NAG Fortran Library Routine Document

D03EBF

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

D03EBF uses the Strongly Implicit Procedure to calculate the solution to a system of simultaneous algebraic equations of five-point molecule form on a two-dimensional topologically-rectangular mesh. ('Topological' means that a polar grid, for example (r, θ), can be used, being equivalent to a rectangular box.)

2 Specification

```
SUBROUTINE D03EBF(N1, N2, N1M, A, B, C, D, E, Q, T, APARAM, ITMAX,
1                 ITCOUN, ITUSED, NDIR, IXN, IYN, CONRES, CONCHN,
2                 RESIDS, CHNGS, WRKSP1, WRKSP2, WRKSP3, IFAIL)
 INTEGER          N1, N2, N1M, ITMAX, ITCOUN, ITUSED, NDIR, IXN, IYN,
1                 IFAIL
 real             A(N1M,N2), B(N1M,N2), C(N1M,N2), D(N1M,N2), E(N1M,N2),
1                 Q(N1M,N2), T(N1M,N2), APARAM, CONRES, CONCHN,
2                 RESIDS(ITMAX), CHNGS(ITMAX), WRKSP1(N1M,N2),
3                 WRKSP2(N1M,N2), WRKSP3(N1M,N2)
```

3 Description

Given a set of simultaneous equations

$$Mt = q \tag{1}$$

(which could be nonlinear) derived, for example, from a finite difference representation of a two-dimensional elliptic partial differential equation and its boundary conditions, the routine determines the values of the dependent variable t. q is a known vector of length $n_1 \times n_2$ and M is a square $(n_1 \times n_2)$ by $(n_1 \times n_2)$ matrix.

The equations must be of five-diagonal form:

$$a_{ij}t_{i,j-1} + b_{ij}t_{i-1,j} + c_{ij}t_{ij} + d_{ij}t_{i+1,j} + e_{ij}t_{i,j+1} = q_{ij}$$

for $i = 1, 2, \ldots, n_1$; $j = 1, 2, \ldots, n_2$, provided $c_{ij} \neq 0.0$. Indeed, if $c_{ij} = 0.0$, then the equation is assumed to be

$$t_{ij} = q_{ij}.$$

For example, if $n_1 = 3$ and $n_2 = 2$, the equations take the form:

$$
\begin{bmatrix}
c_{11} & d_{11} & & e_{11} & & \\
b_{21} & c_{21} & d_{21} & & e_{21} & \\
& b_{31} & c_{31} & & & e_{31} \\
a_{12} & & & c_{12} & d_{12} & \\
& a_{22} & & b_{22} & c_{22} & d_{22} \\
& & a_{32} & & b_{32} & c_{32}
\end{bmatrix}
\begin{bmatrix}
t_{11} \\
t_{21} \\
t_{31} \\
t_{12} \\
t_{22} \\
t_{32}
\end{bmatrix}
=
\begin{bmatrix}
q_{11} \\
q_{21} \\
q_{31} \\
q_{12} \\
q_{22} \\
q_{32}
\end{bmatrix}.
$$

The system is solved iteratively, from a starting approximation $t^{(1)}$, by the formulae

$$r^{(n)} \quad = q - M t^{(n)}$$

$$M s^{(n)} = r^{(n)}$$

$$t^{(n+1)} = t^{(n)} + s^{(n)}.$$

Thus $r^{(n)}$ is the residual of the nth approximate solution $t^{(n)}$, and $s^{(n)}$ is the up-date change vector. The calling program supplies an initial approximation for the values of the dependent variable in the array T, the coefficients of the five-point molecule system of equations in the arrays A, B, C, D and E, and the source terms in the array Q. The routine derives the residual of the latest approximate solution and then uses the approximate LU factorization of the Strongly Implicit Procedure with the necessary acceleration parameter adjustment by calling D03UAF at each iteration. D03EBF combines the newly derived change with the old approximation to obtain the new approximate solution for t. The new solution is checked for convergence against the user-supplied convergence criteria and if these have not been achieved the iterative cycle is repeated. Convergence is based on both the maximum absolute normalised residuals (calculated with reference to the previous approximate solution as these are calculated at the commencement of each iteration) and on the maximum absolute change made to the values of t.

