NAG Fortran Library Routine Document

D03EDF

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

D03EDF solves seven-diagonal systems of linear equations which arise from the discretization of an elliptic partial differential equation on a rectangular region. This routine uses a multigrid technique.

2 Specification

```
SUBROUTINE D03EDF(NGX, NGY, LDA, A, RHS, UB, MAXIT, ACC, US, U, IOUT,
1                 NUMIT, IFAIL)
INTEGER           NGX, NGY, LDA, MAXIT, IOUT, NUMIT, IFAIL
real              A(LDA,7), RHS(LDA), UB(NGX*NGY), ACC, US(LDA), U(LDA)
```

3 Description

D03EDF solves, by multigrid iteration, the seven-point scheme

$$\begin{array}{ccccccc} & A^6_{i,j}u_{i-1,j+1} & + & A^7_{i,j}u_{i,j+1} & & & \\ + & A^3_{i,j}u_{i-1,j} & + & A^4_{i,j}u_{ij} & + & A^5_{i,j}u_{i+1,j} & \\ & & + & A^1_{i,j}u_{i,j-1} & + & A^2_{i,j}u_{i+1,j-1} = f_{ij}, & i = 1,2,\ldots,n_x;\ j = 1,2,\ldots,n_y, \end{array}$$

which arises from the discretization of an elliptic partial differential equation of the form

$$\alpha(x,y)U_{xx} + \beta(x,y)U_{xy} + \gamma(x,y)U_{yy} + \delta(x,y)U_x + \epsilon(x,y)U_y + \phi(x,y)U = \psi(x,y)$$

and its boundary conditions, defined on a rectangular region. This we write in matrix form as

$$Au = f.$$

The algorithm is described in separate reports by Wesseling (1982a), Wesseling (1982b) and McCarthy (1983).

Systems of linear equations, matching the seven-point stencil defined above, are solved by a multigrid iteration. An initial estimate of the solution must be provided by the user. A zero guess may be supplied if no better approximation is available.

A 'smoother' based on incomplete Crout decomposition is used to eliminate the high frequency components of the error. A restriction operator is then used to map the system on to a sequence of coarser grids. The errors are then smoothed and prolongated (mapped onto successively finer grids). When the finest cycle is reached, the approximation to the solution is corrected. The cycle is repeated for MAXIT iterations or until the required accuracy, ACC, is reached.

D03EDF will automatically determine the number l of possible coarse grids, 'levels' of the multigrid scheme, for a particular problem. In other words, D03EDF determines the maximum integer l so that n_x and n_y can be expressed in the form

$$n_x = m2^{l-1} + 1, \quad n_y = n2^{l-1} + 1, \quad \text{with} \quad m \geq 2 \text{ and } n \geq 2.$$

It should be noted that the rate of convergence improves significantly with the number of levels used (see McCarthy (1983)), so that n_x and n_y should be carefully chosen so that $n_x - 1$ and $n_y - 1$ have factors of the form 2^l, with l as large as possible. For good convergence the integer l should be at least 2.

D03EDF has been found to be robust in application, but being an iterative method the problem of divergence can arise. For a strictly diagonally dominant matrix A

$$|A_{ij}^4| > \sum_{k \neq 4} |A_{ij}^k|, \quad i = 1,2,\ldots,n_x;\ j = 1,2,\ldots,n_y$$

no such problem is foreseen. The diagonal dominance of A is not a necessary condition, but should this condition be strongly violated then divergence may occur. The quickest test is to try the routine.

4 References

McCarthy G J (1983) Investigation into the multigrid code MGD1 *Report AERE-R 10889* Harwell

Wesseling P (1982a) MGD1 – A robust and efficient multigrid method *Multigrid Methods. Lecture Notes in Mathematics* **960** 614–630 Springer-Verlag

Wesseling P (1982b) Theoretical aspects of a multigrid method *SIAM J. Sci. Statist. Comput.* **3** 387–407

5 Parameters

1: NGX – INTEGER *Input*

On entry: the number of interior grid points in the x-direction, n_x. NGX $-$ 1 should preferably be divisible by as high a power of 2 as possible.

