F08NUF (CUNMHR/ZUNMHR) – 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

F08NUF (CUNMHR/ZUNMHR) multiplies an arbitrary complex matrix C by the complex unitary matrix Q which was determined by F08NSF (CGEHRD/ZGEHRD) when reducing a complex general matrix to Hessenberg form.

2 Specification

```
      SUBROUTINE F08NUF(SIDE, TRANS, M, N, ILO, IHI, A, LDA, TAU, C,
     1                  LDC, WORK, LWORK, INFO)
      ENTRY      cunmhr(SIDE, TRANS, M, N, ILO, IHI, A, LDA, TAU, C,
     1                  LDC, WORK, LWORK, INFO)
      INTEGER           M, N, ILO, IHI, LDA, LDC, LWORK, INFO
      complex           A(LDA,*), TAU(*), C(LDC,*), WORK(LWORK)
      CHARACTER*1       SIDE, TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3 Description

This routine is intended to be used following a call to F08NSF(CGEHRD/ZGEHRD), which reduces a complex general matrix A to upper Hessenberg form H by a unitary similarity transformation: $A = QHQ^H$. F08NSF represents the matrix Q as a product of $i_{hi} - i_{lo}$ elementary reflectors. Here i_{lo} and i_{hi} are values determined by F08NVF (CGEBAL/ZGEBAL) when balancing the matrix; if the matrix has not been balanced, $i_{lo} = 1$ and $i_{hi} = n$.

This routine may be used to form one of the matrix products

$$QC,\ Q^HC,\ CQ \text{ or } CQ^H,$$

overwriting the result on C (which may be any complex rectangular matrix).

A common application of this routine is to transform a matrix V of eigenvectors of H to the matrix QV of eigenvectors of A.

4 References

[1] Golub G H and van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

5 Parameters

1: SIDE — CHARACTER*1 *Input*

On entry: indicates how Q or Q^H is to be applied to C as follows:

if SIDE = 'L', then Q or Q^H is applied to C from the left;
if SIDE = 'R', then Q or Q^H is applied to C from the right.

Constraint: SIDE = 'L' or 'R'.

2: TRANS — CHARACTER*1 *Input*

On entry: indicates whether Q or Q^H is to be applied to C as follows:

if TRANS = 'N', then Q is applied to C;
if TRANS = 'C', then Q^H is applied to C.

Constraint: TRANS = 'N' or 'C'.

3: M — INTEGER *Input*

On entry: m, the number of rows of the matrix C; m is also the order of Q if SIDE = 'L'.

Constraint: M ≥ 0.

4: N — INTEGER *Input*

On entry: n, the number of columns of the matrix C; n is also the order of Q if SIDE = 'R'.

Constraint: N ≥ 0.

5: ILO — INTEGER *Input*
6: IHI — INTEGER *Input*

On entry: these **must** be the same parameters ILO and IHI, respectively, as supplied to F08NSF (CGEHRD/ZGEHRD).

Constraints:

$1 \leq$ ILO $\leq$ IHI $\leq$ M if SIDE = 'L' and M > 0;
ILO = 1 and IHI = 0 if SIDE = 'L' and M = 0;
$1 \leq$ ILO $\leq$ IHI $\leq$ N if SIDE = 'R' and N > 0;
ILO = 1 and IHI = 0 if SIDE = 'R' and N = 0.

7: A(LDA,$*$) — ***complex*** array *Input*

Note: the second dimension of the array A must be at least max(1,M) if SIDE = 'L' and at least max(1,N) if SIDE = 'R'.

On entry: details of the vectors which define the elementary reflectors, as returned by F08NSF (CGEHRD/ZGEHRD).

8: LDA — INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08NUF (CUNMHR/ZUNMHR) is called.

Constraints:

LDA $\geq$ max(1,M) if SIDE = 'L',
LDA $\geq$ max(1,N) if SIDE = 'R'.

9: TAU($*$) — ***complex*** array *Input*

Note: the dimension of the array TAU must be at least max(1,M$-$1) if SIDE = 'L' and at least max(1,N$-$1) if SIDE = 'R'.

On entry: further details of the elementary reflectors, as returned by F08NSF (CGEHRD/ZGEHRD).

10: C(LDC,$*$) — ***complex*** array *Input/Output*

Note: the second dimension of the array C must be at least max(1,N).

On entry: the m by n matrix C.

On exit: C is overwritten by QC or Q^HC or CQ^H or CQ as specified by SIDE and TRANS.

11: LDC — INTEGER *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which F08NUF (CUNMHR/ZUNMHR) is called.

Constraint: LDC $\geq$ max(1,M).

12: WORK(LWORK) — ***complex*** array *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

13: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08NUF (CUNMHR/ZUNMHR) is called.

Suggested value: for optimum performance LWORK should be at least N $\times$ *nb* if SIDE = 'L' and at least M $\times$ *nb* if SIDE = 'R', where *nb* is the ***blocksize***.

Constraints:

LWORK $\geq$ max(1,N) if SIDE = 'L',
LWORK $\geq$ max(1,M) if SIDE = 'R'.

14: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

INFO < 0

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

7 Accuracy

The computed result differs from the exact result by a matrix E such that

$$\| E \|_2 = O(\epsilon) \| C \|_2,$$

where ϵ is the ***machine precision***.

8 Further Comments

The total number of real floating-point operations is approximately $8nq^2$ if SIDE = 'L' and $8mq^2$ if SIDE = 'R', where $q = i_{hi} - i_{lo}$.

The real analogue of this routine is F08NGF (SORMHR/DORMHR).

9 Example

To compute all the eigenvalues of the matrix A, where

$$A = \begin{pmatrix} -3.97 - 5.04i & -4.11 + 3.70i & -0.34 + 1.01i & 1.29 - 0.86i \\ 0.34 - 1.50i & 1.52 - 0.43i & 1.88 - 5.38i & 3.36 + 0.65i \\ 3.31 - 3.85i & 2.50 + 3.45i & 0.88 - 1.08i & 0.64 - 1.48i \\ -1.10 + 0.82i & 1.81 - 1.59i & 3.25 + 1.33i & 1.57 - 3.44i \end{pmatrix},$$

and those eigenvectors which correspond to eigenvalues λ such that $\mathrm{Re}(\lambda) < 0$. Here A is general and must first be reduced to upper Hessenberg form H by F08NSF (CGEHRD/ZGEHRD). The program then calls F08PSF (CHSEQR/ZHSEQR) to compute the eigenvalues, and F08PXF (CHSEIN/ZHSEIN) to compute the required eigenvectors of H by inverse iteration. Finally F08NUF (CUNMHR/ZUNMHR) is called to transform the eigenvectors of H back to eigenvectors of the original matrix A.

9.1 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.

