NAG Library Routine Document

D03PEF

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

D03PEF integrates a system of linear or nonlinear, first-order, time-dependent partial differential equations (PDEs) in one space variable. The spatial discretization is performed using the Keller box scheme 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 (BDF) method.

2 Specification

```
SUBROUTINE D03PEF (NPDE, TS, TOUT, PDEDEF, BNDARY, U, NPTS, X, NLEFT,      &
                   ACC, RSAVE, LRSAVE, ISAVE, LISAVE, ITASK, ITRACE,       &
                   IND, IFAIL)
INTEGER            NPDE, NPTS, NLEFT, LRSAVE, ISAVE(LISAVE), LISAVE,       &
                   ITASK, ITRACE, IND, IFAIL
REAL (KIND=nag_wp) TS, TOUT, U(NPDE,NPTS), X(NPTS), ACC, RSAVE(LRSAVE)
EXTERNAL           PDEDEF, BNDARY
```

3 Description

D03PEF integrates the system of first-order PDEs

$$G_i(x,t,U,U_x,U_t) = 0, \quad i = 1,2,\ldots,\text{NPDE}. \tag{1}$$

In particular the functions G_i must have the general form

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

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

$$U(x,t) = [U_1(x,t),\ldots,U_{\text{NPDE}}(x,t)]^{\text{T}}, \tag{3}$$

and the vector U_x is its partial derivative with respect to x. Note that $P_{i,j}$ and Q_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 $a \le x \le b$, where $a = x_1$ and $b = x_{\text{NPTS}}$ are the leftmost and rightmost points of a user-defined mesh $x_1, x_2, \ldots, x_{\text{NPTS}}$. The mesh should be chosen in accordance with the expected behaviour of the solution.

The PDE system which is defined by the functions G_i must be specified in PDEDEF.

The initial values of the functions $U(x,t)$ must be given at $t = t_0$. For a first-order system of PDEs, only one boundary condition is required for each PDE component U_i. The NPDE boundary conditions are separated into n_a at the left-hand boundary $x = a$, and n_b at the right-hand boundary $x = b$, such that $n_a + n_b = \text{NPDE}$. The position of the boundary condition for each component should be chosen with care; the general rule is that if the characteristic direction of U_i at the left-hand boundary (say) points into the interior of the solution domain, then the boundary condition for U_i should be specified at the left-hand boundary. Incorrect positioning of boundary conditions generally results in initialization or integration difficulties in the underlying time integration routines.

The boundary conditions have the form:

$$G_i^L(x,t,U,U_t) = 0 \quad \text{at } x = a, \quad i = 1,2,\ldots,n_a \tag{4}$$

at the left-hand boundary, and

$$G_i^R(x,t,U,U_t) = 0 \quad \text{at } x = b, \quad i = 1,2,\ldots,n_b \tag{5}$$

at the right-hand boundary.

Note that the functions G_i^L and G_i^R must not depend on U_x, since spatial derivatives are not determined explicitly in the Keller box scheme (see Keller (1970)). If the problem involves derivative (Neumann) boundary conditions then it is generally possible to restate such boundary conditions in terms of permissible variables. Also note that G_i^L and G_i^R must be linear with respect to time derivatives, so that the boundary conditions have the general form

$$\sum_{j=1}^{\text{NPDE}} E_{i,j}^L \frac{\partial U_j}{\partial t} + S_i^L = 0, \quad i = 1,2,\ldots,n_a \tag{6}$$

at the left-hand boundary, and

$$\sum_{j=1}^{\text{NPDE}} E_{i,j}^R \frac{\partial U_j}{\partial t} + S_i^R = 0, \quad i = 1,2,\ldots,n_b \tag{7}$$

at the right-hand boundary, where $E_{i,j}^L$, $E_{i,j}^R$, S_i^L, and S_i^R depend on x, t and U only.

The boundary conditions must be specified in BNDARY.

