NAG Library Routine Document

F07CVF (ZGTRFS)

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

F07CVF (ZGTRFS) computes error bounds and refines the solution to a complex system of linear equations $AX = B$ or $A^{\mathrm{T}}X = B$ or $A^{\mathrm{H}}X = B$, where A is an n by n tridiagonal matrix and X and B are n by r matrices, using the LU factorization returned by F07CRF (ZGTTRF) and an initial solution returned by F07CSF (ZGTTRS). Iterative refinement is used to reduce the backward error as much as possible.

2 Specification

```
SUBROUTINE F07CVF (TRANS, N, NRHS, DL, D, DU, DLF, DF, DUF, DU2, IPIV, B,       &
                   LDB, X, LDX, FERR, BERR, WORK, RWORK, INFO)

INTEGER                 N, NRHS, IPIV(*), LDB, LDX, INFO
REAL (KIND=nag_wp)      FERR(NRHS), BERR(NRHS), RWORK(N)
COMPLEX (KIND=nag_wp) DL(*), D(*), DU(*), DLF(*), DF(*), DUF(*), DU2(*),        &
                        B(LDB,*), X(LDX,*), WORK(2*N)
CHARACTER(1)            TRANS
```

The routine may be called by its LAPACK name ***zgtrfs***.

3 Description

F07CVF (ZGTRFS) should normally be preceded by calls to F07CRF (ZGTTRF) and F07CSF (ZGTTRS). F07CRF (ZGTTRF) uses Gaussian elimination with partial pivoting and row interchanges to factorize the matrix A as

$$A = PLU,$$

where P is a permutation matrix, L is unit lower triangular with at most one nonzero subdiagonal element in each column, and U is an upper triangular band matrix, with two superdiagonals. F07CSF (ZGTTRS) then utilizes the factorization to compute a solution, $\hat{X}$, to the required equations. Letting $\hat{x}$ denote a column of $\hat{X}$, F07CVF (ZGTRFS) computes a *component-wise backward error*, β, the smallest relative perturbation in each element of A and b such that $\hat{x}$ is the exact solution of a perturbed system

$$(A+E)\hat{x} = b + f, \quad \text{with} \quad |e_{ij}| \leq \beta|a_{ij}|, \quad \text{and} \quad |f_j| \leq \beta|b_j|.$$

The routine also estimates a bound for the *component-wise forward error* in the computed solution defined by $\max|x_i - \hat{x}_i| / \max|\hat{x}_i|$, where x is the corresponding column of the exact solution, X.

4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia http://www.netlib.org/lapack/lug

5 Parameters

1: TRANS – CHARACTER(1) *Input*

On entry: specifies the equations to be solved as follows:

TRANS = 'N'
: Solve $AX = B$ for X.

TRANS = 'T'
: Solve $A^{T}X = B$ for X.

TRANS = 'C'
: Solve $A^{H}X = B$ for X.

Constraint: TRANS = 'N', 'T' or 'C'.

2: N – INTEGER *Input*

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

Constraint: $N \geq 0$.

3: NRHS – INTEGER *Input*

On entry: r, the number of right-hand sides, i.e., the number of columns of the matrix B.

Constraint: $NRHS \geq 0$.

4: DL($*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the dimension of the array DL must be at least $\max(1, N-1)$.

On entry: must contain the $(n-1)$ subdiagonal elements of the matrix A.

5: D($*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the dimension of the array D must be at least $\max(1, N)$.

On entry: must contain the n diagonal elements of the matrix A.

6: DU($*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the dimension of the array DU must be at least $\max(1, N-1)$.

On entry: must contain the $(n-1)$ superdiagonal elements of the matrix A.

7: DLF($*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the dimension of the array DLF must be at least $\max(1, N-1)$.

On entry: must contain the $(n-1)$ multipliers that define the matrix L of the LU factorization of A.

8: DF($*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the dimension of the array DF must be at least $\max(1, N)$.

On entry: must contain the n diagonal elements of the upper triangular matrix U from the LU factorization of A.

9: DUF($*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the dimension of the array DUF must be at least $\max(1, N-1)$.

On entry: must contain the $(n-1)$ elements of the first superdiagonal of U.

10: DU2($*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the dimension of the array DU2 must be at least $\max(1, N-2)$.

On entry: must contain the $(n-2)$ elements of the second superdiagonal of U.

11: IPIV($*$) – INTEGER array *Input*

Note: the dimension of the array IPIV must be at least $\max(1, N)$.

