F04YCF – NAG Fortran Library Routine Document

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              X(N), ESTNRM, WORK(N)
```

3 Description

This routine 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^Tx. 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 the 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 $(B^{-1})^Tx$ 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 [2].

Since $\|A\|_\infty = \|A^T\|_1$, 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 [1] and is described by Higham [2]. A comparison of several techniques for condition number estimation is given by Higham [3].

4 References

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

[2] 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

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

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: ICASE 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 $A^T x$ (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 $\geq$ 1.

3: X(N) — ***real*** array *Input/Output*

On initial entry: X need not be set.

On intermediate exit: X contains the current vector x.

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

On final exit: the array is undefined.

4: ESTNRM — ***real*** *Output*

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

5: WORK(N) — ***real*** array *Input/Output*

On initial entry: WORK need not be set.

On final exit: WORK 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 *Workspace*

7: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (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 detected by the routine:

IFAIL = 1

On entry, 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 [2] for further details.

8 Further Comments

8.1 Timing

The total time taken within the routine 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 the responsibility of the user 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 Fortran 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 $(A^{-1})^T x$, solve the equation $Ay = x$ or $A^T y = x$ 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 $A = A^T$, 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 the F01 Chapter Introduction, the F03 Chapter Introduction or the F07 Chapter Introduction. Note also that many of the 'Black Box' routines in the Chapter Introduction 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 F03AFF) 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 (SGBTRF/DGBTRF) and F07BEF (SGBTRS/DGBTRS));

symmetric positive-definite (F03AEF and F04AGF, or F07FDF (SPOTRF/DPOTRF) and F07FEF (SPOTRS/DPOTRS));

symmetric positive-definite band (F07HDF (SPBTRF/DPBTRF) and F07HEF (SPBTRS/DPBTRS));

symmetric positive-definite tridiagonal (F04FAF);

symmetric positive-definite variable bandwidth (F01MCF and F04MCF);

symmetric positive-definite sparse (F11JAF and F11JBF);

symmetric indefinite (F07PDF (SSPTRF/DSPTRF) and F07PEF (SSPTRS/DSPTRS)).

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

9 Examples

For this routine two examples are presented, in Section 9.1 and Section 9.2. In the example program distributed to sites, there is a single example program for F04YCF, with a main program:

```
*     F04YCF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. External Subroutines ..
      EXTERNAL         EX1, EX2
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F04YCF Example Program Results'
      CALL EX1
      CALL EX2
      STOP
      END
```

The code to solve the two example problems is given in the subroutines EX1 and EX2, in Section 9.1.1 and Section 9.2.1 respectively.

9.1 Example 1

To estimate the condition number $\|A\|_1 \|A^{-1}\|_1$ of the order 5 matrix

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

The code to compute $A^{-1}x$ and $(A^{-1})^T x$ is more complicated than might be expected because there is no single routine to solve $A^T y = x$ for y in this case. Instead we make use of the fact that A is factorized by F03AFF as $PA = LU$, where P is a permutation matrix, L is lower triangular and U is upper triangular. Since the permutation matrix does not affect the 1-norm (i.e., $\|PA\|_1 = \|A\|_1$), it is sufficient to solve $LUy = x$ and $(LU)^T y = U^T L^T y = x$ for y, using calls to F06PJF.

9.1.1 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.

```
*
      SUBROUTINE EX1
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=20,LDA=NMAX)
      real             ZERO
      PARAMETER        (ZERO=0.0e+0)
*     .. Local Scalars ..
      real             ANORM, COND, D1, EPS, ESTNRM
      INTEGER          I, ICASE, ID, IFAIL, J, N
*     .. Local Arrays ..
      real             A(LDA,NMAX), P(NMAX), WORK(NMAX), X(NMAX)
      INTEGER          IWORK(NMAX)
*     .. External Functions ..
      real             F06EKF, X02AJF
      EXTERNAL         F06EKF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         F03AFF, F04YCF, F06PJF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX
*     .. Executable Statements ..
      WRITE (NOUT,*)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Example 1'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*)
      READ (NIN,*)
      READ (NIN,*) N
      WRITE (NOUT,*)
      IF (N.GT.NMAX) THEN
         WRITE (NOUT,99999) 'N is out of range: N =', N, '.'
      ELSE
         READ (NIN,*) ((A(I,J),J=1,N),I=1,N)
*        First compute the norm of A. F06EKF returns the sum of the
*        absolute values of a column of A.
         ANORM = ZERO
         DO 20 J = 1, N
            ANORM = MAX(ANORM,F06EKF(N,A(1,J),1))
   20    CONTINUE
         WRITE (NOUT,99998) 'Computed norm of A =', ANORM
*        Next estimate the norm of inverse(A). We do not form the
*        inverse explicitly.
         EPS = X02AJF()
         IFAIL = 0
*
*        Factorise A as P*A = L*U using F03AFF.
         CALL F03AFF(N,EPS,A,LDA,D1,ID,P,IFAIL)
         ICASE = 0
*
   40    CALL F04YCF(ICASE,N,X,ESTNRM,WORK,IWORK,IFAIL)
*
```

