NAG Library Routine Document

F04YDF

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

F04YDF estimates the 1-norm of a real rectangular matrix without accessing the matrix explicitly. It uses reverse communication for evaluating matrix products. The routine may be used for estimating condition numbers of square matrices.

2 Specification

```
SUBROUTINE F04YDF (IREVCM, M, N, X, LDX, Y, LDY, ESTNRM, T, SEED, WORK,    &
                   IWORK, IFAIL)
INTEGER            IREVCM, M, N, LDX, LDY, T, SEED, IWORK(2*N+5*T+20),     &
                   IFAIL
REAL (KIND=nag_wp) X(LDX,*), Y(LDY,*), ESTNRM, WORK(M*T)
```

3 Description

F04YDF computes an estimate (a lower bound) for the 1-norm

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}| \tag{1}$$

of an m by n real matrix $A = (a_{ij})$. The routine regards the matrix A as being defined by a user-supplied 'Black Box' which, given an $n \times t$ matrix X (with $t \ll n$) or an $m \times t$ matrix Y, can return AX or $A^{\mathrm{T}}Y$. A reverse communication interface is used; thus control is returned to the calling program whenever a matrix product is required.

Note: this routine is **not recommended** for use when the elements of A are known explicitly; it is then more efficient to compute the 1-norm directly from formula (1) above.

The **main use** of the routine is for estimating $\|B^{-1}\|_1$ for a square matrix, B, and hence the **condition number** $\kappa_1(B) = \|B\|_1 \|B^{-1}\|_1$, without forming B^{-1} explicitly ($A = B^{-1}$ above).

If, for example, an LU factorization of B is available, the matrix products $B^{-1}X$ and $B^{-\mathrm{T}}Y$ required by F04YDF may be computed by back- and forward-substitutions, without computing B^{-1}.

The routine can also be used to estimate 1-norms of matrix products such as $A^{-1}B$ and ABC, without forming the products explicitly. Further applications are described by Higham (1988).

Since $\|A\|_\infty = \|A^{\mathrm{T}}\|_1$, F04YDF can be used to estimate the ∞-norm of A by working with A^{T} instead of A.

The algorithm used is described in Higham and Tisseur (2000).

4 References

Higham N J (1988) FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation *ACM Trans. Math. Software* **14** 381–396

Higham N J and Tisseur F (2000) A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra *SIAM J. Matrix. Anal. Appl.* **21** 1185–1201

5 Arguments

Note: this routine uses **reverse communication.** Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the argument **IREVCM**. Between intermediate exits and re-entries, **all arguments other than X and Y must remain unchanged**.

1: IREVCM – INTEGER *Input/Output*

On initial entry: must be set to 0.

On intermediate exit: IREVCM $= 1$ or 2, and X contains the $n \times t$ matrix X and Y contains the $m \times t$ matrix Y. The calling program must

(a) if IREVCM $= 1$, evaluate AX and store the result in Y
or
if IREVCM $= 2$, evaluate $A^{\mathrm{T}}Y$ and store the result in X,

(b) call F04YDF once again, with all the other arguments unchanged.

On intermediate re-entry: IREVCM must be unchanged.

On final exit: IREVCM $= 0$.

2: M – INTEGER *Input*

On entry: the number of rows of the matrix A.

Constraint: M ≥ 0.

3: N – INTEGER *Input*

On entry: n, the number of columns of the matrix A.

Constraint: N ≥ 0.

4: X(LDX, $*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array X must be at least $\max(1, \mathrm{T})$.

On initial entry: need not be set.

On intermediate exit: if IREVCM $= 1$, contains the current matrix X.

On intermediate re-entry: if IREVCM $= 2$, must contain $A^{\mathrm{T}}Y$.

On final exit: the array is undefined.

5: LDX – INTEGER *Input*

On initial entry: the leading dimension of the array X as declared in the (sub)program from which F04YDF is called.

Constraint: LDX $\geq$ N.

6: Y(LDY, $*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array Y must be at least $\max(1, \mathrm{T})$.

On initial entry: need not be set.

On intermediate exit: if IREVCM $= 2$, contains the current matrix Y.

