

D03PDF – 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

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

2 Specification

```

SUBROUTINE D03PDF(NPDE, M, TS, TOUT, PDEDEF, BNDARY, U, NBKPTS,
1          XBKPTS, NPOLY, NPTS, X, UINIT, ACC, W, NW, IW,
2          NIW, ITASK, ITRACE, IND, IFAIL)
  INTEGER      NPDE, M, NBKPTS, NPOLY, NPTS, NW, IW(NIW), NIW,
1          ITASK, ITRACE, IND, IFAIL
  real        TS, TOUT, U(NPDE,NPTS), XBKPTS(NBKPTS), X(NPTS),
1          ACC, W(NW)
  EXTERNAL    PDEDEF, BNDARY, UINIT

```

3 Description

D03PDF integrates the system of parabolic equations:

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

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

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

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 $\frac{\partial U}{\partial t}$.

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 $x_1, x_2, \dots, x_{\text{NBKPTS}}$. 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 system is defined by the functions $P_{i,j}$, Q_i and R_i which must be specified in a subroutine PDEDEF supplied by the user.

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

The functions R_i , for $i = 1, 2, \dots, \text{NPDE}$, 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

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

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

The boundary conditions must be specified in a subroutine BNDARY provided by the user. Thus, the problem is subject to the following restrictions:

- (i) $t_0 < t_{out}$, 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 $x_1, x_2, \dots, x_{\text{NBKPTS}}$;
- (iv) At least one of the functions $P_{i,j}$ must be non-zero 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 8 below.

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 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. 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

- [1] Berzins M (1990) Developments in the NAG Library software for parabolic equations *Scientific Software Systems* (ed J C Mason and M G Cox) Chapman and Hall 59–72
- [2] 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
- [3] Zaturka 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 in the system to be solved.
Constraint: NPDE \geq 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 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 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.
 Its specification is:

	SUBROUTINE PDEDEF(NPDE, T, X, NPTL, U, UX, P, Q, R, IRES) INTEGER NPDE, NPTL, IRES <i>real</i> T, X(NPTL), U(NPDE,NPTL), UX(NPDE,NPTL), 1 P(NPDE,NPDE,NPTL), Q(NPDE,NPTL), R(NPDE,NPTL)	
1:	NPDE — INTEGER <i>On entry:</i> the number of PDEs in the system.	<i>Input</i>
2:	T — <i>real</i> <i>On entry:</i> the current value of the independent variable t .	<i>Input</i>
3:	X(NPTL) — <i>real</i> array <i>On entry:</i> 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 $X(1) < X(2) < \dots < X(NPTL)$.	<i>Input</i>
4:	NPTL — INTEGER <i>On entry:</i> the number of points at which evaluations are required (the value of NPOLY + 1).	<i>Input</i>
5:	U(NPDE,NPTL) — <i>real</i> array <i>On entry:</i> U(i,j) contains the value of the component $U_i(x,t)$ where $x = X(j)$, for $i = 1,2,\dots, NPDE$; $j = 1,2,\dots, NPTL$.	<i>Input</i>
6:	UX(NPDE,NPTL) — <i>real</i> array <i>On entry:</i> UX(i,j) contains the value of the component $\frac{\partial U_i(x,t)}{\partial x}$ where $x = X(j)$, for $i = 1,2,\dots, NPDE$; $j = 1,2,\dots, NPTL$.	<i>Input</i>
7:	P(NPDE,NPDE,NPTL) — <i>real</i> array <i>On exit:</i> P(i,j,k) must be set to the value of $P_{i,j}(x,t,U,U_x)$ where $x = X(k)$, for $i,j = 1,2,\dots, NPDE$; $k = 1,2,\dots, NPTL$.	<i>Output</i>
8:	Q(NPDE,NPTL) — <i>real</i> array <i>On exit:</i> Q(i,j) must be set to the value of $Q_i(x,t,U,U_x)$ where $x = X(j)$, for $i = 1,2,\dots, NPDE$; $j = 1,2,\dots, NPTL$.	<i>Output</i>
9:	R(NPDE,NPTL) — <i>real</i> array <i>On exit:</i> R(i,j) must be set to the value of $R_i(x,t,U,U_x)$ where $x = X(j)$, for $i = 1,2,\dots, NPDE$; $j = 1,2,\dots, NPTL$.	<i>Output</i>
10:	IRES — INTEGER <i>On entry:</i> set to -1 or 1. <i>On exit:</i> 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.	<i>Input/Output</i>