The problem is subject to the following restrictions:

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

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

(iii) The evaluation of the function G_i is done at the mid-points of the mesh intervals by calling the PDEDEF for each mid-point in turn. Any discontinuities in the function **must** therefore be at one or more of the mesh points $x_1, x_2, \ldots, x_{\text{NPTS}}$;

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

In this method of lines approach the Keller box scheme (see Keller (1970)) is applied to each PDE in the space variable only, resulting in a system of ODEs in time for the values of U_i at each mesh point. In total there are NPDE $\times$ NPTS ODEs in the time direction. This system is then integrated forwards in time using a BDF 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, 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

Keller H B (1970) A new difference scheme for parabolic problems *Numerical Solutions of Partial Differential Equations* (ed J Bramble) **2** 327–350 Academic Press

Pennington S V and Berzins M (1994) New NAG Library software for first-order partial differential equations *ACM Trans. Math. Softw.* **20** 63–99

5 Arguments

1: NPDE – INTEGER *Input*

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

Constraint: NPDE ≥ 1.

2: TS – REAL (KIND=nag_wp) *Input/Output*

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

Constraint: TS $<$ TOUT.

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

3: TOUT – REAL (KIND=nag_wp) *Input*

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

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

PDEDEF must compute the functions G_i which define the system of PDEs. PDEDEF is called approximately midway between each pair of mesh points in turn by D03PEF.

The specification of PDEDEF is:

```
SUBROUTINE PDEDEF (NPDE, T, X, U, UT, UX, RES, IRES)
INTEGER            NPDE, IRES
REAL (KIND=nag_wp) T, X, U(NPDE), UT(NPDE), UX(NPDE), RES(NPDE)
```

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 – REAL (KIND=nag_wp) *Input*

On entry: the current value of the space variable x.

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

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

5: UT(NPDE) – REAL (KIND=nag_wp) array *Input*

On entry: UT(i) contains the value of the component $\dfrac{\partial U_i(x,t)}{\partial t}$, for $i = 1,2,\ldots,$NPDE.

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

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

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

On exit: RES(i) must contain the ith component of G, for $i = 1,2,\ldots,$NPDE, where G is defined as

$$G_i = \sum_{j=1}^{\text{NPDE}} P_{i,j}\frac{\partial U_j}{\partial t}, \tag{8}$$

i.e., only terms depending explicitly on time derivatives, or

$$G_i = \sum_{j=1}^{\mathrm{NPDE}} P_{i,j}\frac{\partial U_j}{\partial t} + Q_i, \tag{9}$$

i.e., all terms in equation (2).

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

8: IRES – INTEGER *Input/Output*

On entry: the form of G_i that must be returned in the array RES.

IRES $= -1$
Equation (8) must be used.

IRES $= 1$
Equation (9) must be used.

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 D03PEF returns to the calling subroutine with the error indicator set to IFAIL $= 4$.

PDEDEF must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which D03PEF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

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

BNDARY must compute the functions G_i^L and G_i^R which define the boundary conditions as in equations (4) and (5).

The specification of BNDARY is:

```
SUBROUTINE BNDARY (NPDE, T, IBND, NOBC, U, UT, RES, IRES)
INTEGER            NPDE, IBND, NOBC, IRES
REAL (KIND=nag_wp) T, U(NPDE), UT(NPDE), RES(NOBC)
```

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: IBND – INTEGER *Input*

On entry: determines the position of the boundary conditions.

IBND $= 0$
BNDARY must compute the left-hand boundary condition at $x = a$.

IBND $\neq 0$
Indicates that BNDARY must compute the right-hand boundary condition at $x = b$.

4: NOBC – INTEGER *Input*

On entry: specifies the number of boundary conditions at the boundary specified by IBND.

5: 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 $i = 1, 2, \ldots, \text{NPDE}$.

6: UT(NPDE) – REAL (KIND=nag_wp) array *Input*

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

7: RES(NOBC) – REAL (KIND=nag_wp) array *Output*

On exit: RES(i) must contain the ith component of G^L or G^R, depending on the value of IBND, for $i = 1, 2, \ldots, \text{NOBC}$, where G^L is defined as

$$G_i^L = \sum_{j=1}^{\text{NPDE}} E_{i,j}^L \frac{\partial U_j}{\partial t}, \tag{10}$$

i.e., only terms depending explicitly on time derivatives, or

$$G_i^L = \sum_{j=1}^{\text{NPDE}} E_{i,j}^L \frac{\partial U_j}{\partial t} + S_i^L, \tag{11}$$

i.e., all terms in equation (6), and similarly for G_i^R.
The definitions of G^L and G^R are determined by the input value of IRES.