On entry: must contain the n pivot indices that define the permutation matrix P. At the ith step, row i of the matrix was interchanged with row IPIV(i), and IPIV(i) must always be either i or $(i+1)$, IPIV$(i) = i$ indicating that a row interchange was not performed.

12: B(LDB,$*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the second dimension of the array B must be at least $\max(1, \text{NRHS})$.

On entry: the n by r matrix of right-hand sides B.

13: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07CVF (ZGTRFS) is called.

Constraint: LDB $\geq \max(1, N)$.

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

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

On entry: the n by r initial solution matrix X.

On exit: the n by r refined solution matrix X.

15: LDX – INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07CVF (ZGTRFS) is called.

Constraint: LDX $\geq \max(1, N)$.

16: FERR(NRHS) – REAL (KIND=nag_wp) array *Output*

On exit: estimate of the forward error bound for each computed solution vector, such that $\|\hat{x}_j - x_j\|_\infty / \|\hat{x}_j\|_\infty \leq$ FERR(j), where $\hat{x}_j$ is the jth column of the computed solution returned in the array X and x_j is the corresponding column of the exact solution X. The estimate is almost always a slight overestimate of the true error.

17: BERR(NRHS) – REAL (KIND=nag_wp) array *Output*

On exit: estimate of the component-wise relative backward error of each computed solution vector $\hat{x}_j$ (i.e., the smallest relative change in any element of A or B that makes $\hat{x}_j$ an exact solution).

18: WORK($2 \times N$) – COMPLEX (KIND=nag_wp) array *Workspace*

19: RWORK(N) – REAL (KIND=nag_wp) array *Workspace*

20: INFO – INTEGER *Output*

On exit: INFO $= 0$ unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the routine:

INFO < 0

If INFO $= -i$, the ith argument had an illegal value. An explanatory message is output, and execution of the program is terminated.

7 Accuracy

The computed solution for a single right-hand side, $\hat{x}$, satisfies an equation of the form

$$(A + E)\hat{x} = b,$$

where

$$\|E\|_\infty = O(\epsilon)\|A\|_\infty$$

and ϵ is the ***machine precision***. An approximate error bound for the computed solution is given by

$$\frac{\|\hat{x} - x\|_\infty}{\|x\|_\infty} \leq \kappa(A)\frac{\|E\|_\infty}{\|A\|_\infty},$$

where $\kappa(A) = \|A^{-1}\|_\infty \|A\|_\infty$, the condition number of A with respect to the solution of the linear equations. See Section 4.4 of Anderson *et al.* (1999) for further details.

Routine F07CUF (ZGTCON) can be used to estimate the condition number of A.

8 Further Comments

The total number of floating point operations required to solve the equations $AX = B$ or $A^{\mathrm{T}}X = B$ or $A^{\mathrm{H}}X = B$ is proportional to nr. At most five steps of iterative refinement are performed, but usually only one or two steps are required.

The real analogue of this routine is F07CHF (DGTRFS).

9 Example

This example solves the equations

$$AX = B,$$

where A is the tridiagonal matrix

$$A = \begin{pmatrix} -1.3 + 1.3i & 2.0 - 1.0i & 0 & 0 & 0 \\ 1.0 - 2.0i & -1.3 + 1.3i & 2.0 + 1.0i & 0 & 0 \\ 0 & 1.0 + 1.0i & -1.3 + 3.3i & -1.0 + 1.0i & 0 \\ 0 & 0 & 2.0 - 3.0i & -0.3 + 4.3i & 1.0 - 1.0i \\ 0 & 0 & 0 & 1.0 + 1.0i & -3.3 + 1.3i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 2.4 - 5.0i & 2.7 + 6.9i \\ 3.4 + 18.2i & -6.9 - 5.3i \\ -14.7 + 9.7i & -6.0 - 0.6i \\ 31.9 - 7.7i & -3.9 + 9.3i \\ -1.0 + 1.6i & -3.0 + 12.2i \end{pmatrix}.$$

Estimates for the backward errors and forward errors are also output.