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 D03PDF returns to the calling (sub)program with the error indicator set to IFAIL = 4.

PDEDEF must be declared as EXTERNAL in the (sub)program from which D03PDF 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 (3).

Its specification is:

```

SUBROUTINE BNDARY(NPDE, T, U, UX, IBND, BETA, GAMMA, IRES)
INTEGER          NPDE, IBND, IRES
real           T, U(NPDE), UX(NPDE), BETA(NPDE), GAMMA(NPDE)

```

- | | | |
|----|--|---------------------|
| 1: | NPDE — INTEGER
<i>On entry:</i> the number of PDEs in the system. | <i>Input</i> |
| 2: | T — real
<i>On entry:</i> the current value of the independent variable t . | <i>Input</i> |
| 3: | U(NPDE) — real array
<i>On entry:</i> U(i) contains the value of the component $U_i(x, t)$ at the boundary specified by IBND, for $i = 1, 2, \dots, \text{NPDE}$. | <i>Input</i> |
| 4: | UX(NPDE) — real array
<i>On entry:</i> UX(i) contains the value of the component $\frac{\partial U_i(x, t)}{\partial x}$ at the boundary specified by IBND, for $i = 1, 2, \dots, \text{NPDE}$. | <i>Input</i> |
| 5: | IBND — INTEGER
<i>On entry:</i> specifies which boundary conditions are to be evaluated. If IBND = 0, then BNDARY must set up the coefficients of the left-hand boundary $x = a$. Any other value of IBND indicates that BNDARY must set up the coefficients of the right-hand boundary, $x = b$. | <i>Input</i> |
| 6: | BETA(NPDE) — real array
<i>On exit:</i> BETA(i) must be set to the value of $\beta_i(x, t)$ at the boundary specified by IBND, for $i = 1, 2, \dots, \text{NPDE}$. | <i>Output</i> |
| 7: | GAMMA(NPDE) — real array
<i>On exit:</i> GAMMA(i) must be set to the value of $\gamma_i(x, t, U, U_x)$ at the boundary specified by IBND, for $i = 1, 2, \dots, \text{NPDE}$. | <i>Output</i> |
| 8: | IRES — INTEGER
<i>On entry:</i> set to -1 or 1.

<i>On exit:</i> should usually remain unchanged. However, the user may reset IRES to force the integration routine to take certain actions as described below: | <i>Input/Output</i> |
- 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 D03PDF returns to the calling (sub)program with the error indicator set to IFAIL = 4.

BNDARY must be declared as EXTERNAL in the (sub)program from which D03PDF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

- 7:** U(NPDE,NPTS) — *real* array *Output*
On exit: $U(i, j)$ will contain the computed solution at $t = \text{TS}$.
- 8:** NBKPTS — INTEGER *Input*
On entry: the number of break-points in the interval $[a, b]$.
Constraint: $\text{NBKPTS} \geq 2$.
- 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: $\text{XBKPTS}(1) < \text{XBKPTS}(2) < \dots < \text{XBKPTS}(\text{NBKPTS})$.
- 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: $1 \leq \text{NPOLY} \leq 49$.
- 11:** NPTS — INTEGER *Input*
On entry: the number of mesh points in the interval $[a, b]$.
Constraint: $\text{NPTS} = (\text{NBKPTS} - 1) \times \text{NPOLY} + 1$.
- 12:** X(NPTS) — *real* array *Output*
On exit: the mesh points chosen by D03PDF in the spatial direction. The values of X will satisfy $\text{X}(1) < \text{X}(2) < \dots < \text{X}(\text{NPTS})$.
- 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 $i = 1, 2, \dots, \text{NPDE}$; $j = 1, 2, \dots, \text{NPTS}$.
 Its specification is:

```
SUBROUTINE UINIT(NPDE, NPTS, X, U)
  INTEGER          NPDE, NPTS
  real             X(NPTS), U(NPDE,NPTS)
```