8: IRES – INTEGER *Input/Output*

On entry: the form G_i^L (or G_i^R) that must be returned in the array RES.

IRES $= -1$
Equation (10) must be used.

IRES $= 1$
Equation (11) must be used.

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 D03PEF returns to the calling subroutine with the error indicator set to IFAIL $= 4$.

BNDARY must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D03PEF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

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

On entry: the initial values of $U(x,t)$ at $t = \text{TS}$ and the mesh points $\text{X}(j)$, for $j = 1, 2, \ldots, \text{NPTS}$.

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

7: NPTS – INTEGER *Input*

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

Constraint: $\text{NPTS} \geq 3$.

8: X(NPTS) – REAL (KIND=nag_wp) array *Input*

On entry: the mesh points in the spatial direction. $\text{X}(1)$ must specify the left-hand boundary, a, and X(NPTS) must specify the right-hand boundary, b.

Constraint: $\text{X}(1) < \text{X}(2) < \cdots < \text{X}(\text{NPTS})$.

9: NLEFT – INTEGER *Input*

On entry: the number n_a of boundary conditions at the left-hand mesh point $\text{X}(1)$.

Constraint: $0 \leq \text{NLEFT} \leq \text{NPDE}$.

10: 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:

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

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

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

If 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.

12: LRSAVE – INTEGER *Input*

On entry: the dimension of the array RSAVE as declared in the (sub)program from which D03PEF is called.

Constraint:
$\text{LRSAVE} \geq (4 \times \text{NPDE} + \text{NLEFT} + 14) \times \text{NPDE} \times \text{NPTS} + (3 \times \text{NPDE} + 21) \times \text{NPDE} + 7 \times \text{NPTS} + 54$.

13: 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.

14: LISAVE – INTEGER *Input*

On entry: the dimension of the array ISAVE as declared in the (sub)program from which D03PEF is called.

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

15: 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 = TOUT.

ITASK = 2
Take one step and return.

ITASK = 3
Stop at the first internal integration point at or beyond t = TOUT.

Constraint: ITASK = 1, 2 or 3.

16: ITRACE – INTEGER *Input*

On entry: the level of trace information required from D03PEF and the underlying ODE solver as follows:

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 = 1
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.

ITRACE = 2
Output from the underlying ODE solver is similar to that produced when ITRACE = 1, except that the advisory messages are given in greater detail.

ITRACE $\geq$ 3
Output from the underlying ODE solver is similar to that produced when ITRACE = 2, except that the advisory messages are given in greater detail.

You are advised to set ITRACE = 0, unless you are experienced with Sub-chapter D02M–N.

17: 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 D03PEF.

Constraint: IND $= 0$ or 1.

On exit: IND $= 1$.

18: IFAIL – INTEGER *Input/Output*

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).

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 mesh points X(i) are not ordered correctly,
or NPTS < 3,
or NPDE < 1,
or NLEFT is not in the range 0 to NPDE,
or ACC ≤ 0.0,
or IND $\neq 0$ or 1,
or LRSAVE is too small,
or LISAVE is too small,
or D03PEF called initially with IND $= 1$.

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. Incorrect positioning of boundary conditions may also result in this error. 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 the PDEDEF or BNDARY, when the residual in the underlying ODE solver was being evaluated. Incorrect positioning of boundary conditions may also result in this error.

IFAIL $= 5$

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

IFAIL $= 6$

When evaluating the residual in solving the ODE system, IRES was set to 2 in one of 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 either, 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 advisory message unit).

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

D03PEF 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. You should therefore test the effect of varying the accuracy argument, ACC.

8 Parallelism and Performance

D03PEF is not thread safe and should not be called from a multithreaded user program. Please see Section 3.12.1 in How to Use the NAG Library and its Documentation for more information on thread safety.