On intermediate re-entry: if IREVCM $= 1$, must contain AX.

On final exit: the array is undefined.

7: LDY – INTEGER *Input*

On initial entry: the leading dimension of the array Y as declared in the (sub)program from which F04YDF is called.

Constraint: LDY $\geq$ M.

8: ESTNRM – REAL (KIND=nag_wp) *Input/Output*

On initial entry: need not be set.

On intermediate re-entry: must not be changed.

On final exit: an estimate (a lower bound) for $\|A\|_1$.

9: T – INTEGER *Input*

On entry: the number of columns t of the matrices X and Y. This is an argument that can be used to control the accuracy and reliability of the estimate and corresponds roughly to the number of columns of A that are visited during each iteration of the algorithm.

If T ≥ 2 then a partly random starting matrix is used in the algorithm.

Suggested value: T $= 2$.

Constraint: $1 \leq$ T $\leq$ M.

10: SEED – INTEGER *Input*

On entry: the seed used for random number generation.

If T $= 1$, SEED is not used.

Constraint: if T > 1, SEED ≥ 1.

11: WORK(M $\times$ T) – REAL (KIND=nag_wp) array *Communication Array*
12: IWORK($2 \times$ N $+ 5 \times$ T $+ 20$) – INTEGER array *Communication Array*

On initial entry: need not be set.

On intermediate re-entry: must not be changed.

13: IFAIL – INTEGER *Input/Output*

On initial entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, because for this routine the values of the output arguments may be useful even if IFAIL $\neq 0$ on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On final 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 $= 1$

Internal error; please contact NAG.

IFAIL $= -1$

On entry, IREVCM $= \langle value \rangle$.
Constraint: IREVCM $= 0$, 1 or 2.

On initial entry, IREVCM $= \langle value \rangle$.
Constraint: IREVCM $= 0$.

IFAIL $= -2$

On entry, M $= \langle value \rangle$.
Constraint: M ≥ 0.

IFAIL $= -3$

On entry, N $= \langle value \rangle$.
Constraint: N ≥ 0.

IFAIL $= -5$

On entry, LDX $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: LDX $\geq$ N.

IFAIL $= -7$

On entry, LDY $= \langle value \rangle$ and M $= \langle value \rangle$.
Constraint: LDY $\geq$ M.

IFAIL $= -9$

On entry, M $= \langle value \rangle$ and T $= \langle value \rangle$.
Constraint: $1 \leq$ T $\leq$ M.

IFAIL $= -10$

On entry, T $= \langle value \rangle$ and SEED $= \langle value \rangle$.
Constraint: if T > 1, SEED ≥ 1.

IFAIL $= -99$

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

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

In extensive tests on **random** matrices of size up to $m = n = 450$ the estimate ESTNRM has been found always to be within a factor two of $\|A\|_1$; often the estimate has many correct figures. However, matrices exist for which the estimate is smaller than $\|A\|_1$ by an arbitrary factor; such matrices are very unlikely to arise in practice. See Higham and Tisseur (2000) for further details.

8 Parallelism and Performance

F04YDF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

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

9.1 Timing

For most problems the time taken during calls to F04YDF will be negligible compared with the time spent evaluating matrix products between calls to F04YDF.

The number of matrix products required depends on the matrix A. At most six products of the form $Y = AX$ and five products of the form $X = A^{\mathrm{T}}Y$ will be required. The number of iterations is independent of the choice of t.

9.2 Overflow

It is your responsibility to guard against potential overflows during evaluation of the matrix products. In particular, when estimating $\|B^{-1}\|_1$ using a triangular factorization of B, F04YDF should not be called if one of the factors is exactly singular – otherwise division by zero may occur in the substitutions.

9.3 Choice of *t*

The argument t controls the accuracy and reliability of the estimate. For $t = 1$, the algorithm behaves similarly to the LAPACK estimator xLACON. Increasing t typically improves the estimate, without increasing the number of iterations required.

For $t \geq 2$, random matrices are used in the algorithm, so for repeatable results the same value of SEED should be used each time.

