NAG Library Routine Document

D03PCF/D03PCA

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

D03PCF/D03PCA integrates a system of linear or nonlinear parabolic partial differential equations (PDEs) in one space variable. The spatial discretization is performed using finite differences, and the method of lines is employed to reduce the PDEs to a system of ordinary differential equations (ODEs). The resulting system is solved using a backward differentiation formula method.

D03PCA is a version of D03PCF that has additional parameters in order to make it safe for use in multithreaded applications (see Section 5).

2 Specification

2.1 Specification for D03PCF

```
SUBROUTINE D03PCF (NPDE, M, TS, TOUT, PDEDEF, BNDARY, U, NPTS, X, ACC,      &
                   RSAVE, LRSAVE, ISAVE, LISAVE, ITASK, ITRACE, IND,        &
                   IFAIL)
INTEGER            NPDE, M, NPTS, 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
```

2.2 Specification for D03PCA

```
SUBROUTINE D03PCA (NPDE, M, TS, TOUT, PDEDEF, BNDARY, U, NPTS, X, ACC,      &
                   RSAVE, LRSAVE, ISAVE, LISAVE, ITASK, ITRACE, IND,        &
                   IUSER, RUSER, CWSAV, LWSAV, IWSAV, RWSAV, IFAIL)
INTEGER            NPDE, M, NPTS, LRSAVE, ISAVE(LISAVE), LISAVE, ITASK,     &
                   ITRACE, IND, IUSER(*), IWSAV(505), IFAIL
REAL (KIND=nag_wp) TS, TOUT, U(NPDE,NPTS), X(NPTS), ACC,                    &
                   RSAVE(LRSAVE), RUSER(*), RWSAV(1100)
LOGICAL            LWSAV(100)
CHARACTER(80)      CWSAV(10)
EXTERNAL           PDEDEF, BNDARY
```

3 Description

D03PCF/D03PCA integrates the system of parabolic equations:

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

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

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

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

The integration in time is from t_0 to t_{out}, over the space interval $a \le x \le b$, 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 coordinate system in space is defined by the value of m; $m = 0$ for Cartesian coordinates, $m = 1$ for cylindrical polar

coordinates and $m = 2$ for spherical polar coordinates. The mesh should be chosen in accordance with the expected behaviour of the solution.

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

The initial values of the functions $U(x,t)$ must be given at $t = t_0$. The functions R_i, for $i = 1, 2, \ldots,$ NPDE, which may be thought of as fluxes, are also used in the definition of the boundary conditions for each equation. The boundary conditions must have the form

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

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

The boundary conditions must be specified in BNDARY.

The problem is subject to the following restrictions:

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

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

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

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

(v) if $m > 0$ and $x_1 = 0.0$, which is the left boundary point, then it must be ensured that the PDE solution is bounded at this point. This can be done by either specifying the solution at $x = 0.0$ or by specifying a zero flux there, that is $\beta_i = 1.0$ and $\gamma_i = 0.0$. See also Section 9.

The parabolic equations are approximated by a system of ODEs in time for the values of U_i at mesh points. For simple problems in Cartesian coordinates, 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 $\times$ NPTS ODEs in the time direction. This system is then integrated forwards in time using a backward differentiation formula method.

4 References

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

Berzins M, 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

Dew P M and Walsh J (1981) A set of library routines for solving parabolic equations in one space variable *ACM Trans. Math. Software* **7** 295–314

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

5 Parameters

1: NPDE – INTEGER *Input*

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

Constraint: NPDE ≥ 1.

2: M – INTEGER *Input*

On entry: the coordinate system used:

M = 0
Indicates Cartesian coordinates.

M = 1
Indicates cylindrical polar coordinates.

M = 2
Indicates spherical polar coordinates.

Constraint: M = 0, 1 or 2.

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

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

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

Constraint: TS < TOUT.

4: TOUT – REAL (KIND=nag_wp) *Input*

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

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

PDEDEF must compute the functions $P_{i,j}$, Q_i and R_i which define the system of PDEs. PDEDEF is called approximately midway between each pair of mesh points in turn by D03PCF/D03PCA.

The specification of PDEDEF for D03PCF is:

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

The specification of PDEDEF for D03PCA is:

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

1: NPDE – INTEGER *Input*

On entry: the number of PDEs in the system.