9.1 Program Text

```
    Program f07cvfe

!     F07CVF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: nag_wp, x04dbf, zgtrfs, zgttrf, zgttrs
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, info, ldb, ldx, n, nrhs
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: b(:,:), d(:), df(:), dl(:),        &
                                          dlf(:), du(:), du2(:), duf(:),     &
                                          work(:), x(:,:)
      Real (Kind=nag_wp), Allocatable  :: berr(:), ferr(:), rwork(:)
      Integer, Allocatable             :: ipiv(:)
      Character (1)                    :: clabs(1), rlabs(1)
!     .. Executable Statements ..
      Write (nout,*) 'F07CVF Example Program Results'
      Write (nout,*)
      Flush (nout)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n, nrhs
      ldb = n
      ldx = n
      Allocate (b(ldb,nrhs),d(n),df(n),dl(n-1),dlf(n-1),du(n-1),du2(n-2), &
        duf(n-1),work(2*n),x(ldx,nrhs),berr(nrhs),ferr(nrhs),rwork(n),ipiv(n))

!     Read the tridiagonal matrix A from data file

      Read (nin,*) du(1:n-1)
      Read (nin,*) d(1:n)
      Read (nin,*) dl(1:n-1)

!     Read the right hand matrix B

      Read (nin,*)(b(i,1:nrhs),i=1,n)

!     Copy A into DUF, DF and DLF, and copy B into X

      duf(1:n-1) = du(1:n-1)
      df(1:n) = d(1:n)
      dlf(1:n-1) = dl(1:n-1)
      x(1:n,1:nrhs) = b(1:n,1:nrhs)

!     Factorize the copy of the tridiagonal matrix A
!     The NAG name equivalent of zgttrf is f07crf
      Call zgttrf(n,dlf,df,duf,du2,ipiv,info)

      If (info==0) Then

!       Solve the equations AX = B
!       The NAG name equivalent of zgttrs is f07csf
        Call zgttrs('No transpose',n,nrhs,dlf,df,duf,du2,ipiv,x,ldx,info)

!       Improve the solution and compute error estimates
!       The NAG name equivalent of zgtrfs is f07cvf
        Call zgtrfs('No transpose',n,nrhs,dl,d,du,dlf,df,duf,du2,ipiv,b,ldb,x, &
          ldx,ferr,berr,work,rwork,info)

!       Print the solution and the forward and backward error
!       estimates

!       ifail: behaviour on error exit
```

```
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call x04dbf('General',' ',n,nrhs,x,ldx,'Bracketed','F7.4', &
          'Solution(s)','Integer',rlabs,'Integer',clabs,80,0,ifail)

        Write (nout,*)
        Write (nout,*) 'Backward errors (machine-dependent)'
        Write (nout,99999) berr(1:nrhs)
        Write (nout,*)
        Write (nout,*) 'Estimated forward error bounds (machine-dependent)'
        Write (nout,99999) ferr(1:nrhs)
      Else
        Write (nout,99998) 'The (', info, ',', info, ')', &
          ' element of the factor U is zero'
      End If

99999 Format ((3X,1P,7E11.1))
99998 Format (1X,A,I3,A,I3,A,A)
    End Program f07cvfe
```

9.2 Program Data

```
F07CVF Example Program Data
   5    2                                                     :Values of N and NRHS
 (  2.0, -1.0) (  2.0,  1.0) ( -1.0,  1.0) (  1.0, -1.0) :End of DU
 ( -1.3,  1.3) ( -1.3,  1.3) ( -1.3,  3.3) ( -0.3,  4.3)
 ( -3.3,  1.3)                                            :End of D
 (  1.0, -2.0) (  1.0 , 1.0) (  2.0, -3.0) (  1.0,  1.0) :End of  DL
 (  2.4, -5.0) (  2.7,  6.9)
 (  3.4, 18.2) ( -6.9, -5.3)
 (-14.7,  9.7) ( -6.0, -0.6)
 ( 31.9, -7.7) ( -3.9,  9.3)
 ( -1.0,  1.6) ( -3.0, 12.2)                              :End of B
```

9.3 Program Results

```
 F07CVF Example Program Results

 Solution(s)
                    1                  2
 1  ( 1.0000, 1.0000) ( 2.0000,-1.0000)
 2  ( 3.0000,-1.0000) ( 1.0000, 2.0000)
 3  ( 4.0000, 5.0000) (-1.0000, 1.0000)
 4  (-1.0000,-2.0000) ( 2.0000, 1.0000)
 5  ( 1.0000,-1.0000) ( 2.0000,-2.0000)

 Backward errors (machine-dependent)
       3.7E-17    6.7E-17

 Estimated forward error bounds (machine-dependent)
       5.4E-14    7.3E-14
```