A value of $t = 2$ is recommended for new users.

9.4 Use in Conjunction with NAG Library Routines

To estimate the 1-norm of the inverse of a matrix A, the following skeleton code can normally be used:

```
      ...  code to factorize A ...
       IF (A is not singular) THEN
          IREVCM = 0
10        CALL F04YDF (IREVCM,M,N,X,LDX,Y,LDY,ESTNRM,T,SEED,WORK,   &
                       IWORK,IFAIL)
          IF (IREVCM.NE.0) THEN
             IF (IREVCM.EQ.1) THEN
                ...  code to compute Y=inv(A)X ...
             ELSE
                ...  code to compute X=inv(transpose(A))Y ...
             END IF
             GO TO 10
          END IF
       END IF
```

To compute $A^{-1}X$ or $A^{-\mathrm{T}}Y$, solve the equation $AY = X$ or $A^{\mathrm{T}}X = Y$, storing the result in Y or X respectively. The code will vary, depending on the type of the matrix A, and the NAG routine used to factorize A.

The factorization will normally have been performed by a suitable routine from Chapters F01, F03 or F07. Note also that many of the ‘Black Box’ routines in Chapter F04 for solving systems of equations also return a factorization of the matrix. The example program in Section 10 illustrates how F04YDF can be used in conjunction with NAG Library routines for LU factorization of a real matrix F07ADF (DGETRF).

It is straightforward to use F04YDF for the following other types of matrix, using the named routines for factorization and solution:

nonsymmetric tridiagonal (F01LEF and F04LEF);

nonsymmetric almost block-diagonal (F01LHF and F04LHF);

nonsymmetric band (F07BDF (DGBTRF) and F07BEF (DGBTRS));

symmetric positive definite (F07FDF (DPOTRF) and F07FEF (DPOTRS));

symmetric positive definite band (F07HDF (DPBTRF) and F07HEF (DPBTRS));

symmetric positive definite tridiagonal (F07JAF (DPTSV), F07JDF (DPTTRF) and F07JEF (DPTTRS));

symmetric positive definite variable bandwidth (F01MCF and F04MCF);

symmetric positive definite sparse (F11JAF and F11JBF);

symmetric indefinite (F07PDF (DSPTRF) and F07PEF (DSPTRS));

nonsymmetric sparse (F11MEF and F11MFF; note that F11MGF can also be used here).

For upper or lower triangular matrices, no factorization routine is needed: $Y = A^{-1}X$ and $X = A^{-\mathrm{T}}Y$ may be computed by calls to F06PJF (DTRSV) (or F06PKF (DTBSV) if the matrix is banded, or F06PLF (DTPSV) if the matrix is stored in packed form).

10 Example

For this routine two examples are provided. There is a single example program for F04YDF, with a main program and the code to solve the two example problems is given in Example 1 (EX1) and Example 2 (EX2).

Example 1 (EX1)

This example estimates the condition number $\|A\|_1\|A^{-1}\|_1$ of the matrix A given by

$$A = \begin{pmatrix} 0.7 & -0.2 & 1.0 & 0.0 & 2.0 & 0.1 \\ 0.3 & 0.7 & 0.0 & 1.0 & 0.9 & 0.2 \\ 0.0 & 0.0 & 0.2 & 0.7 & 0.0 & -1.1 \\ 0.0 & 3.4 & -0.7 & 0.2 & 0.1 & 0.1 \\ 0.0 & -4.0 & 0.0 & 1.0 & 9.0 & 0.0 \\ 0.4 & 1.2 & 4.3 & 0.0 & 6.2 & 5.9 \end{pmatrix}.$$

Example 2 (EX2)

This example estimates the condition number of the sparse matrix A (stored in symmetric coordinate storage format) given by

$$A = \begin{pmatrix} 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 3.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 2.0 & 0.0 & 2.0 & 0.0 \\ 2.0 & 0.0 & 4.0 & 0.0 & 5.0 \\ 0.0 & 1.0 & 2.0 & 0.0 & 0.0 \end{pmatrix}.$$

