# D03PHF – 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

D03PHF 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 finite differences, 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 method or a Theta method (switching between Newton's method and functional iteration).

## 2 Specification

```
 SUBROUTINE D03PHF(NPDE, M, TS, TOUT, PDEDEF, BNDARY, U, NPTS, X,
1                  NCODE, ODEDEF, NXI, XI, NEQN, RTOL, ATOL, ITOL,
2                  NORM, LAOPT, ALGOPT, W, NW, IW, NIW, ITASK,
3                  ITRACE, IND, IFAIL)
 INTEGER           NPDE, M, NPTS, NCODE, NXI, NEQN, ITOL, NW,
1                  IW(NIW), NIW, ITASK, ITRACE, IND, IFAIL
 real              TS, TOUT, U(NEQN), X(NPTS), XI(*), RTOL(*),
1                  ATOL(*), ALGOPT(30), W(NW)
 CHARACTER*1       NORM, LAOPT
 EXTERNAL          PDEDEF, BNDARY, ODEDEF
```

## 3 Description

D03PHF integrates the system of parabolic-elliptic equations and coupled ODEs

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

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

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

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

and the vector $U_x$ is the partial derivative with respect to $x$. The vector $V$ is the set of ODE solution values

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

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

In (2), $\xi$ represents a vector of $n_\xi$ 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 $F = [F_1,\ldots,F_{\text{NCODE}}]^T$, $G$ is a vector of length NCODE, $A$ is an NCODE by NCODE matrix, $B$ is an NCODE by ($n_\xi$ × NPDE) matrix and the entries in $G$, $A$ and $B$ may depend on $t$, $\xi$, $U^*$, $U_x^*$ and $V$. In practice the user only needs 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 $a \leq x \leq b$, where $a = x_1$ and $b = x_{\mathrm{NPTS}}$ are the leftmost and rightmost points of a user-defined mesh $x_1, x_2, \ldots, x_{\mathrm{NPTS}}$. The co-ordinate system in space is defined by the values 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 $t = t_0$.

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

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

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

The boundary conditions must be specified in a subroutine BNDARY provided by the user. The function $\gamma_i$ may depend **linearly** on $\dot{V}$.

The problem is subject to the following restrictions:

(i) In (1), $\dot{V}_j(t)$, for $j = 1,2,\ldots,$NCODE, may only appear **linearly** in the functions $Q_i$, for $i = 1,2,\ldots,$NPDE, with a similar restriction for $\gamma$;
(ii) $P_{i,j}$ and the flux $R_i$ must not depend on any time derivatives;
(iii) $t_0 < t_{out}$, so that integration is in the forward direction;
(iv) The evaluation of the terms $P_{i,j}$, $Q_i$ and $R_i$ is done approximately at the mid-points of the mesh X($i$), for $i = 1,2,\ldots,$NPTS, by calling the routine PDEDEF for each mid-point in turn. Any discontinuities in these functions **must** therefore be at one or more of the mesh points $x_1, x_2, \ldots, x_{\mathrm{NPTS}}$;
(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 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 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 $\xi$ in the array XI.

The parabolic equations are approximated by a system of ODEs in time for the values of $U_i$ at mesh points. For simple problems in Cartesian co-ordinates, this system is obtained by replacing the space derivatives by the usual central, three-point finite-difference formula. However, for polar and spherical problems, or problems with nonlinear coefficients, the space derivatives are replaced by a modified three-point formula which maintains second order accuracy. In total there are NPDE × NPTS+NCODE ODEs in time direction. This system is then integrated forwards in time using a backward differentiation formula (BDF) or a Theta 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, Dew P M and Furzeland R M (1989) Developing software for time-dependent problems using the method of lines and differential-algebraic integrators *Appl. Numer. Math.* **5** 375–397

**[3]** Skeel R D and Berzins M (1990) A method for the spatial discretization of parabolic equations in one space variable *SIAM J. Sci. Statist. Comput.* **11 (1)** 1–32

**[4]** 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

# 5 Parameters

**1:** NPDE — INTEGER *Input*

*On entry:* the number of PDEs 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:* $0 \leq \mathrm{M} \leq 2$.

