NAG Library Routine Document

F04YCF

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

F04YCF estimates the 1-norm of a real matrix without accessing the matrix explicitly. It uses reverse communication for evaluating matrix-vector products. The routine may be used for estimating matrix condition numbers.

2 Specification

```
SUBROUTINE F04YCF (ICASE, N, X, ESTNRM, WORK, IWORK, IFAIL)

INTEGER             ICASE, N, IWORK(N), IFAIL
REAL (KIND=nag_wp)  X(N), ESTNRM, WORK(N)
```

3 Description

F04YCF computes an estimate (a lower bound) for the 1-norm

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

of an n 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 input vector x, can return either of the matrix-vector products Ax or $A^{\mathrm{T}}x$. A reverse communication interface is used; thus control is returned to the calling program whenever a matrix-vector 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 $\|B^{-1}\|_1$, 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-vector products $B^{-1}x$ and $B^{-\mathrm{T}}x$ required by F04YCF 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 $\|A\|_\infty = \|A^{\mathrm{T}}\|_1$, F04YCF can be used to estimate the ∞-norm of A by working with A^{T} instead of A.

The algorithm used is based on a method given by Hager (1984) and is described by Higham (1988). A comparison of several techniques for condition number estimation is given by Higham (1987).

Note: F04YDF can also be used to estimate the norm of a real matrix. F04YDF uses a more recent algorithm than F04YCF and it is recommended that F04YDF be used in place of F04YCF.

4 References

Hager W W (1984) Condition estimates *SIAM J. Sci. Statist. Comput.* **5** 311–316

Higham N J (1987) A survey of condition number estimation for triangular matrices *SIAM Rev.* **29** 575–596

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

5 Parameters

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 parameter **ICASE**. Between intermediate exits and re-entries, **all parameters other than X must remain unchanged**.

1: ICASE – INTEGER *Input/Output*

On initial entry: must be set to 0.

On intermediate exit: ICASE $= 1$ or 2, and X(i), for $i = 1, 2, \ldots, n$, contain the elements of a vector x. The calling program must

(a) evaluate Ax (if ICASE $= 1$) or $A^{\mathrm{T}}x$ (if ICASE $= 2$),

(b) place the result in X, and

(c) call F04YCF once again, with all the other parameters unchanged.

On final exit: ICASE $= 0$.

2: N – INTEGER *Input*

On initial entry: n, the order of the matrix A.

Constraint: N ≥ 1.

3: X(N) – REAL (KIND=nag_wp) array *Input/Output*

On initial entry: need not be set.

On intermediate exit: contains the current vector x.

On intermediate re-entry: must contain Ax (if ICASE $= 1$) or $A^{\mathrm{T}}x$ (if ICASE $= 2$).

On final exit: the array is undefined.

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

On initial entry: need not be set.

On intermediate exit: should not be changed.

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

5: WORK(N) – REAL (KIND=nag_wp) array *Input/Output*

On initial entry: need not be set.

On final exit: contains a vector v such that $v = Aw$ where ESTNRM $= \|v\|_1/\|w\|_1$ (w is not returned). If $A = B^{-1}$ and ESTNRM is large, then v is an approximate null vector for B.

6: IWORK(N) – INTEGER array *Communication Array*

7: IFAIL – INTEGER *Input/Output*

On initial 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, because for this routine the values of the output parameters 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 $= 1$

On entry, $\mathrm{N} < 1$.

7 Accuracy

In extensive tests on **random** matrices of size up to $n = 100$ the estimate ESTNRM has been found always to be within a factor eleven 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 (1988) for further details.

8 Further Comments

8.1 Timing

The total time taken within F04YCF is proportional to n. For most problems the time taken during calls to F04YCF will be negligible compared with the time spent evaluating matrix-vector products between calls to F04YCF.

The number of matrix-vector products required varies from 4 to 11 (or is 1 if $n = 1$). In most cases 4 or 5 products are required; it is rare for more than 7 to be needed.

8.2 Overflow

It is your responsibility to guard against potential overflows during evaluation of the matrix-vector products. In particular, when estimating $\|B^{-1}\|_1$ using a triangular factorization of B, F04YCF should not be called if one of the factors is exactly singular – otherwise division by zero may occur in the substitutions.