10.1 Program Text

```
    Program f04ydfe

!     F04YDF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nout = 6
!     .. Executable Statements ..
```

```
      Write (nout,*) 'F04YDF Example Program Results'

      Call ex1

      Call ex2

    Contains
      Subroutine ex1

!       .. Use Statements ..
        Use nag_library, Only: dgetrf, dgetrs, f04ydf, f06raf, nag_wp
!       .. Implicit None Statement ..
        Implicit None
!       .. Parameters ..
        Integer, Parameter             :: nin = 5, nout = 6
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: cond, nrma, nrminv
        Integer                        :: i, ifail, irevcm, lda, ldx, ldy, m,  &
                                          n, seed, t
!       .. Local Arrays ..
        Real (Kind=nag_wp), Allocatable :: a(:,:), work(:), x(:,:), y(:,:)
        Integer, Allocatable           :: ipiv(:), iwork(:)
!       .. Executable Statements ..
        Write (nout,*) 'Example 1'
        Write (nout,*)
!       Skip heading in data file
        Read (nin,'(///A)')
        Read (nin,*) m, n, t

        lda = m
        ldx = n
        ldy = m
        Allocate (a(lda,n),x(ldx,t),y(ldy,t),work(m*t),iwork(2*n+5*t+20),      &
          ipiv(n))

!       Read A from data file
        Read (nin,*)(a(i,1:n),i=1,m)

!       Compute 1-norm of A
        nrma = f06raf('1',m,n,a,lda,work)
        Write (nout,99999) 'The norm of A is: ', nrma

!       Estimate the norm of A^(-1) without explicitly forming A^(-1)

!       Perform an LU factorization so that A=LU where L and U are lower
!       and upper triangular.
        ifail = 0

!       The NAG name equivalent of dgetrf is f07adf
        Call dgetrf(m,n,a,lda,ipiv,ifail)

        seed = 354
        irevcm = 0
loop:   Do
          Call f04ydf(irevcm,m,n,x,ldx,y,ldy,nrminv,t,seed,work,iwork,ifail)
          If (irevcm==0) Then
            Exit loop
          Else If (irevcm==1) Then
!           Compute y = inv(A)*x

!           The NAG name equivalent of dgetrs is f07aef
            Call dgetrs('N',n,t,a,lda,ipiv,x,ldx,ifail)
!           x was overwritten by dgetrs, so set y=x
            y(1:n,1:t) = x(1:n,1:t)
          Else
!           Compute x = transpose(inv(A))*y

!           The NAG name equivalent of dgetrs is f07aef
            Call dgetrs('T',n,t,a,lda,ipiv,y,ldy,ifail)
!           y was overwritten by dgetrs so set x=y
            x(1:n,1:t) = y(1:n,1:t)
          End If
        End Do loop

        Write (nout,99999) 'The estimated norm of inverse(A) is: ', nrminv

!       Compute and print the estimated condition number
        cond = nrminv*nrma
        Write (nout,*)
        Write (nout,99999) 'Estimated condition number of A: ', cond
        Write (nout,*)

99999   Format (1X,A,F6.2)

      End Subroutine ex1

      Subroutine ex2

!       .. Use Statements ..
        Use nag_library, Only: f01brf, f04axf, f04ydf, nag_wp
!       .. Implicit None Statement ..
        Implicit None
!       .. Parameters ..
        Real (Kind=nag_wp), Parameter  :: zero = 0.0_nag_wp
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: asum, cond, nrma, nrminv, pivot,     &
                                          resid
        Integer                        :: i, ifail, irevcm, j, ldx, ldy, licn, &
                                          lirn, n, nin, nout, nz, seed, t
        Logical                        :: grow, lblock
!       .. Local Arrays ..
        Real (Kind=nag_wp), Allocatable :: a(:), w(:), work(:), x(:,:), y(:,:)
        Integer, Allocatable           :: icn(:), ikeep(:), irn(:), iw(:),     &
                                          iwork(:)
        Integer                        :: idisp(10)