- 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: $\text{X}(j)$, contains the values of the j th mesh point, for $j = 1, 2, \dots, \text{NPTS}$.

4: U(NPDE,NPTS) — *real* array *Output*
On exit: U(*i*, *j*) must be set to the initial value $U_i(x_j, t_0)$, for $i = 1, 2, \dots, \text{NPDE}$; $j = 1, 2, \dots, \text{NPTS}$.

UNIT must be declared as EXTERNAL in the (sub)program from which D03PDF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

14: ACC — *real* *Input*

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

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

Constraint: ACC > 0.0.

15: W(NW) — *real* array *Workspace*

16: NW — INTEGER *Input*

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

Constraints:

$$\begin{aligned} \text{NW} &\geq 11 \times \text{NPDE} \times \text{NPTS} + 50 + \text{NWKRES} + \text{LENODE}, \text{ where} \\ \text{NWKRES} &= 3 \times (\text{NPOLY} + 1)^2 + (\text{NPOLY} + 1) \times [\text{NPDE}^2 + \\ &\quad 6 \times \text{NPDE} + \text{NBKPTS} + 1] + 13 \times \text{NPDE} + 5, \text{ and} \\ \text{LENODE} &= \text{NPDE} \times \text{NPTS} \times [3 \times \text{NPDE} \times (\text{NPOLY} + 1) - 2]. \end{aligned}$$

17: 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 last backward differentiation formula method used.

IW(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: NIW — INTEGER *Input*

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

Constraint: NIW = NPDE × NPTS + 24.

19: 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 only and return.

ITASK = 3

stop at first internal integration point at or beyond $t = \text{TOUT}$.

Constraint: $1 \leq \text{ITASK} \leq 3$.

20: ITRACE — INTEGER*Input*

On entry: the level of trace information required from D03PDF and the underlying ODE solver. ITRACE may take the value -1 , 0 , 1 , 2 , or 3 . If $\text{ITRACE} < -1$, then -1 is assumed and similarly if $\text{ITRACE} > 3$, then 3 is assumed. If $\text{ITRACE} = -1$, no output is generated. If $\text{ITRACE} = 0$, only warning messages from the PDE solver are printed on the current error message unit (see X04AAF). If $\text{ITRACE} > 0$, 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 $\text{ITRACE} = 0$, unless they are experienced with the subchapter D02M–N of the NAG Fortran Library.

21: 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 D03PDF.

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

On exit: $\text{IND} = 1$.

22: IFAIL — INTEGER*Input/Output*

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

On exit: $\text{IFAIL} = 0$ unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

- On entry, $\text{TOUT} \leq \text{TS}$,
- or $(\text{TOUT} - \text{TS})$ is too small,
- or $\text{ITASK} \neq 1, 2$ or 3 ,
- or $\text{M} \neq 0, 1$ or 2 ,
- or $\text{M} > 0$ and $\text{XBKPTS}(1) < 0.0$,
- or $\text{NPDE} < 1$,
- or $\text{NBKPTS} < 2$,
- or $\text{NPOLY} < 1$ or $\text{NPOLY} > 49$,

- or $NPTS \neq (NBKPTS - 1) \times NPOLY + 1$,
- or $ACC \leq 0.0$,
- or $IND \neq 0$ or 1 ,
- or break-points $XBKPTS(i)$ are not ordered,
- or NW or NIW are too small,
- or $IND = 1$ on initial entry to D03PDF.