8.3 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
         ICASE = 0
10       CALL F04YCF (ICASE,N,X,ESTNRM,WORK,IWORK,IFAIL)
         IF (ICASE.NE.0) THEN
            IF (ICASE.EQ.1) THEN
               ...  code to compute inv(A)*x ...
            ELSE
               ...  code to compute inv(transpose(A))*x ...
            END IF
            GO TO 10
         END IF
      END IF
```

To compute $A^{-1}x$ or $A^{-\mathrm{T}}x$, solve the equation $Ay = x$ or $A^{\mathrm{T}}y = x$ for y, overwriting y on x. The code will vary, depending on the type of the matrix A, and the NAG routine used to factorize A.

Note that if A is any type of **symmetric** matrix, then $A = A^{\mathrm{T}}$, and the code following the call of F04YCF can be reduced to:

```
IF (ICASE.NE.0) THEN
   ...  code to compute inv(A)*x ...
   GO TO 10
END IF
```

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 9 illustrates how F04YCF can be used in conjunction with NAG Library routines for two important types of matrix: full nonsymmetric matrices (factorized by F07ADF (DGETRF)) and sparse nonsymmetric matrices (factorized by F01BRF).

It is straightforward to use F04YCF 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 (F03BFF, or 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)).

For upper or lower triangular matrices, no factorization routine is needed: $A^{-1}x$ and $A^{-\mathrm{T}}x$ 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).

9 Example

For this routine two examples are presented. There is a single example program for F04YCF, 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)

To estimate the condition number $\|A\|_1 \|A^{-1}\|_1$ of the matrix A given by

$$A = \begin{pmatrix} 1.5 & 2.0 & 3.0 & -2.1 & 0.3 \\ 2.5 & 3.0 & -4.0 & 2.3 & -1.1 \\ 3.5 & 4.0 & 0.5 & -3.1 & -1.4 \\ -0.4 & -3.2 & -2.1 & 3.1 & 2.1 \\ 1.7 & 3.7 & 1.9 & -2.2 & -3.3 \end{pmatrix}.$$

Example 2 (EX2)

To estimate the condition number $\|A\|_1 \|A^{-1}\|_1$ of the matrix A given by

$$A = \begin{pmatrix} 5.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 & 2.0 & 0.0 & 0.0 \\ 0.0 & 2.0 & 3.0 & 0.0 & 0.0 & 0.0 \\ -2.0 & 0.0 & 0.0 & 1.0 & 1.0 & 0.0 \\ -1.0 & 0.0 & 0.0 & -1.0 & 2.0 & -3.0 \\ -1.0 & -1.0 & 0.0 & 0.0 & 0.0 & 6.0 \end{pmatrix}.$$

9.1 Program Text

```
    Program f04ycfe

!     F04YCF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nout = 6
!     .. Executable Statements ..
      Write (nout,*) 'F04YCF Example Program Results'

      Call ex1

      Call ex2

    Contains
      Subroutine ex1

!       .. Use Statements ..
        Use nag_library, Only: dasum, dgetrf, dgetrs, f04ycf, nag_wp
!       .. Parameters ..
        Real (Kind=nag_wp), Parameter    :: zero = 0.0E+0_nag_wp
        Integer, Parameter               :: nin = 5
!       .. Local Scalars ..
        Real (Kind=nag_wp)               :: anorm, cond, estnrm
        Integer                          :: i, icase, ifail, info, j, lda, n
!       .. Local Arrays ..
        Real (Kind=nag_wp), Allocatable  :: a(:,:), p(:), work(:), x(:)
        Integer, Allocatable             :: ipiv(:), iwork(:)
!       .. Intrinsic Procedures ..
        Intrinsic                        :: max
!       .. Executable Statements ..
        Write (nout,*)
        Write (nout,*)
        Write (nout,*) 'Example 1'
        Write (nout,*)
!       Skip heading in data file
        Read (nin,*)
        Read (nin,*)
        Read (nin,*)
        Read (nin,*) n
        lda = n
        Allocate (a(lda,n),p(n),work(n),x(n),iwork(n),ipiv(n))
        Read (nin,*)(a(i,1:n),i=1,n)
!       First compute the norm of A.
!       DASUM (f06ekf) returns the sum of the absolute values of a column of A.
        anorm = zero
        Do j = 1, n
          anorm = max(anorm,dasum(n,a(1,j),1))
        End Do
        Write (nout,99999) 'Computed norm of A =', anorm

!       Next estimate the norm of inverse(A). We do not form the
!       inverse explicitly.

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
!       LU Factorize A
!       The NAG name equivalent of dgetrf is f07adf
        Call dgetrf(n,n,a,lda,ipiv,info)

        icase = 0
loop:   Do
          ifail = 0
          Call f04ycf(icase,n,x,estnrm,work,iwork,ifail)
          If (icase/=0) Then
            If (icase==1) Then
!             Return the vector inv(A)*X
!             The NAG name equivalent of dgetrs is f07aef
              Call dgetrs('N',n,1,a,lda,ipiv,x,n,info)
            Else If (icase==2) Then
!             Return the vector inv(A)'*X
              Call dgetrs('T',n,1,a,lda,ipiv,x,n,info)
            End If
!         Continue until icase is returned as 0.
          Else
            Write (nout,99999) 'Estimated norm of inverse(A) =', estnrm
            cond = anorm*estnrm
            Write (nout,99998) 'Estimated condition number of A =', cond
            Write (nout,*)
            Exit loop
          End If
        End Do loop