**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 evaluate 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 $\dot{V}$. PDEDEF is called approximately midway between each pair of mesh points in turn by D03PHF.

Its specification is:

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

**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 — ***real*** *Input*
*On entry:* the current value of the space variable $x$.

**4:** U(NPDE) — ***real*** array *Input*
*On entry:* U($i$) contains the value of the component $U_i(x,t)$, for $i = 1,2,\ldots,$NPDE.

**5:** UX(NPDE) — ***real*** array *Input*
*On entry:* UX($i$) contains the value of the component $\frac{\partial U_i(x,t)}{\partial x}$, for $i = 1,2,\ldots,$NPDE.

**6:** NCODE — INTEGER *Input*
*On entry:* the number of coupled ODEs in the system.

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

**8:** 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.

**9:** P(NPDE,NPDE) — ***real*** array *Output*
*On exit:* P($i, j$) must be set to the value of $P_{i,j}(x, t, U, U_x, V)$, for $i, j = 1, 2, \ldots$,NPDE.

**10:** Q(NPDE) — ***real*** array *Output*
*On exit:* Q($i$) must be set to the value of $Q_i(x, t, U, U_x, V, \dot{V})$, for $i = 1, 2, \ldots$,NPDE.

**11:** R(NPDE) — ***real*** array *Output*
*On exit:* R($i$) must be set to the value of $R_i(x, t, U, U_x, V)$, for $i = 1, 2, \ldots$,NPDE.

**12:** 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 D03PHF 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 D03PHF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

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

BNDARY must evaluate the functions $\beta_i$ and $\gamma_i$ which describe the boundary conditions, as given in (4).

Its specification is:

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

**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 $\frac{\partial U_i(x,t)}{\partial x}$ 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 $\gamma_j$, 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 entry:* 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 D03PHF 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 D03PHF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

**7:** U(NEQN) — ***real*** array *Input/Output*

*On entry:* the initial values of the dependent variables defined as follows: U(NPDE×($j$−1)+$i$) contain $U_i(x_j, t_0)$, for $i = 1, 2, \ldots$,NPDE; $j = 1, 2, \ldots$,NPTS and U(NPTS×NPDE+$i$) contain $V_i(t_0)$, for $i = 1, 2, \ldots$,NCODE.

*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 $t$ = TS.

**8:** NPTS — INTEGER *Input*

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

*Constraint:* NPTS $\geq 3$.

**9:** X(NPTS) — ***real*** array *Input*

*On entry:* the mesh points in the space direction. X(1) must specify the left-hand boundary, $a$, and X(NPTS) must specify the right-hand boundary, $b$.

*Constraint:* X(1) < X(2) < ... < X(NPTS).

**10:** NCODE — INTEGER *Input*

*On entry:* the number of coupled ODE components.

*Constraint:* NCODE ≥ 0.

**11:** 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 (i.e., NCODE = 0), ODEDEF must be the dummy routine D03PCK. (D03PCK is included in the NAG Fortran Library; however, its name may be implementation-dependent: see the the Users' Note for your implementation for details.)

Its specification is:

```
 SUBROUTINE ODEDEF(NPDE, T, NCODE, V, VDOT, NXI, XI, UCP, UCPX,
1                  RCP, 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(*)
```

**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 $i = 1, 2, \ldots$,NCODE.

**5:** VDOT(∗) — ***real*** array *Input*

*On entry:* VDOT($i$) contains the value of component $\dot{V}_i(t)$, for $i = 1, 2, \ldots$,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 point, $\xi_i$, for $i = 1, 2, \ldots$,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 $i = 1, 2, \ldots$,NPDE; $j = 1, 2, \ldots$,NXI.

