NAG Library Routine Document

D03UAF

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

D03UAF performs at each call one iteration of the Strongly Implicit Procedure. It is used to calculate on successive calls a sequence of approximate corrections to the current estimate of the solution when solving a system of simultaneous algebraic equations for which the iterative update matrix is of five-point molecule form on a two-dimensional topologically-rectangular mesh. ('Topological' means that a polar grid (r, θ), for example, can be used as it is equivalent to a rectangular box.)

2 Specification

```
SUBROUTINE D03UAF (N1, N2, LDA, A, B, C, D, E, APARAM, IT, R, WRKSP1,      &
                   WRKSP2, IFAIL)

INTEGER            N1, N2, LDA, IT, IFAIL
REAL (KIND=nag_wp) A(LDA,N2), B(LDA,N2), C(LDA,N2), D(LDA,N2),            &
                   E(LDA,N2), APARAM, R(LDA,N2), WRKSP1(LDA,N2),          &
                   WRKSP2(LDA,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 solution t may be obtained iteratively from a starting approximation $t^{(1)}$ by the formulae

$$\begin{aligned} r^{(n)} &= q - Mt^{(n)} \\ Ms^{(n)} &= r^{(n)} \\ t^{(n+1)} &= t^{(n)} + s^{(n)}. \end{aligned}$$

Thus $r^{(n)}$ is the residual of the nth approximate solution $t^{(n)}$, and $s^{(n)}$ is the update change vector.

D03UAF determines the approximate change vector s corresponding to a given residual r, i.e., it determines an approximate solution to a set of equations

$$Ms = r \tag{2}$$

where M is a square $(n_1 \times n_2)$ by $(n_1 \times n_2)$ matrix and r is a known vector of length $n_1 \times n_2$. The set of equations (2) must be of five-diagonal form

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

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

$$s_{ij} = r_{ij}.$$

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

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

The calling program supplies the current residual r at each iteration and the coefficients of the five-point molecule system of equations on which the update procedure is based. The routine performs one iteration, using the approximate LU factorization of the Strongly Implicit Procedure with the necessary acceleration parameter adjustment, to calculate the approximate solution s of the set of equations (2). The change s overwrites the residual array for return to the calling program. The calling program must combine this change stored in r with the old approximation to obtain the new approximate solution for t. It must then recalculate the residuals and, if the accuracy requirements have not been satisfied, commence the next iterative cycle.

Clearly there is no requirement that the iterative update matrix passed in the form of the five-diagonal element arrays A, B, C, D and E is the same as that used to calculate the residuals, and therefore the one governing the problem. However, the convergence may be impaired if they are not equal. Indeed, if the system of equations (1) is not precisely of the five-diagonal form illustrated above but has a few additional terms, then the methods of deferred or defect correction can be employed. The residual is calculated by the calling program using the full system of equations, but the update formula is based on a five-diagonal system (2) of the form given above. For example, the solution of a system of nine-diagonal equations each involving the combination of terms with $t_{i\pm1,j\pm1}, t_{i\pm1,j}, t_{i,j\pm1}$ and t_{ij} could use the five-diagonal coefficients on which to base the update, provided these incorporate the major features of the equations.

Problems in topologically non-rectangular regions can be solved using the routine by surrounding the region with a circumscribing topological rectangle. The equations for the nodal values external to the region of interest are set to zero (i.e., $c_{ij} = r_{ij} = 0$) 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, one can use an array of all zeros as the initial approximation from which the first set of residuals are determined.

The routine can be used to solve linear elliptic equations in which case the arrays A, B, C, D, E and the quantities q will be unchanged during the iterative cycles, or for solving nonlinear elliptic equations in which case some or all of these arrays may require updating as each new approximate solution is derived. Depending 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 (see Jacobs (1972)).

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 or of the update matrix formed from the arrays A, B, C, D and 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), the calling program should subtract a typical nodal value from the whole solution t at every iteration to keep rounding errors to a minimum.

4 References

Ames W F (1977) *Nonlinear Partial Differential Equations in Engineering* (2nd Edition) Academic Press

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 coordinate direction, n_1.

Constraint: N1 > 1.

2: N2 – INTEGER *Input*

On entry: the number of nodes in the second coordinate direction, n_2.

Constraint: N2 > 1.

3: LDA – INTEGER *Input*

On entry: the first dimension of the arrays A, B, C, D, E, R, WRKSP1 and WRKSP2 as declared in the (sub)program from which D03UAF is called.