2: T – REAL (KIND=nag_wp) *Input*

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

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

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

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

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

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

On exit: Q(i) must be set to the value of $Q_i(x,t,U,U_x)$, for $i = 1,2,\ldots,$NPDE.

8: R(NPDE) – REAL (KIND=nag_wp) array *Output*

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

9: IRES – INTEGER *Input/Output*

On entry: set to -1 or 1.

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

IRES $= 2$
Indicates to the integrator that control should be passed back immediately to the calling (sub)routine with the error indicator set to IFAIL $= 6$.

IRES $= 3$
Indicates to the integrator that the current time step should be abandoned and a smaller time step used instead. You may wish to set IRES $= 3$ when a physically meaningless input or output value has been generated. If you consecutively set IRES $= 3$, then D03PCF/D03PCA returns to the calling subroutine with the error indicator set to IFAIL $= 4$.

Note: *the following are additional parameters for specific use with D03PCA. Users of D03PCF therefore need not read the remainder of this description*.

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

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

PDEDEF is called with the parameters IUSER and RUSER as supplied to D03PCF/D03PCA. You are free to use the arrays IUSER and RUSER to supply information to PDEDEF as an alternative to using COMMON global variables.

PDEDEF must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D03PCF/D03PCA 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).

The specification of BNDARY for D03PCF is:

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

The specification of BNDARY for D03PCA is:

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

1: NPDE – INTEGER *Input*

On entry: the number of PDEs in the system.

2: T – REAL (KIND=nag_wp) *Input*

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

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

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

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

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

5: IBND – INTEGER *Input*

On entry: determines the position of the boundary conditions.

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

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

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

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

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

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

8: IRES – INTEGER *Input/Output*

On entry: set to -1 or 1.

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

IRES $= 2$
: Indicates to the integrator that control should be passed back immediately to the calling (sub)routine with the error indicator set to IFAIL $= 6$.

IRES $= 3$
: Indicates to the integrator that the current time step should be abandoned and a smaller time step used instead. You may wish to set IRES $= 3$ when a physically meaningless input or output value has been generated. If you consecutively set IRES $= 3$, then D03PCF/D03PCA returns to the calling subroutine with the error indicator set to IFAIL $= 4$.

Note: *the following are additional parameters for specific use with D03PCA. Users of D03PCF therefore need not read the remainder of this description.*

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

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

BNDARY is called with the parameters IUSER and RUSER as supplied to D03PCF/D03PCA. You are free to use the arrays IUSER and RUSER to supply information to BNDARY as an alternative to using COMMON global variables.

BNDARY must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D03PCF/D03PCA is called. Parameters denoted as *Input* must **not** be changed by this procedure.

7: 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 $t = \text{TS}$.

8: NPTS – INTEGER *Input*

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

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

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

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 D03PCF/D03PCA is called.

Constraint: $\text{LRSAVE} \geq (6 \times \text{NPDE} + 10) \times \text{NPDE} \times \text{NPTS} + (3 \times \text{NPDE} + 21) \times \text{NPDE} + 7 \times \text{NPTS} + 54$.

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 D03PCF/D03PCA 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
One step and return.

ITASK = 3
Stop at 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 D03PCF/D03PCA and the underlying ODE solver. ITRACE may take the value -1, 0, 1, 2 or 3.

ITRACE = -1
No output is generated.

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

ITRACE $>$ 0
Output from the underlying ODE solver is printed on the current advisory message unit (see X04ABF). This output contains details of Jacobian entries, the nonlinear iteration and the time integration during the computation of the ODE system.

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

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

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 parameters TOUT and IFAIL should be reset between calls to D03PCF/D03PCA.

Constraint: IND = 0 or 1.

On exit: IND = 1.

18: IFAIL – INTEGER *Input/Output*

Note: *for D03PCA, IFAIL does not occur in this position in the parameter list. See the additional parameters described below.*

On entry: IFAIL must be set to 0, −1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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 parameter, the recommended value is 0. **When the value −1 or 1 is used it is essential to test the value of IFAIL on exit.**

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

Note: *the following are additional parameters for specific use with D03PCA. Users of D03PCF therefore need not read the remainder of this description*.

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

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

IUSER and RUSER are not used by D03PCF/D03PCA, but are passed directly to PDEDEF and BNDARY and may be used to pass information to these routines as an alternative to using COMMON global variables.