Constraint: NGX $\geq$ 3.

2: NGY – INTEGER *Input*

On entry: the number of interior grid points in the y-direction, n_y. NGY $-$ 1 should preferably be divisible by as high a power of 2 as possible.

Constraint: NGY $\geq$ 3.

3: LDA – INTEGER *Input*

On entry: the first dimension of the array A="x(82)="x(82) declared in the (sub)program from which D03EDF is called., which must also be a lower bound for the dimensions of the arrays RHS, US and U. It is always sufficient to set LDA $\geq (4 \times (\text{NGX} + 1) \times (\text{NGY} + 1))/3$, but slightly smaller values may be permitted, depending on the values of NGX and NGY. If on entry, LDA is too small, an error message gives the minimum permitted value. (LDA must be large enough to allow space for the coarse-grid approximations.)

4: A(LDA,7) – ***real*** array *Input/Output*

On entry: A$(i + (j-1) \times \text{NGX}, k)$ must be set to A_{ij}^k, for i = 1,2,...,NGX; j = 1,2,...,NGY and k = 1,2,...,7.

On exit: A is overwritten.

5: RHS(LDA) – ***real*** array *Input/Output*

On entry: RHS$(i + (j-1) \times \text{NGX})$ must be set to f_{ij}, for i = 1,2,...,NGX; j = 1,2,...,NGY.

On exit: the first NGX $\times$ NGY elements are unchanged and the rest of the array is used as workspace.

6: UB(NGX*NGY) – ***real*** array *Input/Output*

On entry: UB$(i + (j-1) \times \text{NGX})$ must be set to the initial estimate for the solution u_{ij}.

On exit: the corresponding component of the residual $r = f - \text{A}u$.

7: MAXIT – INTEGER *Input*

On entry: the maximum permitted number of multigrid iterations. If MAXIT $= 0$, no multigrid iterations are performed, but the coarse-grid approximations and incomplete Crout decompositions are computed, and may be output if IOUT is set accordingly.

Constraint: MAXIT ≥ 0.

8: ACC – ***real*** *Input*

On entry: the required tolerance for convergence of the residual 2-norm:

$$\|r\|_2 = \sqrt{\sum_{k=1}^{\text{NGX}\times\text{NGY}} (r_k)^2}$$

where $r = f - Au$ and u is the computed solution. Note that the norm is not scaled by the number of equations. The routine will stop after fewer than MAXIT iterations if the residual 2-norm is less than the specified tolerance. (If MAXIT > 0, at least one iteration is always performed.)

If on entry ACC $= 0.0$, then the ***machine precision*** is used as a default value for the tolerance; if ACC > 0.0, but ACC is less than the ***machine precision***, then the routine will stop when the residual 2-norm is less than the ***machine precision*** and IFAIL will be set to 4.

Constraint: ACC ≥ 0.0.

9: US(LDA) – ***real*** array *Output*

On exit: the residual 2-norm, stored in element US(1).

10: U(LDA) – ***real*** array *Output*

On exit: the computed solution u_{ij} is returned in U$(i + (j-1) \times \text{NGX})$, for $i = 1,2,...,\text{NGX}$; $j = 1,2,...,\text{NGY}$.

11: IOUT – INTEGER *Input*

On entry: controls the output of printed information to the advisory message unit as returned by X04ABF:

IOUT=0

No output.

IOUT=1

The solution u_{ij}, for $i = 1,2,...,\text{NGX}$; $j = 1,2,...,\text{NGY}$.

IOUT=2

The residual 2-norm after each iteration, with the reduction factor over the previous iteration.

IOUT=3

As for IOUT $= 1$ and IOUT $= 2$.

IOUT=4

As for IOUT=3, plus the final residual (as returned in UB).

IOUT=5

As for IOUT=4, plus the initial elements of A and RHS.

IOUT=6

As for IOUT=5, plus the Galerkin coarse grid approximations.

IOUT=7

As for IOUT=6, plus the incomplete Crout decompositions.