D03PEF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

D03PEF 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

The Keller box scheme can be used to solve higher-order problems which have been reduced to first-order by the introduction of new variables (see the example problem in D03PKF). In general, a second-order problem can be solved with slightly greater accuracy using the Keller box scheme instead of a finite difference scheme (D03PCF/D03PCA or D03PHF/D03PHA for example), but at the expense of increased CPU time due to the larger number of function evaluations required.

It should be noted that the Keller box scheme, in common with other central-difference schemes, may be unsuitable for some hyperbolic first-order problems such as the apparently simple linear advection equation $U_t + aU_x = 0$, where a is a constant, resulting in spurious oscillations due to the lack of dissipation. This type of problem requires a discretization scheme with upwind weighting (D03PFF for example), or the addition of a second-order artificial dissipation term.

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

10 Example

This example is the simple first-order system

$$\frac{\partial U_1}{\partial t} + \frac{\partial U_1}{\partial x} + \frac{\partial U_2}{\partial x} = 0,$$

$$\frac{\partial U_2}{\partial t} + 4\frac{\partial U_1}{\partial x} + \frac{\partial U_2}{\partial x} = 0,$$

for $t \in [0,1]$ and $x \in [0,1]$.

The initial conditions are

$$U_1(x,0) = \exp(x), \quad U_2(x,0) = \sin(x),$$

and the Dirichlet boundary conditions for U_1 at $x = 0$ and U_2 at $x = 1$ are given by the exact solution:

$$U_1(x,t) = \tfrac{1}{2}\{\exp(x+t) + \exp(x-3t)\} + \tfrac{1}{4}\{\sin(x-3t) - \sin(x+t)\},$$