Problems in topologically non-rectangular regions can be solved using the routine by surrounding the region by a circumscribing topological rectangle. The equations for the nodal values external to the region of interest are set to zero (i.e., $c_{ij} = t_{ij} = 0$) and the boundary conditions are incorporated into the equations for the appropriate nodes.

If there is no better initial approximation when starting the iterative cycle, an array of all zeros can be used as the initial approximation.

The routine can be used to solve linear elliptic equations in which case the arrays A, B, C, D, E and Q are constants and for which a single call provides the required solution. It can also be used to solve nonlinear elliptic equations in which case some or all of these arrays may require updating during the progress of the iterations as more accurate solutions are derived. The routine will then have to be called repeatedly in an outer iterative cycle. Dependent on the nonlinearity, some under relaxation of the coefficients and/or source terms may be needed during their recalculation using the new estimates of the solution.

The routine can also be used to solve each step of a time-dependent parabolic equation in two space dimensions. The solution at each time step can be expressed in terms of an elliptic equation if the Crank–Nicolson or other form of implicit time integration is used.

Neither diagonal dominance, nor positive definiteness, of the matrix M formed from the arrays A, B, C, D, E is necessary to ensure convergence.

For problems in which the solution is not unique in the sense that an arbitrary constant can be added to the solution, for example Laplace's equation with all Neumann boundary conditions, a parameter is incorporated so that the solution can be rescaled by subtracting a specified nodal value from the whole solution t after the completion of every iteration to keep rounding errors to a minimum for those cases when the convergence is slow.

4   References

Jacobs D A H (1972) The strongly implicit procedure for the numerical solution of parabolic and elliptic partial differential equations *Note RD/L/N66/72* Central Electricity Research Laboratory

Stone H L (1968) Iterative solution of implicit approximations of multi-dimensional partial differential equations *SIAM J. Numer. Anal.* **5** 530–558

5   Parameters

1:     N1 – INTEGER   *Input*

On entry: the number of nodes in the first co-ordinate direction, n_1.

Constraint: N1 > 1.

2:      N2 – INTEGER *Input*

On entry: the number of nodes in the second co-ordinate direction, n_2.

Constraint: N2 > 1.

3:      N1M – INTEGER *Input*

On entry: the first dimension of the arrays A, B, C, D, E, Q, T, WRKSP1, WRKSP2 and WRKSP3 as declared in the (sub)program from which D03EBF is called.

Constraint: N1M ≥ N1.

4:      A(N1M,N2) – ***real*** array *Input*

On entry: A(i,j) must contain the coefficient of the 'southerly' term involving $t_{i,j-1}$ in the (i,j)th equation of the system (1), for $i = 1,2,...,$N1; $j = 1,2,...,$N2. The elements of A for $j = 1$ must be zero after incorporating the boundary conditions, since they involve nodal values from outside the rectangle.

5:      B(N1M,N2) – ***real*** array *Input*

On entry: B(i,j) must contain the coefficient of the 'westerly' term involving $t_{i-1,j}$ in the (i,j)th equation of the system (1), for $i = 1,2,...,$N1; $j = 1,2,...,$N2. The elements of B for $i = 1$ must be zero after incorporating the boundary conditions, since they involve nodal values from outside the rectangle.

6:      C(N1M,N2) – ***real*** array *Input*

On entry: C(i,j) must contain the coefficient of the 'central' term involving t_{ij} in the (i,j)th equation of the system (1), for $i = 1,2,...,$N1; $j = 1,2,...,$N2. The elements of C are checked to ensure that they are non-zero. If any element is found to be zero, the corresponding algebraic equation is assumed to be $t_{ij} = q_{ij}$. This feature can be used to define the equations for nodes at which, for example, Dirichlet boundary conditions are applied, or for nodes external to the problem of interest, by setting C$(i,j) = 0.0$ at appropriate points, and the corresponding value of Q(i,j) to the appropriate value, namely the prescribed value of T(i,j) in the Dirichlet case or zero at an external point.

7:      D(N1M,N2) – ***real*** array *Input*

On entry: D(i,j) must contain the coefficient of the 'easterly' term involving $t_{i+1,j}$ in the (i,j)th equation of the system (1), for $i = 1,2,...,$N1; $j = 1,2,...,$N2. The elements of D for $i = $ N1 must be zero after incorporating the boundary conditions, since they involve nodal values from outside the rectangle.