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

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

If IND = 1, CWSAV must be unchanged from the previous call to the routine.

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

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

If IND = 1, LWSAV must be unchanged from the previous call to the routine.

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

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

If IND = 1, IWSAV must be unchanged from the previous call to the routine.

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

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

If IND = 1, RWSAV must be unchanged from the previous call to the routine.

25: IFAIL – INTEGER *Input/Output*

Note: see the parameter description for IFAIL above.

6 Error Indicators and Warnings

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

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, TOUT $\leq$ TS,
or TOUT $-$ TS is too small,
or ITASK $\neq 1$, 2 or 3,
or M $\neq 0$, 1 or 2,
or M > 0 and X$(1) < 0.0$,
or the mesh points X(i) are not ordered,
or NPTS < 3,
or NPDE < 1,
or ACC ≤ 0.0,
or IND $\neq 0$ or 1,
or LRSAVE is too small,
or LISAVE is too small.

IFAIL $= 2$

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

IFAIL $= 3$

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

IFAIL $= 4$

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

IFAIL $= 5$

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

IFAIL $= 6$

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

IFAIL $= 7$

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

IFAIL $= 8$

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

IFAIL $= 9$ (D02NNF)

A serious error has occurred in an internal call to the specified routine. Check the problem specification and all 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 you are not operating in one step mode, that is when ITASK $\neq 2$.)

IFAIL $= 11$

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

IFAIL $= 12$

Not applicable.

IFAIL $= 13$

Not applicable.

IFAIL $= 14$

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

IFAIL $= -99$

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

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

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

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

D03PCF/D03PCA 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 parameter, ACC.

8 Parallelism and Performance

D03PCF/D03PCA is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

D03PCF/D03PCA 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

D03PCF/D03PCA is designed to solve parabolic systems (possibly including some elliptic equations) with second-order derivatives in space. The parameter specification allows you to include equations with only first-order derivatives in the space direction but there is no guarantee that the method of integration will be satisfactory for such systems. The position and nature of the boundary conditions in particular are critical in defining a stable problem. It may be advisable in such cases to reduce the whole system to first-order and to use the Keller box scheme routine D03PEF.

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

10 Example

We use the example given in Dew and Walsh (1981) which consists of an elliptic-parabolic pair of PDEs. The problem was originally derived from a single third-order in space PDE. The elliptic equation is

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r^2\frac{\partial U_1}{\partial r}\right) = 4\alpha\left(U_2 + r\frac{\partial U_2}{\partial r}\right)$$

and the parabolic equation is

$$\left(1 - r^2\right)\frac{\partial U_2}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(r\left(\frac{\partial U_2}{\partial r} - U_2U_1\right)\right)$$

where $(r,t) \in [0,1] \times [0,1]$. The boundary conditions are given by

$$U_1 = \frac{\partial U_2}{\partial r} = 0 \quad \text{at } r = 0,$$

and

$$\frac{\partial}{\partial r}(rU_1) = 0 \quad \text{and} \quad U_2 = 0 \quad \text{at } r = 1.$$

The first of these boundary conditions implies that the flux term in the second PDE, $\left(\frac{\partial U_2}{\partial r} - U_2U_1\right)$, is zero at $r = 0$.

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

$$U_1 = 2\alpha r \quad \text{and} \quad U_2 = 1.0, \quad r \in [0,1].$$

The value $\alpha = 1$ was used in the problem definition. A mesh of 20 points was used with a circular mesh spacing to cluster the points towards the right-hand side of the spatial interval, $r = 1$.