$$U_2(x,t) = \exp(x-3t) - \exp(x+t) + \tfrac{1}{2}\{\sin(x+t) + \sin(x-3t)\}.$$

10.1 Program Text

```
!   D03PEF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module d03pefe_mod

!     D03PEF 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, exact, pdedef, uinit
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: half = 0.5_nag_wp
      Real (Kind=nag_wp), Parameter    :: one = 1.0_nag_wp
      Integer, Parameter, Public       :: nin = 5, nleft = 1, nout = 6,        &
                                          npde = 2
    Contains
      Subroutine uinit(npde,npts,x,u)
!       Routine for PDE initial values

!       .. 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 ..
        Integer                        :: i
!       .. Intrinsic Procedures ..
        Intrinsic                      :: exp, sin
!       .. Executable Statements ..
        Do i = 1, npts
          u(1,i) = exp(x(i))
          u(2,i) = sin(x(i))
        End Do
        Return
      End Subroutine uinit
      Subroutine pdedef(npde,t,x,u,ut,ux,res,ires)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: t, x
        Integer, Intent (Inout)        :: ires
        Integer, Intent (In)           :: npde
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: res(npde)
        Real (Kind=nag_wp), Intent (In) :: u(npde), ut(npde), ux(npde)
!       .. Executable Statements ..
        If (ires==-1) Then
          res(1) = ut(1)
          res(2) = ut(2)
        Else
          res(1) = ut(1) + ux(1) + ux(2)
          res(2) = ut(2) + 4.0_nag_wp*ux(1) + ux(2)
        End If
        Return
      End Subroutine pdedef
      Subroutine bndary(npde,t,ibnd,nobc,u,ut,res,ires)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: t
        Integer, Intent (In)           :: ibnd, nobc, npde
        Integer, Intent (Inout)        :: ires
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: res(nobc)
        Real (Kind=nag_wp), Intent (In) :: u(npde), ut(npde)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: t1, t3
!       .. Intrinsic Procedures ..
        Intrinsic                      :: exp, sin
!       .. Executable Statements ..
        If (ires==-1) Then
          res(1) = 0.0_nag_wp
        Else If (ibnd==0) Then
          t3 = -3.0_nag_wp*t
          t1 = t
```

```
          res(1) = u(1) - half*((exp(t3)+exp(t1))+half*(sin(t3)-sin(t1)))
        Else
          t3 = one - 3.0_nag_wp*t
          t1 = one + t
          res(1) = u(2) - ((exp(t3)-exp(t1))+half*(sin(t3)+sin(t1)))
        End If
        Return
      End Subroutine bndary
      Subroutine exact(t,npde,npts,x,u)
!       Exact solution (for comparison purposes)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: t
        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)              :: xt, xt3
        Integer                         :: i
!       .. Intrinsic Procedures ..
        Intrinsic                       :: exp, sin
!       .. Executable Statements ..
        Do i = 1, npts
          xt3 = x(i) - 3.0_nag_wp*t
          xt = x(i) + t
          u(1,i) = half*((exp(xt3)+exp(xt))+half*(sin(xt3)-sin(xt)))
          u(2,i) = (exp(xt3)-exp(xt)) + half*(sin(xt3)+sin(xt))
        End Do
        Return
      End Subroutine exact
    End Module d03pefe_mod
    Program d03pefe

!     D03PEF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d03pef, nag_wp
      Use d03pefe_mod, Only: bndary, exact, nin, nleft, nout, npde, pdedef,    &
                             uinit
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)              :: acc, tout, ts
      Integer                         :: i, ifail, ind, it, itask, itrace,    &
                                         lisave, lrsave, neqn, npts, nwkres
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: eu(:,:), rsave(:), u(:,:), x(:)
      Integer, Allocatable             :: isave(:)
!     .. Intrinsic Procedures ..
      Intrinsic                       :: real
!     .. Executable Statements ..
      Write (nout,*) 'D03PEF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) npts
      nwkres = npde*(npts+21+3*npde) + 7*npts + 4
      neqn = npde*npts
      lisave = neqn + 24
      lrsave = 11*neqn + (4*npde+nleft+2)*neqn + 50 + nwkres

      Allocate (eu(npde,npts),rsave(lrsave),u(npde,npts),x(npts),              &
        isave(lisave))
      Read (nin,*) acc
      Read (nin,*) itrace

!     Set spatial-mesh points

      Do i = 1, npts
        x(i) = real(i-1,kind=nag_wp)/real(npts-1,kind=nag_wp)
      End Do
```

```
      ind = 0
      itask = 1

      Call uinit(npde,npts,x,u)

!     Loop over output value of t
      Read (nin,*) ts, tout

      Do it = 1, 5
        tout = 0.2_nag_wp*real(it,kind=nag_wp)

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call d03pef(npde,ts,tout,pdedef,bndary,u,npts,x,nleft,acc,rsave,       &
          lrsave,isave,lisave,itask,itrace,ind,ifail)

        If (it==1) Then
          Write (nout,99997) acc, npts
          Write (nout,99999) x(5), x(13), x(21), x(29), x(37)
        End If

!       Check against the exact solution

        Call exact(tout,npde,npts,x,eu)

        Write (nout,99998) ts
        Write (nout,99995) u(1,5:37:8)
        Write (nout,99994) eu(1,5:37:8)
        Write (nout,99993) u(2,5:37:8)
        Write (nout,99992) eu(2,5:37:8)
      End Do
      Write (nout,99996) isave(1), isave(2), isave(3), isave(5)

99999 Format (' X        ',5F10.4,/)
99998 Format (' T = ',F5.2)
99997 Format (/,/,'  Accuracy requirement =',E10.3,' Number of points = ',I3,  &
        /)
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 (' Approx U1',5F10.4)
99994 Format (' Exact  U1',5F10.4)
99993 Format (' Approx U2',5F10.4)
99992 Format (' Exact  U2',5F10.4,/)
    End Program d03pefe
```