**9:** UCPX(NPDE,∗) — ***real*** array *Input*

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

**11:** UCPT(NPDE,∗) — ***real*** array *Input*

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

**12:** UCPTX(NPDE,∗) — ***real*** array *Input*

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

**13:** F(∗) — ***real*** array *Output*

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

$$F = G - A\dot{V} - B\begin{pmatrix} U_t^* \\ U_{xt}^* \end{pmatrix}, \tag{5}$$

or

$$F = -A\dot{V} - B\begin{pmatrix} U_t^* \\ U_{xt}^* \end{pmatrix}. \tag{6}$$

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 D03PHF returns to the calling (sub)program with the error indicator set to IFAIL = 4.

ODEDEF must be declared as EXTERNAL in the (sub)program from which D03PHF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

**12:** NXI — INTEGER *Input*

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

*Constraints:*

NXI = 0 if NCODE = 0,
NXI ≥ 0 if NCODE > 0.

**13:** XI(∗) — ***real*** array *Input*

**Note:** the dimension of the array XI must be at least max(1, NXI).

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

*Constraint:* X(1) ≤ XI(1) < XI(2) < … < XI(NXI) ≤ X(NPTS).

**14:** NEQN — INTEGER *Input*

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

*Constraint:* NEQN = NPDE × NPTS+NCODE.

**15:** 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 $i$.

**16:** 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 $i$.

**17:** ITOL — INTEGER *Input*

*On entry:* a value to indicate the form of the local error test. ITOL indicates to D03PHF 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) × \|U($i$)\| + ATOL(1) |
| 2 | scalar | vector | RTOL(1) × \|U($i$)\| + ATOL($i$) |
| 3 | vector | scalar | RTOL($i$) × \|U($i$)\| + ATOL(1) |
| 4 | vector | vector | RTOL($i$) × \|U($i$)\| + ATOL($i$) |

In the above, $e_i$ denotes the estimated local error for the $i$th 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 $\leq 4$.

**18:** NORM — CHARACTER*1 *Input*

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

'M' – maximum 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'.

**19:** LAOPT — CHARACTER*1 *Input*

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

'F' – full matrix routines to be used;
'B' – banded matrix routines to be used;
'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 (NCODE = 0).

**20:** 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) $\leq 0.99$.

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, $t_{\mathrm{crit}}$, beyond which integration must not be attempted. The use of $t_{\mathrm{crit}}$ 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 $t_{\mathrm{crit}}$ 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, i.e., 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.

**21:** W(NW) — ***real*** array *Workspace*

**22:** NW — INTEGER *Input*

*On entry:* the dimension of the array W as declared in the (sub)program from which D03PHF is called. Its size depends on the type of matrix algebra selected:

LAOPT = 'F',
NW $\geq$ NEQN $\times$ NEQN + NEQN + NWKRES + LENODE,
LAOPT = 'B',
NW $\geq$ (3$\times$MLU+1) $\times$ NEQN + NWKRES + LENODE,
LAOPT = 'S',
NW $\geq 4 \times$ NEQN + 11 $\times$ NEQN/2 + 1 + NWKRES + LENODE.

where MLU = the lower or upper half bandwidths, and

MLU = 2 $\times$ NPDE$-$1, for PDE problems only, and

MLU = NEQN$-$1, for coupled PDE/ODE problems.

NWKRES = NPDE $\times$ (NPTS+6$\times$NXI+3$\times$NPDE+15) + NXI + NCODE + 7 $\times$ NPTS + 2 when NCODE $> 0$, and NXI $> 0$.

NWKRES = NPDE $\times$ (NPTS+3$\times$NPDE+21) $\times$ NCODE + 7 $\times$ NPTS+3 when NCODE $> 0$, and NXI = 0.

NWKRES = NPDE $\times$ (NPTS+3$\times$NPDE+21) + 7 $\times$ NPTS + 4 when NCODE = 0.

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

LENODE = 9 $\times$ NEQN+50, when the 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 ITRACE $> 0$ and the routine returns with IFAIL = 15.

**23:** IW(NIW) — INTEGER array *Output*

*On entry:* 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.