Constraint: LDA $\geq$ N1.

4: A(LDA, N2) – REAL (KIND=nag_wp) array *Input*

On entry: A(i,j) must contain the coefficient of the 'southerly' term involving $s_{i,j-1}$ in the (i,j)th equation of the system (2), for $i = 1, 2, \ldots,$ N1 and $j = 1, 2, \ldots,$ 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(LDA, N2) – REAL (KIND=nag_wp) array *Input*

On entry: B(i,j) must contain the coefficient of the 'westerly' term involving $s_{i-1,j}$ in the (i,j)th equation of the system (2), for $i = 1, 2, \ldots,$ N1 and $j = 1, 2, \ldots,$ 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(LDA, N2) – REAL (KIND=nag_wp) array *Input*

On entry: C(i,j) must contain the coefficient of the 'central' term involving s_{ij} in the (i,j)th equation of the system (2), for $i = 1, 2, \ldots,$ N1 and $j = 1, 2, \ldots,$ N2. The elements of C are checked to ensure that they are nonzero. If any element is found to be zero, the corresponding algebraic equation is assumed to be $s_{ij} = r_{ij}$. 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. The corresponding value of R(i,j) is set equal to the appropriate value, namely the difference between the prescribed value of t_{ij} and the current value of t_{ij} in the Dirichlet case, or zero at an external point.

7: D(LDA, N2) – REAL (KIND=nag_wp) array *Input*

On entry: D(i,j) must contain the coefficient of the 'easterly' term involving $s_{i+1,j}$ in the (i,j)th equation of the system (2), for $i = 1, 2, \ldots,$ N1 and $j = 1, 2, \ldots,$ 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(LDA, N2) – REAL (KIND=nag_wp) array *Input*

On entry: E(i,j) must contain the coefficient of the 'northerly' term involving $s_{i,j+1}$ in the (i,j)th equation of the system (2), for $i = 1, 2, \ldots,$ N1 and $j = 1, 2, \ldots,$ 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: APARAM – REAL (KIND=nag_wp) *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: $0.0 < \text{APARAM} \leq \left((\text{N1}-1)^2 + (\text{N2}-1)^2\right)/2.0$.

10: IT – INTEGER *Input*

On entry: the iteration number. It must be initialized, but not necessarily to 1, before the first call, and must be incremented by one in the calling program for each subsequent call. D03UAF uses the counter to select the appropriate acceleration parameter from a sequence of nine, each one being used twice in succession. (Note that the acceleration parameter depends on the value of APARAM.)

11: R(LDA, N2) – REAL (KIND=nag_wp) array *Input/Output*

On entry: $\text{R}(i,j)$ must contain the current residual r_{ij} on the right-hand side of the (i,j)th equation of the system (2), for $i = 1, 2, \ldots, \text{N1}$ and $j = 1, 2, \ldots, \text{N2}$.

On exit: these residuals are overwritten by the corresponding components of solution s to the system (2), i.e., the changes to be made to the vector t to reduce the residuals supplied.

12: WRKSP1(LDA, N2) – REAL (KIND=nag_wp) array *Workspace*

13: WRKSP2(LDA, N2) – REAL (KIND=nag_wp) array *Workspace*

14: IFAIL – INTEGER *Input/Output*

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).

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, LDA $<$ N1.

IFAIL $= 3$

On entry, APARAM ≤ 0.0.

IFAIL $= 4$

On entry, APARAM $> \left((\text{N1}-1)^2+(\text{N2}-1)^2\right)/2.0$.

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

The improvement in accuracy for each iteration, i.e., on each call, depends on the size of the system and on the condition of the update matrix characterised by the five-diagonal coefficient arrays. The ultimate accuracy obtainable depends on the above factors and on the ***machine precision***. However, since D03UAF works with residuals and the update vector, the calling program can, in most cases where at each iteration all the residuals are usually of about the same size, calculate the residuals from extended precision values of the function, source term and equation coefficients if greater accuracy is required. 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. The final accuracy obtained can be judged approximately from the rate of convergence determined from the changes to the dependent variable t and in particular the change on the last iteration.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken is approximately proportional to N1 $\times$ N2 for each call.

When used with deferred or defect correction, the residual is calculated in the calling program from a different system of equations to those represented by the five-point molecule coefficients used by D03UAF as the basis of the iterative update procedure. When using deferred correction the overall rate of convergence depends not only on the items detailed in Section 7 but also on the difference between the two coefficient matrices used.

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 may be associated with an ill-conditioned matrix.

10 Example

This example solves Laplace's equation in a rectangle with a non-uniform grid spacing in the x and y coordinate directions and with Dirichlet boundary conditions specifying the function on the perimeter of the rectangle equal to $e^{(1.0+x)/y(n_2)} \times \cos(y/y(n_2))$.

10.1 Program Text