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 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 the user-supplied subroutines $PDEDEF$ 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. 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 $PDEDEF$ 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 the user-supplied routines, $PDEDEF$ or $BNDARY$, $IRES$ 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 ACC 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 $ITASK \neq 2$.)

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.

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 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. The user should therefore test the effect of varying the accuracy parameter, ACC.

8 Further Comments

The routine is designed to solve parabolic systems (possibly including elliptic equations) with second-order derivatives in space. 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

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 $U_2(x, t)$,

$$0 = \frac{\partial^2 U_1}{\partial x^2} - U_2 \quad (4)$$

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

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

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

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

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

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

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 .

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.

```

*   D03PDF Example Program Text
*   Mark 15 Release. NAG Copyright 1991.
*   .. Parameters ..
INTEGER          NOUT
PARAMETER       (NOUT=6)
INTEGER          NBKPTS, NEL, NPDE, NPOLY, M, NPTS, NEQN, NIW,
+               NPL1, NWKRES, MU, LENODE, NW, ITYPE, INTPTS
PARAMETER       (NBKPTS=10, NEL=NBKPTS-1, NPDE=2, NPOLY=3, M=0,
+               NPTS=NEL*NPOLY+1, NEQN=NPDE*NPTS, NIW=NEQN+24,
+               NPL1=NPOLY+1, NWKRES=3*NPL1*NPL1+NPL1*
+               (NPDE*NPDE+6*NPDE+NBKPTS+1)+13*NPDE+5,
+               MU=NPDE*(NPOLY+1)-1, LENODE=(3*MU+1)*NEQN,
+               NW=11*NEQN+50+NWKRES+LENODE, ITYPE=1, INTPTS=6)
*   .. Scalars in Common ..
  real          PIBY2
*   .. Local Scalars ..
  real          ACC, PI, TOUT, TS
INTEGER          I, IFAIL, IND, IT, ITASK, ITRACE
*   .. Local Arrays ..
  real          U(NPDE,NPTS), UOUT(NPDE,INTPTS,ITYPE), W(NW),
+               X(NPTS), XBKPTS(NBKPTS), XOUT(6)
INTEGER          IW(NIW)
*   .. External Functions ..
  real          X01AAF
EXTERNAL        X01AAF
*   .. External Subroutines ..
EXTERNAL        BNDARY, D03PDF, D03PYF, PDEDEF, UINIT
*   .. Common blocks ..
COMMON         /PIVAL/PIBY2
*   .. Data statements ..
DATA          XOUT(1)/-1.0e+0/, XOUT(2)/-0.6e+0/,
+            XOUT(3)/-0.2e+0/, XOUT(4)/0.2e+0/,
+            XOUT(5)/0.6e+0/, XOUT(6)/1.0e+0/
*   .. Executable Statements ..
WRITE (NOUT,*) 'D03PDF Example Program Results'
PIBY2 = 0.5e0*X01AAF(PI)
ACC = 1.0e-4
ITRACE = 0

*
*   Set the break-points
*
DO 20 I = 1, NBKPTS
  XBKPTS(I) = -1.0e0 + (I-1.0e0)*2.0e0/(NBKPTS-1.0e0)
20 CONTINUE
*
  IND = 0
  ITASK = 1
  TS = 0.0e0
  TOUT = 0.1e-4
  WRITE (NOUT,99999) NPOLY, NEL
  WRITE (NOUT,99998) ACC, NPTS
  WRITE (NOUT,99997) (XOUT(I),I=1,6)
*
*   Loop over output values of t

