# NAG Library Routine Document

# F08NUF (ZUNMHR)

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

## 2 Specification

```
SUBROUTINE F08NUF (SIDE, TRANS, M, N, ILO, IHI, A, LDA, TAU, C, LDC,       &
                   WORK, LWORK, INFO)

INTEGER                 M, N, ILO, IHI, LDA, LDC, LWORK, INFO
COMPLEX (KIND=nag_wp) A(LDA,*), TAU(*), C(LDC,*), WORK(max(1,LWORK))
CHARACTER(1)            SIDE, TRANS
```

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

## 3 Description

F08NUF (ZUNMHR) is intended to be used following a call to F08NSF (ZGEHRD), which reduces a complex general matrix $A$ to upper Hessenberg form $H$ by a unitary similarity transformation: $A = QHQ^{\mathrm{H}}$. F08NSF (ZGEHRD) represents the matrix $Q$ as a product of $i_{\mathrm{hi}} - i_{\mathrm{lo}}$ elementary reflectors. Here $i_{\mathrm{lo}}$ and $i_{\mathrm{hi}}$ are values determined by F08NVF (ZGEBAL) when balancing the matrix; if the matrix has not been balanced, $i_{\mathrm{lo}} = 1$ and $i_{\mathrm{hi}} = n$.

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

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

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

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

## 5 Parameters

1: SIDE – CHARACTER(1) *Input*

*On entry*: indicates how $Q$ or $Q^{\mathrm{H}}$ is to be applied to $C$.

SIDE = 'L'
: $Q$ or $Q^{\mathrm{H}}$ is applied to $C$ from the left.

SIDE = 'R'
: $Q$ or $Q^{\mathrm{H}}$ 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 $C$.

TRANS = 'N'
: $Q$ is applied to $C$.

TRANS = 'C'
: $Q^{H}$ 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*: $\mathrm{M} \geq 0$.

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*: $\mathrm{N} \geq 0$.

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

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

*Constraints*:

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

7: $\mathrm{A}(\mathrm{LDA},*)$ – COMPLEX (KIND=nag_wp) array *Input*

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

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

8: LDA – INTEGER *Input*

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

*Constraints*:

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

9: $\mathrm{TAU}(*)$ – COMPLEX (KIND=nag_wp) array *Input*

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

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

10: $\mathrm{C}(\mathrm{LDC},*)$ – COMPLEX (KIND=nag_wp) array *Input/Output*

**Note**: the second dimension of the array C must be at least $\max(1,\mathrm{N})$.

*On entry*: the $m$ by $n$ matrix $C$.

*On exit*: C is overwritten by $QC$ or $Q^{\mathrm{H}}C$ or $CQ$ or $CQ^{\mathrm{H}}$ 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 (ZUNMHR) is called.

*Constraint*: $\mathrm{LDC} \geq \max(1, \mathrm{M})$.

12: WORK$(\max(1, \mathrm{LWORK}))$ – COMPLEX (KIND=nag_wp) array *Workspace*

*On exit*: if $\mathrm{INFO} = 0$, the real part of WORK$(1)$ contains the minimum value of LWORK required for optimal performance.

13: LWORK – INTEGER *Input*

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

If $\mathrm{LWORK} = -1$, a workspace query is assumed; the routine only calculates the optimal size of the WORK array, returns this value as the first entry of the WORK array, and no error message related to LWORK is issued.

*Suggested value*: for optimal performance, $\mathrm{LWORK} \geq \mathrm{N} \times nb$ if SIDE = 'L' and at least $\mathrm{M} \times nb$ if SIDE = 'R', where $nb$ is the optimal ***block size***.

*Constraints*:

if SIDE = 'L', $\mathrm{LWORK} \geq \max(1, \mathrm{N})$ or $\mathrm{LWORK} = -1$;
if SIDE = 'R', $\mathrm{LWORK} \geq \max(1, \mathrm{M})$ or $\mathrm{LWORK} = -1$.

14: INFO – INTEGER *Output*

*On exit*: $\mathrm{INFO} = 0$ unless the routine detects an error (see Section 6).