8:      E(N1M,N2) – ***real*** array *Input*

On entry: E(i,j) must contain the coefficient of the 'northerly' term involving $t_{i,j+1}$ in the (i,j)th equation of the system (1), for $i = 1,2,...,$N1; $j = 1,2,...,$N2. The elements of E for $j = $ N2 must be zero after incorporating the boundary conditions, since they involve nodal values from outside the rectangle.

9:      Q(N1M,N2) – ***real*** array *Input*

On entry: Q(i,j) must contain q_{ij} for $i = 1,2,...,$N1; $j = 1,2,...,$N2, i.e., the source term values at the nodal points for the system (1).

10:     T(N1M,N2) – ***real*** array *Input/Output*

On entry: T(i,j) must contain the element t_{ij} of the approximate solution to the equations supplied by the calling program as an initial starting value, for $i = 1,2,...,$N1; $j = 1,2,...,$N2.

If no better approximation is known, an array of zeros can be used.

On exit: the solution derived by the routine.

11:     APARAM – **real**   *Input*

On entry: the iteration acceleration factor. A value of 1.0 is adequate for most typical problems. However, if convergence is slow, the value can be reduced, typically to 0.2 or 0.1. If divergence is obtained, the value can be increased, typically to 2.0, 5.0 or 10.0.

Constraint: $0.0 < \text{APARAM} \leq ((\text{N1} - 1)^2 + (\text{N2} - 1)^2)/2.0$.

12:     ITMAX – INTEGER   *Input*

On entry: the maximum number of iterations to be used by the routine in seeking the solution. A reasonable value might be 30 if N1 = N2 = 10 or 100 if N1 = N2 = 50.

13:     ITCOUN – INTEGER   *Input/Output*

On entry: on the first call of D03EBF, ITCOUN must be set to 0. On subsequent entries, its value must be unchanged from the previous call.

On exit: its value is increased by the number of iterations used on this call (namely ITUSED). It therefore stores the accumulated number of iterations actually used. For subsequent calls for the same problem, i.e., with the same N1 and N2 but possibly different coefficients and/or source terms, as occur with nonlinear systems or with time-dependent systems, ITCOUN is the accumulated number of iterations. This applies to the second and subsequent calls to D03EBF. In this way a suitable cycling of the sequence of iteration parameters is obtained in the calls to D03UAF.

14:     ITUSED – INTEGER   *Output*

On exit: the number of iterations actually used on that call.

15:     NDIR – INTEGER   *Input*

On entry: indicates whether or not the system of equations has a unique solution. For systems which have a unique solution, NDIR must be set to any non-zero value. For systems derived from, for example, Laplace's equation with all Neumann boundary conditions, i.e., problems in which an arbitrary constant can be added to the solution, NDIR should be set to 0 and the values of the next two parameters must be specified. For such problems the routine subtracts the value of the function derived at the node (IXN, IYN) from the whole solution after each iteration to reduce the possibility of large rounding errors. The user must also ensure that for such problems the appropriate consistency condition on the source terms Q is satisfied.

16:     IXN – INTEGER   *Input*

On entry: IXN is ignored unless NDIR is equal to zero, in which case it must specify the first index of the nodal point at which the solution is to be set to zero. The node should not correspond to a corner node, or to a node external to the region of interest.

17:     IYN – INTEGER   *Input*

On entry: IYN is ignored unless NDIR is equal to zero, in which case it must specify the second index of the nodal point at which the solution is to be set to zero. The node should not correspond to a corner node, or to a node external to the region of interest.

18:     CONRES – **real**   *Input*

On entry: the convergence criterion to be used on the maximum absolute value of the normalised residual vector components. The latter is defined as the residual of the algebraic equation divided by the central coefficient when the latter is not equal to 0.0, and defined as the residual when the central coefficient is zero.

Clearly CONRES should not be less than a reasonable multiple of the **machine precision**.

19:     CONCHN – **real**   *Input*

On entry: the convergence criterion to be used on the maximum absolute value of the change made at each iteration to the elements of the array T, namely the dependent variable. Clearly CONCHN

should not be less than a reasonable multiple of the **machine precision** multiplied by the maximum value of T attained.