```
    Program d03uafe

!     D03UAF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: d03uaf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter    :: two = 2.0_nag_wp
      Real (Kind=nag_wp), Parameter    :: zero = 0.0_nag_wp
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: adel, aparam, ares, delmax, delmn,   &
                                          resmax, resmn
      Integer                          :: i, ifail, it, j, lda, n1, n2, nits
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:), b(:,:), c(:,:), d(:,:),      &
                                          e(:,:), q(:,:), r(:,:), t(:,:),      &
                                          wrksp1(:,:), wrksp2(:,:), x(:), y(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: abs, cos, exp, max, real
!     .. Executable Statements ..
      Write (nout,*) 'D03UAF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n1, n2, nits
      lda = n1
      Allocate (a(lda,n2),b(lda,n2),c(lda,n2),d(lda,n2),e(lda,n2),q(lda,n2), &
        r(lda,n2),t(lda,n2),wrksp1(lda,n2),wrksp2(lda,n2),x(n1),y(n2))
      Read (nin,*) x(1:n1)
      Read (nin,*) y(1:n2)
      aparam = one

!     Set up difference equation coefficients, source terms and
!     initial S
      a(1:n1,1:n2) = zero
      b(1:n1,1:n2) = zero
      d(1:n1,1:n2) = zero
      e(1:n1,1:n2) = zero
      q(1:n1,1:n2) = zero
      t(1:n1,1:n2) = zero
!     Specification for internal nodes
      Do j = 2, n2 - 1
        a(2:n1-1,j) = two/((y(j)-y(j-1))*(y(j+1)-y(j-1)))
        e(2:n1-1,j) = two/((y(j+1)-y(j))*(y(j+1)-y(j-1)))
      End Do
      Do i = 2, n1 - 1
        b(i,2:n2-1) = two/((x(i)-x(i-1))*(x(i+1)-x(i-1)))
        d(i,2:n2-1) = two/((x(i+1)-x(i))*(x(i+1)-x(i-1)))
      End Do
      c(1:n1,1:n2) = -a(1:n1,1:n2) - b(1:n1,1:n2) - d(1:n1,1:n2) - &
        e(1:n1,1:n2)
!     Specification for boundary nodes
      Do j = 1, n2
        q(1,j) = exp((x(1)+one)/y(n2))*cos(y(j)/y(n2))
        q(n1,j) = exp((x(n1)+one)/y(n2))*cos(y(j)/y(n2))
      End Do
      Do i = 1, n1
        q(i,1) = exp((x(i)+one)/y(n2))*cos(y(1)/y(n2))
        q(i,n2) = exp((x(i)+one)/y(n2))*cos(y(n2)/y(n2))
      End Do