99999   Format (1X,A,F8.4)
99998   Format (1X,A,F5.1)
      End Subroutine ex1
      Subroutine ex2

!       .. Use Statements ..
        Use nag_library, Only: f01brf, f04axf, f04ycf, nag_wp
!       .. Parameters ..
        Real (Kind=nag_wp), Parameter    :: tenth = 0.1E+0_nag_wp
        Real (Kind=nag_wp), Parameter    :: zero = 0.0E+0_nag_wp
        Integer, Parameter               :: nin = 5
!       .. Local Scalars ..
        Real (Kind=nag_wp)               :: anorm, cond, estnrm, resid, sum, u
        Integer                          :: i, icase, ifail, j, licn, lirn, n, &
                                            nz
        Logical                          :: grow, lblock
!       .. Local Arrays ..
        Real (Kind=nag_wp), Allocatable  :: a(:), w(:), work1(:), x(:)
        Integer, Allocatable             :: icn(:), ikeep(:), irn(:), iw(:),   &
                                            iwork(:)
        Integer                          :: idisp(10)
        Logical                          :: abort(4)
!       .. Intrinsic Procedures ..
        Intrinsic                        :: abs, max
!       .. Executable Statements ..
        Write (nout,*)
        Write (nout,*)
        Write (nout,*) 'Example 2'
        Write (nout,*)
!       Skip heading in data file
        Read (nin,*)
        Read (nin,*)
!       Input N, the order of matrix A, and NZ, the number of non-zero
!       elements of A.
        Read (nin,*) n, nz
        licn = 4*nz
        lirn = 2*nz
        Allocate (a(licn),w(n),work1(n),x(n),icn(licn),ikeep(5*n),irn(lirn), &
          iw(8*n),iwork(n))
!       Input the elements of A, along with row and column information.
        Read (nin,*)(a(i),irn(i),icn(i),i=1,nz)
!       First compute the norm of A.
        anorm = zero
        Do i = 1, n
          sum = zero
          Do j = 1, nz
            If (icn(j)==i) sum = sum + abs(a(j))
          End Do
          anorm = max(anorm,sum)
        End Do
        Write (nout,99999) 'Computed norm of A =', anorm
!       Next estimate the norm of inverse(A). We do not form the
!       inverse explicitly.
!       Factorise A into L*U using F01BRF.
        u = tenth
        lblock = .True.
        grow = .True.
        abort(1) = .True.
        abort(2) = .True.
        abort(3) = .False.
        abort(4) = .True.

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call f01brf(n,nz,a,licn,irn,lirn,icn,u,ikeep,iw,w,lblock,grow,abort, &
          idisp,ifail)

        icase = 0
loop:   Do
          ifail = 0
          Call f04ycf(icase,n,x,estnrm,work1,iwork,ifail)

          If (icase/=0) Then
!           Return X := inv(A)*X or X = inv(A)'*X, depending on the
!           value of ICASE, by solving A*Y = X or A'*Y = X,
!           overwriting Y on X.
            Call f04axf(n,a,licn,icn,ikeep,x,w,icase,idisp,resid)
!         Continue until icase is returned as 0.
          Else
            Write (nout,99999) 'Estimated norm of inverse(A) =', estnrm
            cond = anorm*estnrm
            Write (nout,99998) 'Estimated condition number of A =', cond
            Exit loop
          End If
        End Do loop

99999   Format (1X,A,F8.4)
99998   Format (1X,A,F5.1)
      End Subroutine ex2
    End Program f04ycfe
```

9.2 Program Data

```
F04YCF Example Program Data

Example 1
  5                             : n

  1.5   2.0   3.0  -2.1   0.3
  2.5   3.0  -4.0   2.3  -1.1
  3.5   4.0   0.5  -3.1  -1.4
 -0.4  -3.2  -2.1   3.1   2.1
  1.7   3.7   1.9  -2.2  -3.3    : matrix A

Example 2
 6  15                                                    : n, nz

 5.0  1  1   2.0  2  2  -1.0  2  3   2.0  2  4   3.0  3  3
-2.0  4  1   1.0  4  4   1.0  4  5  -1.0  5  1  -1.0  5  4
 2.0  5  5  -3.0  5  6  -1.0  6  1  -1.0  6  2   6.0  6  6  : matrix A
```

9.3 Program Results

```
 F04YCF Example Program Results


 Example 1

 Computed norm of A = 15.9000
 Estimated norm of inverse(A) =  1.7635
 Estimated condition number of A = 28.0
 Example 2

 Computed norm of A =  9.0000
 Estimated norm of inverse(A) =  1.9333
 Estimated condition number of A = 17.4
```