Convergence is achieved when both the convergence criteria are satisfied. The user can therefore set convergence on either the residual or on the change, or (as is recommended) on a requirement that both are below prescribed limits.

20: RESIDS(ITMAX) – **real** array   *Output*

On exit: the maximum absolute value of the residuals calculated at the ith iteration, for $i = 1,2,...,$ITUSED. The user who wants to know the maximum absolute residual of the solution which is returned must calculate this in the calling program. The sequence of values RESIDS indicates the rate of convergence.

21: CHNGS(ITMAX) – **real** array   *Output*

On exit: the maximum absolute value of the changes made to the components of the dependent variable T at the ith iteration, for $i = 1,2,...,$ITUSED. The sequence of values CHNGS indicates the rate of convergence.

22: WRKSP1(N1M,N2) – **real** array   *Workspace*
23: WRKSP2(N1M,N2) – **real** array   *Workspace*
24: WRKSP3(N1M,N2) – **real** array   *Workspace*

25: IFAIL – INTEGER   *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

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

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, for users 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.**

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, N1 < 2,
or   N2 < 2.

IFAIL $= 2$

On entry, N1M $<$ N1.

IFAIL $= 3$

On entry, APARAM ≤ 0.0.

IFAIL $= 4$

On entry, APARAM $> ((N1 - 1)^2 + (N2 - 1)^2)/2.0$.

IFAIL $= 5$

Convergence was not achieved after ITMAX iterations.

7   Accuracy

The improvement in accuracy for each iteration depends on the size of the system and on the condition of the up-date matrix characterised by the five-diagonal coefficient arrays. The ultimate accuracy obtainable depends on the above factors and on the ***machine precision***. The rate of convergence obtained with the Strongly Implicit Procedure is not always smooth because of the cyclic use of nine acceleration parameters. The convergence may become slow with very large problems, for example when N1 = N2 = 60. The final accuracy may be judged approximately from the rate of convergence determined from the sequence of values returned in CHNGS and the magnitude of the maximum absolute value of the change vector on the last iteration stored in CHNGS(ITUSED).

8   Further Comments

The time taken by the routine per iteration is approximately proportional to N1 × N2.

Convergence may not always be obtained when the problem is very large and/or the coefficients of the equations have widely disparate values. The latter case is often associated with a near ill-conditioned matrix.

9   Example

To solve Laplace's equation in a rectangle with a non-uniform grid spacing in the x and y co-ordinate directions and with Dirichlet boundary conditions specifying the function on the perimeter of the rectangle equal to

$$e^{(1.0+x)/y(n_2)} \times \cos(y/y(n_2)).$$

9.1   Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     D03EBF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER           N1, N2, N1M, ITMAX
      PARAMETER         (N1=6,N2=10,N1M=N1,ITMAX=18)
      INTEGER           NOUT
      PARAMETER         (NOUT=6)
*     .. Local Scalars ..
      real              APARAM, CONCHN, CONRES
      INTEGER           I, IFAIL, ITCOUN, ITUSED, IXN, IYN, J, NDIR
*     .. Local Arrays ..
      real              A(N1M,N2), B(N1M,N2), C(N1M,N2), CHNGS(ITMAX),
     +                  D(N1M,N2), E(N1M,N2), Q(N1M,N2), RESIDS(ITMAX),
     +                  T(N1M,N2), WRKSP1(N1M,N2), WRKSP2(N1M,N2),
     +                  WRKSP3(N1M,N2), X(N1M), Y(N2)
*     .. External Subroutines ..
      EXTERNAL          D03EBF
*     .. Intrinsic Functions ..
      INTRINSIC         COS, EXP
*     .. Data statements ..
      DATA              X(1), X(2), X(3), X(4), X(5), X(6)/0.0e0, 1.0e0,
     +                  3.0e0, 6.0e0, 10.0e0, 15.0e0/
      DATA              Y(1), Y(2), Y(3), Y(4), Y(5), Y(6), Y(7), Y(8),
     +                  Y(9), Y(10)/0.0e0, 1.0e0, 3.0e0, 6.0e0, 10.0e0,
     +                  15.0e0, 21.0e0, 28.0e0, 36.0e0, 45.0e0/
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D03EBF Example Program Results'
      WRITE (NOUT,*)
      APARAM = 1.0e0
      ITCOUN = 0
      NDIR = 1
      CONRES = 0.1e-5
      CONCHN = 0.1e-5
```

