NAG Library Routine Document

F04ZDF

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

F04ZDF estimates the 1-norm of a complex 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 F04ZDF (IREVCM, M, N, X, LDX, Y, LDY, ESTNRM, T, SEED, WORK,    &
                   RWORK, IWORK, IFAIL)

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

3 Description

F04ZDF 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 complex 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 $A^{\mathrm{H}}Y$, where A^{H} is the complex conjugate transpose. 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 the formula (1) above.

The **main use** of the routine is for estimating $\left\|B^{-1}\right\|_1$ for a square matrix B, and hence the **condition number** $\kappa_1(B) = \|B\|_1 \left\|B^{-1}\right\|_1$, 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 $B^{-\mathrm{H}}Y$ required by F04ZDF 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 in Higham (1988).

Since $\|A\|_\infty = \|A^{\mathrm{H}}\|_1$, F04ZDF can be used to estimate the ∞-norm of A by working with A^{H} 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 $A^{\mathrm{H}}Y$ and store the result in X, where A^{H} is the complex conjugate transpose;

(b) call F04ZDF once again, with all the 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 initial entry: n, the number of columns of the matrix A.

Constraint: N ≥ 0.

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

Note: the second dimension of the array X must be at least max(1, 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 $A^{\mathrm{H}}Y$.

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 F04ZDF is called.

Constraint: LDX $\geq$ N.

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

Note: the second dimension of the array Y must be at least max(1, 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 F04ZDF 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) – COMPLEX (KIND=nag_wp) array *Communication Array*
12: RWORK($2 \times$ N) – REAL (KIND=nag_wp) array *Communication Array*
13: 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.

14: IFAIL – INTEGER *Input/Output*

On 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, if you are not familiar with this argument, 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 $= 1$

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

F04ZDF 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 F04ZDF will be negligible compared with the time spent evaluating matrix products between calls to F04ZDF.

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 $X = A^{\mathrm{H}}Y$ 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 $\left\|B^{-1}\right\|_1$ using a triangular factorization of B, F04ZDF 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 F04ZDF (IREVCM,M,N,X,LDX,Y,LDY,ESTNRM,T,SEED,WORK,  &
                       RWORK,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(herm(A))Y ...
             END IF
             GO TO 10
          END IF
       END IF
```

To compute $A^{-1}X$ or $A^{-\mathrm{H}}Y$, solve the equation $AY = X$ or $A^{\mathrm{H}}X = Y$ 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 example program in Section 10 illustrates how F04ZDF can be used in conjunction with NAG Library routine for LU factorization of complex matrices F07ARF (ZGETRF)).

It is also straightforward to use F04ZDF for Hermitian positive definite matrices, using F06TFF, F07FRF (ZPOTRF) and F07FSF (ZPOTRS) for factorization and solution.

For upper or lower triangular square matrices, no factorization routine is needed: $Y = A^{-1}X$ and $X = A^{-\mathrm{H}}Y$ may be computed by calls to F06SJF (ZTRSV) (or F06SKF (ZTBSV) if the matrix is banded, or F06SLF (ZTPSV) if the matrix is stored in packed form).

10 Example

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

$$A = \begin{pmatrix} 0.7+0.1i & -0.2+0.0i & 1.0+0.0i & 0.0+0.0i & 0.0+0.0i & 0.1+0.0i \\ 0.3+0.0i & 0.7+0.0i & 0.0+0.0i & 1.0+0.2i & 0.9+0.0i & 0.2+0.0i \\ 0.0+5.9i & 0.0+0.0i & 0.2+0.0i & 0.7+0.0i & 0.4+6.1i & 1.1+0.4i \\ 0.0+0.1i & 0.0+0.1i & -0.7+0.0i & 0.2+0.0i & 0.1+0.0i & 0.1+0.0i \\ 0.0+0.0i & 4.0+0.0i & 0.0+0.0i & 1.0+0.0i & 9.0+0.0i & 0.0+0.1i \\ 4.5+6.7i & 0.1+0.4i & 0.0+3.2i & 1.2+0.0i & 0.0+0.0i & 7.8+0.2i \end{pmatrix}.$$

10.1 Program Text

```
    Program f04zdfe

!     F04ZDF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f04zdf, f06uaf, nag_wp, zgetrf, zgetrs
!     .. 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 ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), work(:), x(:,:), y(:,:)
      Real (Kind=nag_wp), Allocatable  :: rwork(:)
      Real (Kind=nag_wp)               :: workr(1)
      Integer, Allocatable             :: ipiv(:), iwork(:)
!     .. Executable Statements ..
      Write (nout,*) 'F04ZDF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) m, n, t

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

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

!     Compute 1-norm of A
      nrma = f06uaf('1',m,n,a,lda,workr)
      Write (nout,99999) 'The norm of A is: ', nrma

!     Estimate the norm of A^(-1) without forming A^(-1)
      irevcm = 0
      ifail = 0
```

```
      seed = 652

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

!     The NAG name equivalent of zgetrf is f07arf
      Call zgetrf(m,n,a,lda,ipiv,ifail)

loop: Do
        Call f04zdf(irevcm,m,n,x,ldx,y,ldy,nrminv,t,seed,work,rwork,iwork,     &
          ifail)
        If (irevcm/=0) Then
          If (irevcm==1) Then
!           Compute Y = inv(A)*X

!           The NAG name equivalent of zgetrf is f07asf
            Call zgetrs('N',n,t,a,lda,ipiv,x,ldx,ifail)
!           X was overwritten by ZGETRS, so set Y=X
            y(1:n,1:t) = x(1:n,1:t)
          Else
!           Compute X = herm(inv(A))*Y

!           The NAG name equivalent of zgetrf is f07asf
            Call zgetrs('C',n,t,a,lda,ipiv,y,ldy,ifail)
!           Y was overwritten by ZGETRS, so set X=Y
            x(1:n,1:t) = y(1:n,1:t)
          End If
        Else

          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,*)
          Exit loop
        End If
      End Do loop
99999 Format (1X,A,F6.2)

    End Program f04zdfe
```

10.2 Program Data

```
F04ZDF Example Program Data

6    6    2                                                  :Values of M, N, t

(0.7,0.1) (-0.2,0.0) ( 1.0,0.0) (0.0,0.0) (0.0,0.0) (0.1,0.0)
(0.3,0.0) ( 0.7,0.0) ( 0.0,0.0) (1.0,0.2) (0.9,0.0) (0.2,0.0)
(0.0,5.9) ( 0.0,0.0) ( 0.2,0.0) (0.7,0.0) (0.4,6.1) (1.1,0.4)
(0.0,0.1) ( 0.0,0.1) (-0.7,0.0) (0.2,0.0) (0.1,0.0) (0.1,0.0)
(0.0,0.0) ( 4.0,0.0) ( 0.0,0.0) (1.0,0.0) (9.0,0.0) (0.0,0.1)
(4.5,6.7) ( 0.1,0.4) ( 0.0,3.2) (1.2,0.0) (0.0,0.0) (7.8,0.2) :End of matrix A
```

10.3 Program Results

```
 F04ZDF Example Program Results

 The norm of A is:  16.11
 The estimated norm of inverse(A) is:  24.02

 Estimated condition number of A: 387.08
```