**24:** NIW — INTEGER *Input*

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

LAOPT = 'F',
NIW $\geq$ 24,

LAOPT = 'B',
NIW $\geq$ NEQN+24,

LAOPT = 'S',
NIW $\geq 25 \times$ NEQN+24.

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

**25:** ITASK — INTEGER *Input*

*On entry:* 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 $t =$ TOUT (by overshooting and interpolating).

ITASK = 2
take one step in the time direction 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 $t = t_{\mathrm{crit}}$ where $t_{\mathrm{crit}}$ is described under the parameter ALGOPT.

ITASK = 5
take one step in the time direction and return, without passing $t_{\mathrm{crit}}$, where $t_{\mathrm{crit}}$ is described under the parameter ALGOPT.

*Constraint:* $1 \leq$ ITASK $\leq 5$.

**26:** ITRACE — INTEGER *Input*

*On entry:* the level of trace information required from D03PHF 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 the subchapter D02M–N of the NAG Fortran Library.

**27:** 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 D03PHF.

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

*On exit:* IND = 1.

**28:** IFAIL — INTEGER *Input/Output*

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

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

# 6 Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, TOUT−TS is too small,
or ITASK ≠ 1, 2, 3, 4 or 5,
or M ≠ 0, 1 or 2,
or at least one of the coupling points defined in array XI is outside the interval [X(1),X(NPTS)],
or M > 0 and X(1) < 0.0,
or NPTS < 3,
or NPDE < 1,
or NORM ≠ 'A' or 'M',
or LAOPT ≠ 'F', 'B' or 'S',
or ITOL ≠ 1, 2, 3 or 4,
or IND ≠ 0 or 1,
or incorrectly defined user mesh, i.e., X($i$) ≥ X($i$+1) for some $i = 1, 2, \ldots,$NPTS−1,
or NW or NIW are too small,
or NCODE and NXI are incorrectly defined,
or IND = 1 on initial entry to D03PHF,
or NEQN ≠ NPDE × NPTS+NCODE,
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 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 specifications 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 advisory message unit). If using the sparse matrix algebra option, the values of ALGOPT(29) and ALGOPT(30) may be inappropriate.

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 $i$th) 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 parameters 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. It may be advisable in such cases to reduce the whole system to first-order and to use the Keller box scheme routine D03PKF.

The time taken by the routine depends on the complexity of the parabolic system and on the accuracy requested. For a given system and a fixed accuracy it is approximately proportional to NEQN.