# 6 Error Indicators and Warnings

$\mathrm{INFO} < 0$

If $\mathrm{INFO} = -i$, argument $i$ 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 $\epsilon$ is the ***machine precision***.

# 8 Parallelism and Performance

F08NUF (ZUNMHR) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F08NUF (ZUNMHR) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

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

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

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

# 10 Example

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

## 10.1 Program Text

```
    Program f08nufe

!     F08NUF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: nag_wp, x04dbf, zgehrd, zhsein, zhseqr, zunmhr
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: thresh
      Integer                          :: i, ifail, info, lda, ldc, ldh, ldvl, &
                                          ldz, lwork, m, n
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), c(:,:), h(:,:), tau(:),    &
                                            vl(:,:), w(:), work(:), z(:,:)
      Real (Kind=nag_wp), Allocatable  :: rwork(:)
      Integer, Allocatable             :: ifaill(:), ifailr(:)
      Logical, Allocatable             :: select(:)
      Character (1)                    :: clabs(1), rlabs(1)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: aimag, real
!     .. Executable Statements ..
      Write (nout,*) 'F08NUF Example Program Results'
      Flush (nout)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n
      ldz = 1
      lda = n
      ldc = n
      ldh = n
      ldvl = n
      lwork = 64*n
      Allocate (a(lda,n),c(ldc,n),h(ldh,n),tau(n),vl(ldvl,n),w(n),work(lwork), &
        z(ldz,1),rwork(n),ifaill(n),ifailr(n),select(n))

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

```
      Read (nin,*) thresh

!     Reduce A to upper Hessenberg form H = (Q**H)*A*Q
!     The NAG name equivalent of zgehrd is f08nsf
      Call zgehrd(n,1,n,a,lda,tau,work,lwork,info)

!     Copy A to H
      h(1:n,1:n) = a(1:n,1:n)

!     Calculate the eigenvalues of H (same as A)
!     The NAG name equivalent of zhseqr is f08psf
      Call zhseqr('Eigenvalues','No vectors',n,1,n,h,ldh,w,z,ldz,work,lwork, &
        info)

      Write (nout,*)
      If (info>0) Then
        Write (nout,*) 'Failure to converge.'
      Else
        Write (nout,*) 'Eigenvalues'
        Write (nout,99999)(' (',real(w(i)),',',aimag(w(i)),')',i=1,n)
        Flush (nout)

        Do i = 1, n
          select(i) = real(w(i)) < thresh
        End Do

!       Calculate the eigenvectors of H (as specified by SELECT),
!       storing the result in C
!       The NAG name equivalent of zhsein is f08pxf
        Call zhsein('Right','QR','No initial vectors',select,n,a,lda,w,vl, &
          ldvl,c,ldc,n,m,work,rwork,ifaill,ifailr,info)

!       Calculate the eigenvectors of A = Q * (eigenvectors of H)
!       The NAG name equivalent of zunmhr is f08nuf
        Call zunmhr('Left','No transpose',n,m,1,n,a,lda,tau,c,ldc,work,lwork, &
          info)

!       Print eigenvectors

        Write (nout,*)
        Flush (nout)

!       Normalize the eigenvectors
        Do i = 1, m
          c(1:n,i) = c(1:n,i)/c(1,i)
        End Do

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call x04dbf('General',' ',n,m,c,ldc,'Bracketed','F7.4', &
          'Contents of array C','Integer',rlabs,'Integer',clabs,80,0,ifail)

      End If

99999 Format ((3X,4(A,F7.4,A,F7.4,A:)))
    End Program f08nufe
```

## 10.2 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
```

### 10.3 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 C
                     1                  2
 1  ( 1.0000, 0.0000) ( 1.0000, 0.0000)
 2  (-0.0210, 0.3590) ( 1.1997,-0.6339)
 3  ( 0.1035, 0.3683) (-1.3192,-0.5912)
 4  (-0.0664,-0.3436) (-0.1319, 0.7904)
```