IOUT=8

As for IOUT=7, plus the residual after each iteration.

The elements A(p,k), the Galerkin coarse grid approximations and the incomplete Crout decompositions are output in the format:

Y-index $= j$

X-index $= i$A$(p,1)$ A$(p,2)$ A$(p,3)$ A$(p,4)$ A$(p,5)$ A$(p,6)$ A$(p,7)$

where $p = i + (j-1) \times$ NGX, i = 1,2,...,NGX and j = 1,2,...,NGY.

The vectors U(p), UB(p), RHS(p) are output in matrix form with NGY rows and NGX columns. Where NGX > 10, the NGX values for a given j-value are produced in rows of 10. Values of IOUT > 4 may yield considerable amounts of output.

Constraint: $0 \leq$ IOUT ≤ 8.

12: NUMIT – INTEGER *Output*

On exit: the number of iterations performed.

13: 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, NGX < 3,
or NGY < 3,
or LDA is too small,
or ACC < 0.0,
or MAXIT < 0,
or IOUT < 0,
or IOUT > 8.

IFAIL $= 2$

MAXIT iterations have been performed with the residual 2-norm decreasing at each iteration but the residual 2-norm has not been reduced to less than the specified tolerance (see ACC). Examine the progress of the iteration by setting IOUT ≥ 2.

IFAIL $= 3$

As for IFAIL $= 2$, except that at one or more iterations the residual 2-norm did not decrease. It is likely that the method fails to converge for the given matrix A.

IFAIL = 4

On entry, ACC is less than the ***machine precision***. The routine terminated because the residual norm is less than the ***machine precision***.

7 Accuracy

See ACC (Section 5).

8 Further Comments

The rate of convergence of this routine is strongly dependent upon the number of levels, l, in the multigrid scheme, and thus the choice of NGX and NGY is very important. The user is advised to experiment with different values of NGX and NGY to see the effect they have on the rate of convergence; for example, using a value such as NGX = 65 $(= 2^6 + 1)$ followed by NGX = 64 (for which $l = 1$).

9 Example

The program solves the elliptic partial differential equation

$$U_{xx} - \alpha U_{xy} + U_{yy} = -4, \quad \alpha = 1.7$$

on the unit square $0 \leq x, y \leq 1$, with boundary conditions

$$\text{U} = 0 \text{ on } \begin{cases} x = 0, & (0 \leq y \leq 1) \\ y = 0, & (0 \leq x \leq 1) \\ y = 1, & (0 \leq x \leq 1) \end{cases} U = 1 \text{ on } x = 1, \quad 0 \leq y \leq 1.$$

For the equation to be elliptic, α must be less than 2.

The equation is discretized on a square grid with mesh spacing h in both directions using the following approximations:

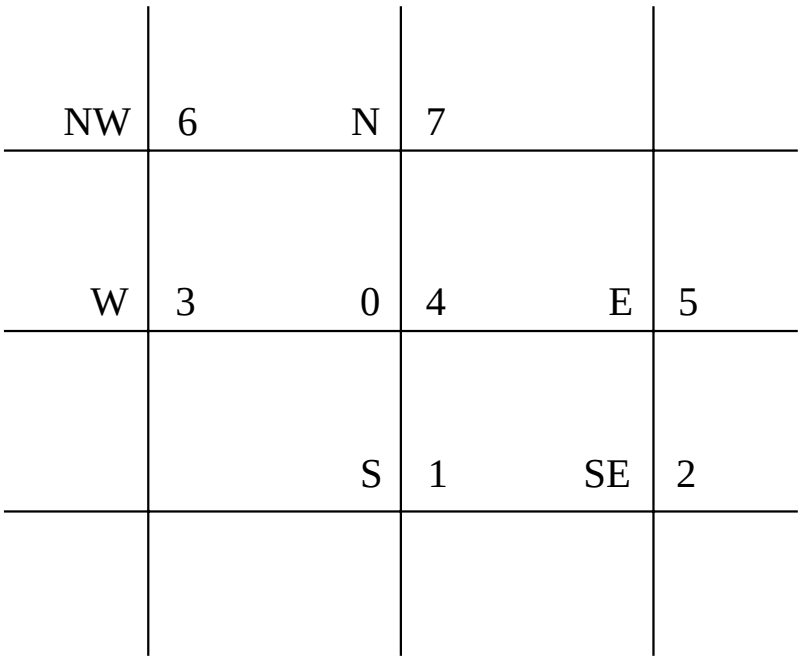

Figure 1

$$\begin{aligned} U_{xx} &\simeq \tfrac{1}{h^2}(U_{\mathrm{E}} - 2U_{\mathrm{O}} + U_{\mathrm{W}}) \\ U_{yy} &\simeq \tfrac{1}{h^2}(U_{\mathrm{N}} - 2U_{\mathrm{O}} + U_{\mathrm{S}}) \\ U_{xy} &\simeq \tfrac{1}{2h^2}(U_{\mathrm{N}} - U_{\mathrm{NW}} + U_{\mathrm{E}} - 2U_{\mathrm{O}} + U_{\mathrm{W}} - U_{\mathrm{SE}} + U_{\mathrm{S}}). \end{aligned}$$

Thus the following equations are solved:

$$\begin{array}{rcccl} & \frac{1}{2}\alpha u_{i-1,j+1} & + & (1-\frac{1}{2}\alpha)u_{i,j+1} & \\ + & (1-\frac{1}{2}\alpha)u_{i+1,j} & + & (-4+\alpha)u_{ij} \quad + & (1-\frac{1}{2}\alpha)u_{i+1,j} \\ & & + & (1-\frac{1}{2}\alpha)u_{i,j-1} \quad + & \frac{1}{2}\alpha u_{i+1,j-1} = -4h^2 \end{array}$$