```

```

*
  DO 40 IT = 1, 5
    IFAIL = -1
    TOUT = 10.0e0*TOUT
*
    CALL D03PDF(NPDE,M,TS,TOUT,PDEDEF,BNDARY,U,NBKPTS,XBKPTS,NPOLY,
+             NPTS,X,UNIT,ACC,W,NW,IW,NIW,ITASK,ITRACE,IND,
+             IFAIL)
*
    Interpolate at required spatial points
*
    CALL D03PYF(NPDE,U,NBKPTS,XBKPTS,NPOLY,NPTS,XOUT,INTPTS,ITYPE,
+             UOUT,W,NW,IFAIL)
    WRITE (NOUT,99996) TS, (UOUT(1,I,1),I=1,INTPTS)
    WRITE (NOUT,99995) (UOUT(2,I,1),I=1,INTPTS)
40 CONTINUE
*
    Print integration statistics
*
    WRITE (NOUT,99994) IW(1), IW(2), IW(3), IW(5)
    STOP
*
99999 FORMAT (' Polynomial degree = ',I4,' No. of elements = ',I4)
99998 FORMAT (' Accuracy requirement = ',e9.3,' Number of points = ',
+           I5,/)
99997 FORMAT (' T / X ',6F8.4,/)
99996 FORMAT (1X,F6.4,' U(1)',6F8.4)
99995 FORMAT (8X,'U(2)',6F8.4,/)
99994 FORMAT (' Number of integration steps in time ',
+           I4,/' Number of residual evaluations of resulting ODE sys',
+           'tem',I4,/' Number of Jacobian evaluations ',
+           ',I4,/' Number of iterations of nonlinear solve',
+           'r ',I4,/)
    END
*
SUBROUTINE UINIT(NPDE,NPTS,X,U)
*
  .. Scalar Arguments ..
  INTEGER          NPDE, NPTS
*
  .. Array Arguments ..
  real             U(NPDE,NPTS), X(NPTS)
*
  .. Scalars in Common ..
  real             PIBY2
*
  .. Local Scalars ..
  INTEGER          I
*
  .. Intrinsic Functions ..
  INTRINSIC       SIN
*
  .. Common blocks ..
  COMMON          /PIVAL/PIBY2
*
  .. Executable Statements ..
  DO 20 I = 1, NPTS
    U(1,I) = -SIN(PIBY2*X(I))
    U(2,I) = -PIBY2*PIBY2*U(1,I)
20 CONTINUE
  RETURN
  END
*

```

```

SUBROUTINE PDEDEF(NPDE,T,X,NPTL,U,DUDX,P,Q,R,IRES)
*
.. Scalar Arguments ..
real          T
INTEGER        IRES, NPDE, NPTL
*
.. Array Arguments ..
real          DUDX(NPDE,NPTL), P(NPDE,NPDE,NPTL),
+              Q(NPDE,NPTL), R(NPDE,NPTL), U(NPDE,NPTL),
+              X(NPTL)
*
.. Local Scalars ..
INTEGER        I
*
.. Executable Statements ..
DO 20 I = 1, NPTL
    Q(1,I) = U(2,I)
    Q(2,I) = U(1,I)*DUDX(2,I) - DUDX(1,I)*U(2,I)
    R(1,I) = DUDX(1,I)
    R(2,I) = DUDX(2,I)
    P(1,1,I) = 0.0e0
    P(1,2,I) = 0.0e0
    P(2,1,I) = 0.0e0
    P(2,2,I) = 1.0e0
20 CONTINUE
RETURN
END

*
SUBROUTINE BNDARY(NPDE,T,U,UX,IBND,BETA,GAMMA,IRES)
*
.. Scalar Arguments ..
real          T
INTEGER        IBND, IRES, NPDE
*
.. Array Arguments ..
real          BETA(NPDE), GAMMA(NPDE), U(NPDE), UX(NPDE)
*
.. Executable Statements ..
IF (IBND.EQ.0) THEN
    BETA(1) = 1.0e0
    GAMMA(1) = 0.0e0
    BETA(2) = 0.0e0
    GAMMA(2) = U(1) - 1.0e0
ELSE
    BETA(1) = 1.0e+0
    GAMMA(1) = 0.0e0
    BETA(2) = 0.0e0
    GAMMA(2) = U(1) + 1.0e0
END IF
RETURN
END

```

9.2 Program Data

None.

9.3 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

```