10.2 Program Data

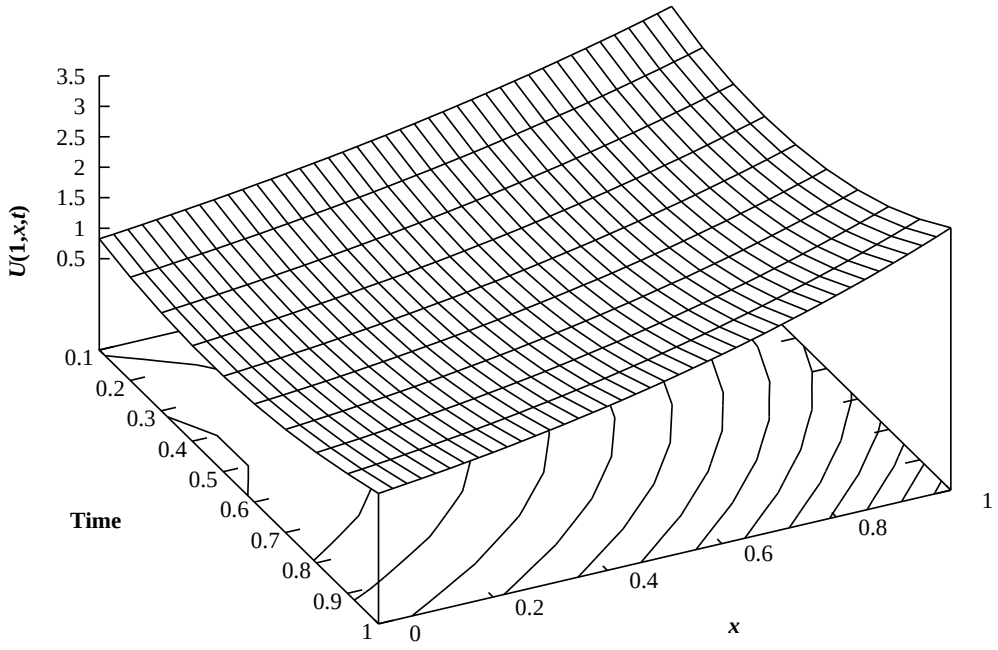
```
D03PEF Example Program Data
  41                : npts
  0.1E-5            : acc
  0                 : itrace
  0.0 0.0           : ts, tout
```

10.3 Program Results

```
 D03PEF Example Program Results


  Accuracy requirement = 0.100E-05 Number of points =  41

 X           0.1000    0.3000    0.5000    0.7000    0.9000

 T =  0.20
 Approx U1    0.7845    1.0010    1.2733    1.6115    2.0281
 Exact  U1    0.7845    1.0010    1.2733    1.6115    2.0281
 Approx U2   -0.8352   -0.8159   -0.8367   -0.9128   -1.0609
 Exact  U2   -0.8353   -0.8160   -0.8367   -0.9129   -1.0609
```

```
T =  0.40
Approx U1    0.6481    0.8533    1.1212    1.4627    1.8903
Exact  U1    0.6481    0.8533    1.1212    1.4627    1.8903
Approx U2   -1.5216   -1.6767   -1.8934   -2.1917   -2.5944
Exact  U2   -1.5217   -1.6767   -1.8935   -2.1917   -2.5945

T =  0.60
Approx U1    0.6892    0.8961    1.1747    1.5374    1.9989
Exact  U1    0.6892    0.8962    1.1747    1.5374    1.9989
Approx U2   -2.0047   -2.3434   -2.7677   -3.3002   -3.9680
Exact  U2   -2.0048   -2.3436   -2.7678   -3.3003   -3.9680

T =  0.80
Approx U1    0.8977    1.1247    1.4320    1.8349    2.3514
Exact  U1    0.8977    1.1247    1.4320    1.8349    2.3512
Approx U2   -2.3403   -2.8675   -3.5110   -4.2960   -5.2536
Exact  U2   -2.3405   -2.8677   -3.5111   -4.2961   -5.2537

T =  1.00
Approx U1    1.2470    1.5206    1.8828    2.3528    2.9519
Exact  U1    1.2470    1.5205    1.8829    2.3528    2.9518
Approx U2   -2.6229   -3.3338   -4.1998   -5.2505   -6.5218
Exact  U2   -2.6232   -3.3340   -4.2001   -5.2507   -6.5219

Number of integration steps in time =    149
Number of function evaluations =    399
Number of Jacobian evaluations =    13
Number of iterations =    323
```

Example Program
Solution, $U(1,x,t)$, of First-order System using Keller, Box and BDF



Solution, $U(2,x,t)$, of First-order System using Keller, Box and BDF