```
*        Set up difference equation coefficients, source terms and
*        initial conditions.
         DO 40 J = 1, N2
            DO 20 I = 1, N1
               IF ((I.NE.1) .AND. (I.NE.N1) .AND. (J.NE.1) .AND. (J.NE.N2))
     +             THEN
*              Specification for internal nodes
               A(I,J) = 2.0e0/((Y(J)-Y(J-1))*(Y(J+1)-Y(J-1)))
               E(I,J) = 2.0e0/((Y(J+1)-Y(J))*(Y(J+1)-Y(J-1)))
               B(I,J) = 2.0e0/((X(I)-X(I-1))*(X(I+1)-X(I-1)))
               D(I,J) = 2.0e0/((X(I+1)-X(I))*(X(I+1)-X(I-1)))
               C(I,J) = -A(I,J) - B(I,J) - D(I,J) - E(I,J)
               Q(I,J) = 0.0e0
               T(I,J) = 0.0e0
            ELSE
*              Specification for boundary nodes
               A(I,J) = 0.0e0
               B(I,J) = 0.0e0
               C(I,J) = 0.0e0
               D(I,J) = 0.0e0
               E(I,J) = 0.0e0
               Q(I,J) = EXP((X(I)+1.0e0)/Y(N2))*COS(Y(J)/Y(N2))
               T(I,J) = 0.0e0
            END IF
   20       CONTINUE
   40    CONTINUE
         WRITE (NOUT,*) 'Iteration      Maximum        Maximum'
         WRITE (NOUT,*) ' number        residual        change'
         IFAIL = 1
*
         CALL D03EBF(N1,N2,N1M,A,B,C,D,E,Q,T,APARAM,ITMAX,ITCOUN,ITUSED,
     +               NDIR,IXN,IYN,CONRES,CONCHN,RESIDS,CHNGS,WRKSP1,WRKSP2,
     +               WRKSP3,IFAIL)
*
         WRITE (NOUT,99999) (I,RESIDS(I),CHNGS(I),I=1,ITUSED)
*        Check error flag
         IF (IFAIL.EQ.0) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Table of calculated function values'
            WRITE (NOUT,*)
            WRITE (NOUT,*)
     +   ' I    1          2          3          4          5          6'
            WRITE (NOUT,*) ' J'
            DO 60 J = 1, N2
               WRITE (NOUT,99998) J, (T(I,J),I=1,N1)
   60       CONTINUE
         ELSE
            WRITE (NOUT,99997) 'Error in D03EBF  IFAIL =', IFAIL
         END IF
         STOP
*
99999 FORMAT (2X,I2,10X,e11.4,4X,e11.4)
99998 FORMAT (1X,I2,1X,6(F9.3,2X))
99997 FORMAT (1X,A,I4)
      END
```

9.2   Program Data

None.

9.3   Program Results

```
 D03EBF Example Program Results

 Iteration      Maximum        Maximum
  number        residual        change
    1          0.1427E+01     0.1427E+01
    2          0.6671E-02     0.2176E-01
    3          0.8422E-03     0.1621E-02
    4          0.7635E-04     0.1810E-03
```

```
        5              0.5434E-05      0.1199E-04
        6              0.6471E-06      0.1245E-05
        7              0.5467E-07      0.1081E-06

 Table of calculated function values

    I    1         2         3         4         5         6
    J
    1    1.022     1.045     1.093     1.168     1.277     1.427
    2    1.022     1.045     1.093     1.168     1.277     1.427
    3    1.020     1.043     1.091     1.166     1.274     1.424
    4    1.013     1.036     1.083     1.158     1.266     1.414
    5    0.997     1.020     1.066     1.140     1.246     1.392
    6    0.966     0.988     1.033     1.104     1.207     1.348
    7    0.913     0.934     0.976     1.044     1.141     1.274
    8    0.831     0.850     0.888     0.950     1.038     1.160
    9    0.712     0.728     0.762     0.814     0.890     0.994
   10    0.552     0.565     0.591     0.631     0.690     0.771
```