10.1 Program Text

the following program illustrates the use of D03PCF. An equivalent program illustrating the use of D03PCA is available with the supplied Library and is also available from the NAG web site.

```
!   D03PCF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Module d03pcfe_mod

!     D03PCF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                                :: bndary, pdedef, uinit
!     .. Parameters ..
```

```
      Integer, Parameter, Public           :: nin = 5, nout = 6, npde = 2
!     .. Local Scalars ..
      Real (Kind=nag_wp), Public, Save     :: alpha
    Contains
      Subroutine pdedef(npde,t,x,u,ux,p,q,r,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)     :: p(npde,npde), q(npde), r(npde)
        Real (Kind=nag_wp), Intent (In)      :: u(npde), ux(npde)
!       .. Executable Statements ..
        q(1) = 4.0_nag_wp*alpha*(u(2)+x*ux(2))
        q(2) = 0.0_nag_wp
        r(1) = x*ux(1)
        r(2) = ux(2) - u(1)*u(2)
        p(1,1) = 0.0_nag_wp
        p(1,2) = 0.0_nag_wp
        p(2,1) = 0.0_nag_wp
        p(2,2) = 1.0_nag_wp - x*x
        Return
      End Subroutine pdedef
      Subroutine bndary(npde,t,u,ux,ibnd,beta,gamma,ires)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: t
        Integer, Intent (In)                 :: ibnd, npde
        Integer, Intent (Inout)              :: ires
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: beta(npde), gamma(npde)
        Real (Kind=nag_wp), Intent (In)      :: u(npde), ux(npde)
!       .. Executable Statements ..
        If (ibnd==0) Then
          beta(1) = 0.0_nag_wp
          beta(2) = 1.0_nag_wp
          gamma(1) = u(1)
          gamma(2) = -u(1)*u(2)
        Else
          beta(1) = 1.0_nag_wp
          beta(2) = 0.0_nag_wp
          gamma(1) = -u(1)
          gamma(2) = u(2)
        End If
        Return
      End Subroutine bndary
      Subroutine uinit(u,x,npts)

!       Routine for PDE initial conditon

!       .. Scalar Arguments ..
        Integer, Intent (In)                 :: npts
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: u(2,npts)
        Real (Kind=nag_wp), Intent (In)      :: x(npts)
!       .. Local Scalars ..
        Integer                              :: i
!       .. Executable Statements ..
        Do i = 1, npts
          u(1,i) = 2.0_nag_wp*alpha*x(i)
          u(2,i) = 1.0_nag_wp
        End Do
        Return
      End Subroutine uinit
    End Module d03pcfe_mod
    Program d03pcfe

!     D03PCF Example Main Program

!     .. Use Statements ..
```

```
      Use nag_library, Only: d03pcf, d03pzf, nag_wp, x01aaf
      Use d03pcfe_mod, Only: alpha, bndary, nin, nout, npde, pdedef, uinit
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                  :: acc, hx, pi, piby2, tout, ts
      Integer                             :: i, ifail, ind, intpts, it,       &
                                             itask, itrace, itype, lisave,    &
                                             lrsave, m, neqn, npts, nwk
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable     :: rsave(:), u(:,:), uout(:,:,:),   &
                                             x(:), xout(:)
      Integer, Allocatable                :: isave(:)
!     .. Intrinsic Procedures ..
      Intrinsic                           :: real, sin
!     .. Executable Statements ..
      Write (nout,*) 'D03PCF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) intpts, npts, itype
      neqn = npde*npts
      lisave = neqn + 24
      nwk = (10+6*npde)*neqn
      lrsave = nwk + (21+3*npde)*npde + 7*npts + 54
      Allocate (rsave(lrsave),u(npde,npts),uout(npde,intpts,itype),x(npts), &
        xout(intpts),isave(lisave))

      Read (nin,*) xout(1:intpts)
      Read (nin,*) acc, alpha
      Read (nin,*) m, itrace
      ind = 0
      itask = 1

!     Set spatial mesh points

      piby2 = 0.5_nag_wp*x01aaf(pi)
      hx = piby2/real(npts-1,kind=nag_wp)
      x(1) = 0.0_nag_wp
      x(npts) = 1.0_nag_wp
      Do i = 2, npts - 1
        x(i) = sin(hx*real(i-1,kind=nag_wp))
      End Do

!     Set initial conditions
      Read (nin,*) ts, tout

!     Set the initial values
      Call uinit(u,x,npts)

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

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call d03pcf(npde,m,ts,tout,pdedef,bndary,u,npts,x,acc,rsave,lrsave, &
          isave,lisave,itask,itrace,ind,ifail)

        If (it==1) Then
          Write (nout,99999) acc, alpha
          Write (nout,99998) xout(1:6)
        End If

!       Interpolate at required spatial points

        ifail = 0
        Call d03pzf(npde,m,u,npts,x,xout,intpts,itype,uout,ifail)

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

```
!     Print integration statistics

      Write (nout,99997) isave(1), isave(2), isave(3), isave(5)