        Logical                        :: abort(4)
!       .. Intrinsic Procedures ..
        Intrinsic                      :: abs, max
!       .. Executable Statements ..
        Continue
        nout = 6
        nin = 5
        Write (nout,'(//1X,A/)') 'Example 2'

!       Skip heading in data file
        Read (nin,'(//A)')

!       Input N, the order of the matrix A, and NZ the number of nonzero
!       elements of A, together with t the norm estimation parameter.
        Read (nin,*) n, nz, t
        licn = 4*nz
        lirn = 2*nz
        ldx = n
        ldy = n

!       Allocate the required memory
        Allocate (a(licn),icn(licn),irn(lirn),x(ldx,t),y(ldy,t),work(n*t),     &
          ikeep(5*n),iwork(2*n+5*t+20),iw(8*n),w(n))

!       Input the elements of A, along with row and column information.
        Read (nin,*)(a(i),irn(i),icn(i),i=1,nz)

!       Compute 1-norm of A
        nrma = zero
        Do i = 1, n
          asum = zero
          Do j = 1, nz
            If (icn(j)==i) Then
              asum = asum + abs(a(j))
            End If
          End Do
          nrma = max(nrma,asum)
        End Do
        Write (nout,99999) 'The norm of A is: ', nrma

!       Perform an LU factorization so that A=LU where L and U are lower
!       and upper triangular, using F01BRF
        pivot = 0.1_nag_wp
        grow = .True.
        lblock = .True.
        abort(1) = .True.
        abort(2) = .True.
        abort(3) = .False.
        abort(4) = .True.

        ifail = 0
        Call f01brf(n,nz,a,licn,irn,lirn,icn,pivot,ikeep,iw,w,lblock,grow,     &
          abort,idisp,ifail)

!       Compute an estimate of the 1-norm of inv(A)
        seed = 412
        irevcm = 0
loop:   Do
          Call f04ydf(irevcm,n,n,x,ldx,y,ldy,nrminv,t,seed,work,iwork,ifail)
          If (irevcm==0) Then
            Exit loop
          Else If (irevcm==1) Then
!           Compute y = inv(A)*x
            Do i = 1, t
              Call f04axf(n,a,licn,icn,ikeep,x(1,i),w,irevcm,idisp,resid)
            End Do
!           x was overwritten by f04axf, so set y=x
            y(1:n,1:t) = x(1:n,1:t)
          Else
!           Compute x = transpose(inv(A))*y
            Do i = 1, t
              Call f04axf(n,a,licn,icn,ikeep,y(1,i),w,irevcm,idisp,resid)
            End Do
!           y was overwritten by f04axf so set x=y
            x(1:n,1:t) = y(1:n,1:t)
          End If
        End Do loop

        Write (nout,99999) 'The estimated norm of inverse(A) is: ', nrminv

!       Compute and print the estimated condition number
        cond = nrminv*nrma
        Write (nout,*)
        Write (nout,99999) 'Estimated condition number of A: ', cond
        Write (nout,*)

99999   Format (1X,A,F6.2)

      End Subroutine ex2
    End Program f04ydfe
```

10.2 Program Data

```
F04YDF Example Program Data

Example 1

6      6      2                             :Values of M, N and t

0.7   -0.2    1.0    0.0    2.0    0.1
0.3    0.7    0.0    1.0    0.9    0.2
0.0    0.0    0.2    0.7    0.0   -1.1
0.0    3.4   -0.7    0.2    0.1    0.1
0.0   -4.0    0.0    1.0    9.0    0.0
0.4    1.2    4.3    0.0    6.2    5.9       :End of matrix A

Example 2
```

```
5      10     2                                 :Values of N, NZ and t

3.0  2  1
2.0  4  1
1.0  2  2
2.0  3  2
1.0  5  2
4.0  4  3
2.0  5  3
1.0  1  4
2.0  3  4
5.0  4  5                                       :End of matrix A
```

10.3 Program Results

```
 F04YDF Example Program Results
 Example 1

 The norm of A is:  18.20
 The estimated norm of inverse(A) is:   2.97

 Estimated condition number of A:  54.14



 Example 2

 The norm of A is:   6.00
 The estimated norm of inverse(A) is:   3.37

 Estimated condition number of A:  20.20
```