```
*     F08NUF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LDH, LDZ, LWORK, LDVL, LDVR
      PARAMETER        (NMAX=8,LDA=NMAX,LDH=NMAX,LDZ=1,LWORK=64*NMAX,
     +                 LDVL=NMAX,LDVR=NMAX)
*     .. Local Scalars ..
      real             THRESH
      INTEGER          I, IFAIL, INFO, J, M, N
*     .. Local Arrays ..
      complex          A(LDA,NMAX), H(LDH,NMAX), TAU(NMAX),
     +                 VL(LDVL,NMAX), VR(LDVR,NMAX), W(NMAX),
     +                 WORK(LWORK), Z(LDZ,1)
      real             RWORK(NMAX)
      INTEGER          IFAILL(NMAX), IFAILR(NMAX)
      LOGICAL          SELECT(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         F06TFF, X04DBF, cgehrd, chsein, chseqr, cunmhr
*     .. Intrinsic Functions ..
      INTRINSIC        real, imag
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08NUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,N)
*
         READ (NIN,*) THRESH
*
*        Reduce A to upper Hessenberg form H = (Q**H)*A*Q
*
         CALL cgehrd(N,1,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Copy A to H
*
         CALL F06TFF('General',N,N,A,LDA,H,LDH)
*
*        Calculate the eigenvalues of H (same as A)
*
         CALL chseqr('Eigenvalues','No vectors',N,1,N,H,LDH,W,Z,LDZ,
     +               WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
            WRITE (NOUT,*) 'Eigenvalues'
```

```
            WRITE (NOUT,99999) (' (',real(W(I)),',',imag(W(I)),')',I=1,
     +        N)
*
            DO 20 I = 1, N
               SELECT(I) = real(W(I)) .LT. THRESH
   20       CONTINUE
*
*           Calculate the eigenvectors of H (as specified by SELECT),
*           storing the result in VR
*
            CALL chsein('Right','QR','No initial vectors',SELECT,N,A,
     +                  LDA,W,VL,LDVL,VR,LDVR,N,M,WORK,RWORK,IFAILL,
     +                  IFAILR,INFO)
*
*           Calculate the eigenvectors of A = Q * (eigenvectors of H)
*
            CALL cunmhr('Left','No transpose',N,M,1,N,A,LDA,TAU,VR,LDVR,
     +                  WORK,LWORK,INFO)
*
*           Print eigenvectors
*
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04DBF('General',' ',N,M,VR,LDVR,'Bracketed','F7.4',
     +                  'Contents of array VR','Integer',RLABS,
     +                  'Integer',CLABS,80,0,IFAIL)
*
         END IF
      END IF
      STOP
*
99999 FORMAT ((3X,4(A,F7.4,A,F7.4,A,:)))
      END
```

9.2 Program Data

```
F08NUF Example Program Data
  4                                                            :Value of N
 (-3.97,-5.04) (-4.11, 3.70) (-0.34, 1.01) ( 1.29,-0.86)
 ( 0.34,-1.50) ( 1.52,-0.43) ( 1.88,-5.38) ( 3.36, 0.65)
 ( 3.31,-3.85) ( 2.50, 3.45) ( 0.88,-1.08) ( 0.64,-1.48)
 (-1.10, 0.82) ( 1.81,-1.59) ( 3.25, 1.33) ( 1.57,-3.44)   :End of matrix A
  0.0                                                          :Value of THRESH
```

9.3 Program Results

```
F08NUF Example Program Results

Eigenvalues
   (-6.0004,-6.9998) (-5.0000, 2.0060) ( 7.9982,-0.9964) ( 3.0023,-3.9998)

Contents of array VR
                  1                  2
1  ( 1.0000, 0.0000) ( 0.2613, 0.5284)
2  (-0.0210, 0.3590) ( 0.6485, 0.4683)
3  ( 0.1035, 0.3683) (-0.0323,-0.8516)
4  (-0.0664,-0.3436) (-0.4521, 0.1368)
```