99999 Format (//' Accuracy requirement  = ',E12.5/' Parameter ALPHA =', &
        '       ',E12.3/)
99998 Format ('   T  /  X   ',6F8.4/)
99997 Format (' Number of integration steps in time                  ', &
        I4/' Number of residual evaluations of resulting ODE system', &
        I4/' Number of Jacobian evaluations                        ', &
        I4/' Number of iterations of nonlinear solver              ',I4)
99996 Format (1X,F7.4,' U(1)',6F8.4)
99995 Format (9X,'U(2)',6F8.4/)
    End Program d03pcfe
```

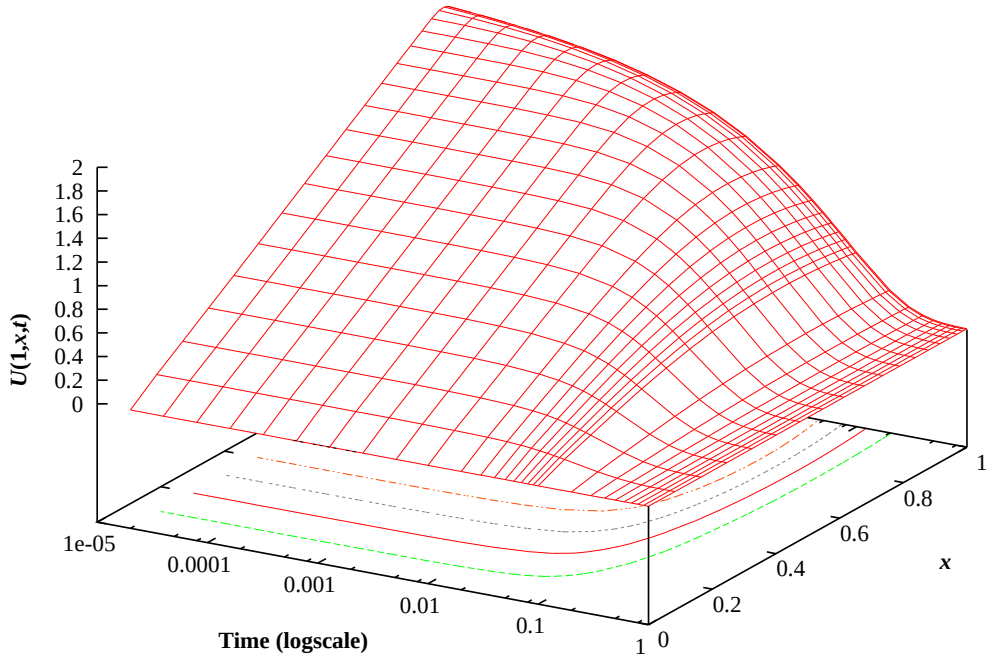
10.2 Program Data

```
D03PCF Example Program Data
  6 20 1                    : intpts, npts, itype
  0.0 0.4 0.6 0.8 0.9 1.0   : xout(1:intpts)
  1.0E-3 1.0                : acc, alpha
  1 0                       : m, itrace
  0.0 0.1E-4                : ts, tout
```

10.3 Program Results

```
 D03PCF Example Program Results


 Accuracy requirement  =  0.10000E-02
 Parameter ALPHA =          0.100E+01

   T  /  X     0.0000  0.4000  0.6000  0.8000  0.9000  1.0000

  0.0001 U(1)  0.0000  0.8008  1.1988  1.5990  1.7958  1.8485
         U(2)  0.9997  0.9995  0.9994  0.9988  0.9663 -0.0000

  0.0010 U(1)  0.0000  0.7982  1.1940  1.5841  1.7179  1.6734
         U(2)  0.9969  0.9952  0.9937  0.9484  0.6385 -0.0000

  0.0100 U(1)  0.0000  0.7676  1.1239  1.3547  1.3635  1.2830
         U(2)  0.9627  0.9495  0.8754  0.5537  0.2908 -0.0000

  0.1000 U(1)  0.0000  0.3908  0.5007  0.5297  0.5120  0.4744
         U(2)  0.5468  0.4299  0.2995  0.1479  0.0724 -0.0000

  1.0000 U(1)  0.0000  0.0007  0.0008  0.0008  0.0008  0.0007
         U(2)  0.0010  0.0007  0.0005  0.0002  0.0001 -0.0000

 Number of integration steps in time                    78
 Number of residual evaluations of resulting ODE system 378
 Number of Jacobian evaluations                          25
 Number of iterations of nonlinear solver               190
```

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



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