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.

```
*     D03EDF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      real             ALPHA
      PARAMETER        (ALPHA=1.7e0)
      INTEGER          LEVELS, NGX, NGY, LDA
      PARAMETER        (LEVELS=3,NGX=2**LEVELS+1,NGY=NGX,LDA=4*(NGX+1)
     +                 *(NGY+1)/3)
*     .. Local Scalars ..
      real             ACC, HX, HY
      INTEGER          I, IFAIL, IOUT, IX, IY, J, K, MAXIT, NUMIT
*     .. Local Arrays ..
      real             A(LDA,7), RHS(LDA), U(LDA), UB(NGX*NGY), US(LDA)
*     .. External Subroutines ..
      EXTERNAL         D03EDF, X04ABF
*     .. Intrinsic Functions ..
      INTRINSIC        real
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D03EDF Example Program Results'
      WRITE (NOUT,*)
      ACC = 1.0e-4
      CALL X04ABF(1,NOUT)
      MAXIT = 15
*     ** Set IOUT.GE.2 to obtain intermediate output **
      IOUT = 0
      HX = 1.0e0/real(NGX+1)
      HY = 1.0e0/real(NGY+1)
      WRITE (NOUT,99999) 'NGX = ', NGX, '  NGY = ', NGY, '  ACC =', ACC,
     +  '  MAXIT = ', MAXIT
*     Set up operator, right-hand side and initial guess for
*     step-lengths HX and HY
      DO 40 J = 1, NGY
         DO 20 I = 1, NGX
            K = (J-1)*NGX + I
            A(K,1) = 1.0e0 - 0.5e0*ALPHA
            A(K,2) = 0.5e0*ALPHA
            A(K,3) = 1.0e0 - 0.5e0*ALPHA
            A(K,4) = -4.0e0 + ALPHA
            A(K,5) = 1.0e0 - 0.5e0*ALPHA
            A(K,6) = 0.5e0*ALPHA
            A(K,7) = 1.0e0 - 0.5e0*ALPHA
            RHS(K) = -4.0e0*HX*HY
            UB(K) = 0.0e0
   20    CONTINUE
   40 CONTINUE
*     Correction for the boundary conditions
*     Horizontal boundaries --
      DO 60 I = 2, NGX - 1
*        Boundary condition on Y=0 -- U=0
```

```
         IX = I
         A(IX,1) = 0.0e0
         A(IX,2) = 0.0e0
*        Boundary condition on Y=1 -- U=0
         IX = I + (NGY-1)*NGX
         A(IX,6) = 0.0e0
         A(IX,7) = 0.0e0
   60 CONTINUE
*     Vertical boundaries --
      DO 80 J = 2, NGY - 1
*        Boundary condition on X=0 -- U=0
         IY = (J-1)*NGX + 1
         A(IY,3) = 0.0e0
         A(IY,6) = 0.0e0
*        Boundary condition on X=1 -- U=1
         IY = J*NGX
         RHS(IY) = RHS(IY) - A(IY,5) - A(IY,2)
         A(IY,2) = 0.0e0
         A(IY,5) = 0.0e0
   80 CONTINUE
*     Now the four corners --
*     Bottom left corner
      K = 1
      A(K,1) = 0.0e0
      A(K,2) = 0.0e0
      A(K,3) = 0.0e0
      A(K,6) = 0.0e0
*     Top left corner
      K = 1 + (NGY-1)*NGX
      A(K,3) = 0.0e0
      A(K,6) = 0.0e0
      A(K,7) = 0.0e0
*     Bottom right corner
*     Use average value at discontinuity ( = 0.5 )
      K = NGX
      RHS(K) = RHS(K) - A(K,2)*0.5e0 - A(K,5)
      A(K,1) = 0.0e0
      A(K,2) = 0.0e0
      A(K,5) = 0.0e0
*     Top right corner
      K = NGX*NGY
      RHS(K) = RHS(K) - A(K,2) - A(K,5)
      A(K,2) = 0.0e0
      A(K,5) = 0.0e0
      A(K,6) = 0.0e0
      A(K,7) = 0.0e0
*     Solve the equations
      IFAIL = 0
*
      CALL D03EDF(NGX,NGY,LDA,A,RHS,UB,MAXIT,ACC,US,U,IOUT,NUMIT,IFAIL)
*
      WRITE (NOUT,*)
      WRITE (NOUT,99998) 'Residual norm =', US(1)
      WRITE (NOUT,99997) 'Number of iterations =', NUMIT
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Solution'
      WRITE (NOUT,*)
      WRITE (NOUT,99996) '  I/J', (I,I=1,NGX)
      DO 100 J = 1, NGY
         WRITE (NOUT,99995) J, (U(I+(J-1)*NGX),I=1,NGX)
  100 CONTINUE
      STOP
*
99999 FORMAT (1X,A,I3,A,I3,A,1P,e10.2,A,I3)
99998 FORMAT (1X,A,1P,e12.2)
99997 FORMAT (1X,A,I5)
99996 FORMAT (1X,A,10I7,:)
99995 FORMAT (1X,I3,2X,10F7.3,:)
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 D03EDF Example Program Results

 NGX =   9  NGY =   9  ACC =  1.00E-04  MAXIT =  15

 Residual norm =    1.61E-05
 Number of iterations =    4

 Solution

   I/J      1      2      3      4      5      6      7      8      9
   1    0.024  0.047  0.071  0.095  0.120  0.148  0.185  0.261  0.579
   2    0.047  0.094  0.142  0.192  0.245  0.310  0.412  0.636  0.913
   3    0.071  0.142  0.215  0.292  0.378  0.489  0.663  0.862  0.969
   4    0.095  0.191  0.289  0.393  0.511  0.656  0.810  0.915  0.967
   5    0.119  0.239  0.361  0.486  0.616  0.741  0.836  0.895  0.939
   6    0.143  0.284  0.419  0.543  0.648  0.729  0.786  0.832  0.893
   7    0.164  0.315  0.438  0.527  0.593  0.641  0.682  0.734  0.823
   8    0.174  0.306  0.378  0.427  0.462  0.492  0.528  0.591  0.717
   9    0.155  0.202  0.229  0.248  0.264  0.282  0.313  0.376  0.523
```