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

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

$$\frac{\partial U_1}{\partial x} = -V_1\dot{V}_1$$

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

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

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.

```
*     D03PHF Example Program Text
*     Mark 16 Revised. NAG Copyright 1993.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NPDE, NPTS, NCODE, M, NXI, NEQN, NIW, NWKRES,
     +                 LENODE, NW
      PARAMETER        (NPDE=1,NPTS=21,NCODE=1,M=0,NXI=1,
     +                 NEQN=NPDE*NPTS+NCODE,NIW=24,
     +                 NWKRES=NPDE*(NPTS+6*NXI+3*NPDE+15)
     +                 +NCODE+NXI+7*NPTS+2,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(NPTS), RTOL(1), U(NEQN),
     +                  W(NW), X(NPTS), XI(1)
      INTEGER           IW(NIW)
*     .. External Subroutines ..
      EXTERNAL          BNDARY, D03PHF, EXACT, ODEDEF, PDEDEF, UVINIT
*     .. Common blocks ..
      COMMON            /TAXIS/TS
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D03PHF Example Program Results'
      ITRACE = 0
      ITOL = 1
      ATOL(1) = 1.0e-4
      RTOL(1) = ATOL(1)
      WRITE (NOUT,99997) ATOL, NPTS
*
*     Set break-points
*
      DO 20 I = 1, NPTS
         X(I) = (I-1.0e0)/(NPTS-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,99999) X(1), X(5), X(9), X(13), X(21)
      CALL UVINIT(NPDE,NPTS,X,U,NCODE,NEQN)
      DO 60 IT = 1, 5
         TOUT = 0.1e0*(2**IT)
         IFAIL = -1
*
```

```
         CALL D03PHF(NPDE,M,TS,TOUT,PDEDEF,BNDARY,U,NPTS,X,NCODE,ODEDEF,
     +               NXI,XI,NEQN,RTOL,ATOL,ITOL,NORM,LAOPT,ALGOPT,W,NW,
     +               IW,NIW,ITASK,ITRACE,IND,IFAIL)
*
*        Check against the exact solution
*
         CALL EXACT(TOUT,NPTS,X,EXY)
         WRITE (NOUT,99998) TS
         WRITE (NOUT,99995) U(1), U(5), U(9), U(13), U(21), U(22)
         WRITE (NOUT,99994) EXY(1), EXY(5), EXY(9), EXY(13), EXY(21), TS
   60 CONTINUE
      WRITE (NOUT,99996) IW(1), IW(2), IW(3), IW(5)
      STOP
*
99999 FORMAT (' X        ',5F9.3,/)
99998 FORMAT (' T = ',F6.3)
99997 FORMAT (//'  Simple coupled PDE using BDF ',/'  Accuracy require',
     +       'ment =',e10.3,' Number of points = ',I4,/)
99996 FORMAT (' Number of integration steps in time = ',I6,/' Number o',
     +       'f function evaluations = ',I6,/' Number of Jacobian eval',
     +       'uations =',I6,/' Number of iterations = ',I6,/)
99995 FORMAT (1X,'App.  sol.  ',F7.3,4F9.3,'  ODE sol. =',F8.3)
99994 FORMAT (1X,'Exact sol.  ',F7.3,4F9.3,'  ODE sol. =',F8.3,/)
      END
*
      SUBROUTINE UVINIT(NPDE,NPTS,X,U,NCODE,NEQN)
*     Routine for PDE initial values
*     .. Scalar Arguments ..
      INTEGER           NCODE, NEQN, NPDE, NPTS
*     .. Array Arguments ..
      real              U(NEQN), X(NPTS)
*     .. Scalars in Common ..
      real              TS
*     .. Local Scalars ..
      INTEGER           I
*     .. Intrinsic Functions ..
      INTRINSIC         EXP
*     .. Common blocks ..
      COMMON            /TAXIS/TS
*     .. Executable Statements ..
      DO 20 I = 1, NPTS
         U(I) = EXP(TS*(1.0e0-X(I))) - 1.0e0
   20 CONTINUE
      U(NEQN) = TS
      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,U,UX,NCODE,V,VDOT,P,Q,R,IRES)
*     .. Scalar Arguments ..
      real              T, X
      INTEGER           IRES, NCODE, NPDE
*     .. Array Arguments ..
      real              P(NPDE,NPDE), Q(NPDE), R(NPDE), U(NPDE),
     +                  UX(NPDE), V(*), VDOT(*)
*     .. Executable Statements ..
      P(1,1) = V(1)*V(1)
      R(1) = UX(1)
      Q(1) = -X*UX(1)*V(1)*VDOT(1)
      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 purpose)
*     .. 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

```
D03PHF Example Program Results


 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.    0.494    0.379    0.273    0.176    0.002  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.    1.229    0.901    0.622    0.384    0.008  ODE sol. =   0.798
Exact sol.    1.226    0.896    0.616    0.377    0.000  ODE sol. =   0.800

T =  1.600
App.  sol.    3.959    2.610    1.629    0.917    0.027  ODE sol. =   1.594
Exact sol.    3.953    2.597    1.612    0.896    0.000  ODE sol. =   1.600

T =  3.200
App.  sol.   23.470   11.974    5.886    2.665    0.074  ODE sol. =   3.184
Exact sol.   23.533   11.936    5.821    2.597    0.000  ODE sol. =   3.200

Number of integration steps in time =     32
Number of function evaluations =    443
Number of Jacobian evaluations =    15
Number of iterations =    106
```