype,uout,    &
          rsave,lrsave,ifail)

        Write (nout,99996) ts, uout(1,1:intpts,1)
        Write (nout,99995) uout(2,1:intpts,1)
      End Do

!     Print integration statistics

      Write (nout,99994) isave(1), isave(2), isave(3), isave(5)

99999 Format (' Polynomial degree =',I4,'   No. of elements = ',I4)
99998 Format (' Accuracy requirement =',E10.3,'  Number of points = ',I5,/)
99997 Format ('  T /   X    ',6F8.4,/)
99996 Format (1X,F7.4,' U(1)',6F8.4)
99995 Format (9X,'U(2)',6F8.4,/)
99994 Format (' Number of integration steps in time                  ',I4,/,  &
        ' Number of residual evaluations of resulting ODE system',I4,/,        &
        ' Number of Jacobian evaluations                         ',I4,/,        &
        ' Number of iterations of nonlinear solver               ',I4)
    End Program d03pdfe
```

10.2 Program Data

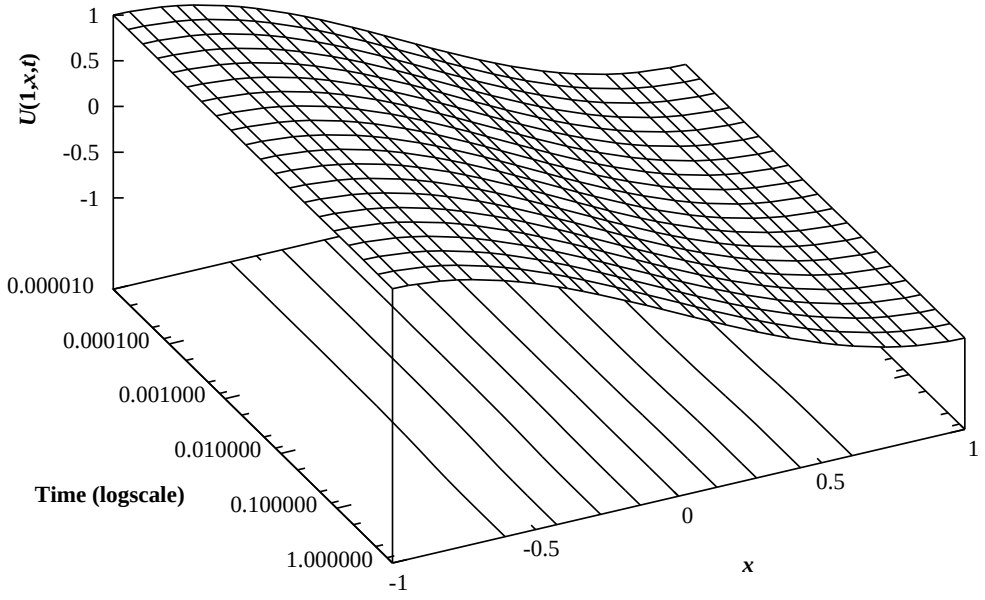
```
D03PDF Example Program Data
  6 10 3 1                    : intpts, nbkpts, npoly, itype
  -1.0 -0.6 -0.2 0.2 0.6 1.0  : xout
  1.0E-4                      : acc
  0 0                         : m, itrace
  0.0 0.1E-4                  : ts, tout
```

10.3 Program Results

```
 D03PDF Example Program Results
 Polynomial degree =   3   No. of elements =    9
 Accuracy requirement = 0.100E-03  Number of points =    28

  T /   X     -1.0000 -0.6000 -0.2000  0.2000  0.6000  1.0000

  0.0001 U(1)  1.0000  0.8090  0.3090 -0.3090 -0.8090 -1.0000
         U(2) -2.4850 -1.9957 -0.7623  0.7623  1.9957  2.4850

  0.0010 U(1)  1.0000  0.8085  0.3088 -0.3088 -0.8085 -1.0000
         U(2) -2.5583 -1.9913 -0.7606  0.7606  1.9913  2.5583

  0.0100 U(1)  1.0000  0.8051  0.3068 -0.3068 -0.8051 -1.0000
         U(2) -2.6962 -1.9481 -0.7439  0.7439  1.9481  2.6962

  0.1000 U(1)  1.0000  0.7951  0.2985 -0.2985 -0.7951 -1.0000
         U(2) -2.9022 -1.8339 -0.6338  0.6338  1.8339  2.9022

  1.0000 U(1)  1.0000  0.7939  0.2972 -0.2972 -0.7939 -1.0000
         U(2) -2.9233 -1.8247 -0.6120  0.6120  1.8247  2.9233
```

```
Number of integration steps in time                      50
Number of residual evaluations of resulting ODE system 407
Number of Jacobian evaluations                           18
Number of iterations of nonlinear solver                122
```

Example Program
Solution, $U(1,x,t)$, of Elliptic-parabolic Pair using Chebyshev Collocation and BDF



Solution, $U(2,x,t)$, of Elliptic-parabolic Pair using Chebyshev Collocation and BDF