```
         IF (ICASE.NE.0) THEN
            IF (ICASE.EQ.1) THEN
*              Return the vector inv(P*A)*X by solving the equations
*              L*U*Y = X, overwriting Y on X. First solve L*Z = X for Z.
               CALL F06PJF('Lower','No Transpose','Non-Unit',N,A,LDA,X,
     +                     1)
*              Then solve U*Y = Z for Y.
               CALL F06PJF('Upper','No Transpose','Unit',N,A,LDA,X,1)
            ELSE IF (ICASE.EQ.2) THEN
*              Return the vector inv(P*A)'*X by solving U'*L'*Y = X,
*              overwriting Y on X. First solve U'*Z = X for Z.
               CALL F06PJF('Upper','Transpose','Unit',N,A,LDA,X,1)
*              Then solve L'*Y = Z for Y.
               CALL F06PJF('Lower','Transpose','Non-Unit',N,A,LDA,X,1)
            END IF
*           Continue until ICASE is returned as 0.
            GO TO 40
         ELSE
            WRITE (NOUT,99998) 'Estimated norm of inverse(A) =', ESTNRM
         END IF
         COND = ANORM*ESTNRM
         WRITE (NOUT,99997) 'Estimated condition number of A =', COND
         WRITE (NOUT,*)
      END IF
*
99999 FORMAT (1X,A,I5,A)
99998 FORMAT (1X,A,F8.4)
99997 FORMAT (1X,A,F5.1)
      END
```

9.1.2 Program Data

```
F04YCF Example Program Data

Example 1
  5                               :Value of 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    :End of matrix A
```

9.1.3 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
```

9.2 Example 2

To estimate the condition number $\|A\|_1\|A^{-1}\|_1$ of the order 6 matrix

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

9.2.1 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.

```
*
      SUBROUTINE EX2
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NZMAX, LICN, LIRN
      PARAMETER        (NMAX=20,NZMAX=25,LICN=4*NZMAX,LIRN=2*NZMAX)
      real             TENTH, ZERO
      PARAMETER        (TENTH=0.1e+0,ZERO=0.0e+0)
*     .. Local Scalars ..
      real             ANORM, COND, ESTNRM, RESID, SUM, U
      INTEGER          I, ICASE, IFAIL, J, N, NZ
      LOGICAL          GROW, LBLOCK
*     .. Local Arrays ..
      real             A(LICN), W(NMAX), WORK1(NMAX), X(NMAX)
      INTEGER          ICN(LICN), IDISP(10), IKEEP(5*NMAX), IRN(LIRN),
     +                 IW(8*NMAX), IWORK(NMAX)
      LOGICAL          ABORT(4)
*     .. External Subroutines ..
      EXTERNAL         F01BRF, F04AXF, F04YCF
*     .. Intrinsic Functions ..
      INTRINSIC        ABS, MAX
*     .. Executable Statements ..
      WRITE (NOUT,*)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Example 2'
*     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
      WRITE (NOUT,*)
      IF (N.GT.NMAX .OR. NZ.GT.NZMAX) THEN
         WRITE (NOUT,99999) 'N or NZ is out of range: N =', N,
     +     ',   NZ =', NZ, '.'
      ELSE
*        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 = 0
         DO 40 I = 1, N
            SUM = ZERO
```

```
            DO 20 J = 1, NZ
               IF (ICN(J).EQ.I) SUM = SUM + ABS(A(J))
   20       CONTINUE
            ANORM = MAX(ANORM,SUM)
   40    CONTINUE
         WRITE (NOUT,99998) '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 = 110
*
         CALL F01BRF(N,NZ,A,LICN,IRN,LIRN,ICN,U,IKEEP,IW,W,LBLOCK,GROW,
     +               ABORT,IDISP,IFAIL)
         ICASE = 0
*
   60    CALL F04YCF(ICASE,N,X,ESTNRM,WORK1,IWORK,IFAIL)
*
         IF (ICASE.NE.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.
            GO TO 60
         ELSE
            WRITE (NOUT,99998) 'Estimated norm of inverse(A) =', ESTNRM
         END IF
         COND = ANORM*ESTNRM
         WRITE (NOUT,99997) 'Estimated condition number of A =', COND
      END IF
*
99999 FORMAT (1X,A,I5,A,I5,A)
99998 FORMAT (1X,A,F8.4)
99997 FORMAT (1X,A,F5.1)
      END
```

9.2.2 Program Data

```
Example 2
 6  15                                                         :Values of N and 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  :End of matrix A
```

9.2.3 Program Results

```
Example 2

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