!     Iterative loop
      Do it = 1, nits
```

```
!       Calculate the residuals
        resmax = zero
        resmn = zero
        Do j = 1, n2
          Do i = 1, n1
            If (c(i,j)/=zero) Then
!             Five point molecule formula
              r(i,j) = q(i,j) - a(i,j)*t(i,j-1) - b(i,j)*t(i-1,j) - &
                c(i,j)*t(i,j) - d(i,j)*t(i+1,j) - e(i,j)*t(i,j+1)
            Else
!             Explicit equation
              r(i,j) = q(i,j) - t(i,j)
            End If
            ares = abs(r(i,j))
            resmax = max(resmax,ares)
            resmn = resmn + ares
          End Do
        End Do
        resmn = resmn/(real(n1*n2,kind=nag_wp))

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call d03uaf(n1,n2,lda,a,b,c,d,e,aparam,it,r,wrksp1,wrksp2,ifail)

        If (it==1) Then
          Write (nout,99997) 'Iteration', 'Residual', 'Change'
          Write (nout,99996) 'No', 'Max.', 'Mean', 'Max.', 'Mean'
        End If

!       Update the dependent variable
        delmax = zero
        delmn = zero
        Do j = 1, n2
          Do i = 1, n1
            t(i,j) = t(i,j) + r(i,j)
            adel = abs(r(i,j))
            delmax = max(delmax,adel)
            delmn = delmn + adel
          End Do
        End Do
        delmn = delmn/(real(n1*n2,kind=nag_wp))
        Write (nout,99999) it, resmax, resmn, delmax, delmn
!     Convergence tests here if required
      End Do
!     End of iterative loop
      Write (nout,*)
      Write (nout,*) 'Table of calculated function values'
      Write (nout,*)
      Write (nout,99995) 'I', 1, (i,i=2,6)
      Write (nout,*) ' J'
      Do j = 1, n2
        Write (nout,99998) j, (t(i,j),i=1,n1)
      End Do


99999 Format (1X,I3,4(2X,E11.4))
99998 Format (1X,I2,1X,6(F9.3,2X))
99997 Format (1X,A,6X,A,19X,A)
99996 Format (3X,A,7X,A,8X,A,11X,A,6X,A/)
99995 Format (4X,A,4X,I1,5I11)
    End Program d03uafe
```

10.2 Program Data

```
D03UAF Example Program Data
 6  10  10                        : n1, n2, nits
 0.0  1.0  3.0  6.0  10.0  15.0   : x
 0.0  1.0  3.0  6.0  10.0  15.0
 21.0  28.0  36.0  45.0           : y
```

10.3 Program Results

```
D03UAF Example Program Results

Iteration      Residual                  Change
  No       Max.         Mean          Max.       Mean

   1   0.1427E+01   0.4790E+00   0.1427E+01   0.1031E+01
   2   0.1098E-02   0.3871E-03   0.2176E-01   0.6158E-02
   3   0.7364E-03   0.5926E-04   0.1621E-02   0.2475E-03
   4   0.2036E-04   0.2914E-05   0.1810E-03   0.2259E-04
   5   0.6946E-05   0.6214E-06   0.1199E-04   0.2347E-05
   6   0.2267E-06   0.4215E-07   0.1245E-05   0.2270E-06
   7   0.5625E-07   0.4500E-08   0.1081E-06   0.1761E-07
   8   0.2305E-08   0.3998E-09   0.1289E-07   0.1794E-08
   9   0.4733E-09   0.7397E-10   0.1422E-08   0.1841E-09
  10   0.7109E-10   0.8598E-11   0.3214E-09   0.2791E-10

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
```

Example Program
Laplace's Equation on a Non-uniform Grid