NAG Toolbox

nag_lapack_zunmhr (f08nu)

1 Purpose

nag_lapack_zunmhr (f08nu) multiplies an arbitrary complex matrix C by the complex unitary matrix Q which was determined by nag_lapack_zgehrd (f08ns) when reducing a complex general matrix to Hessenberg form.

2 Syntax

```
[c, info] = nag_lapack_zunmhr(side, trans, ilo, ihi, a, tau, c, 'm', m, 'n', n)
[c, info] = f08nu(side, trans, ilo, ihi, a, tau, c, 'm', m, 'n', n)
```

3 Description

nag_lapack_zunmhr (f08nu) is intended to be used following a call to nag_lapack_zgehrd (f08ns), which reduces a complex general matrix A to upper Hessenberg form H by a unitary similarity transformation: $A = QHQ^{\mathrm{H}}$. nag_lapack_zgehrd (f08ns) represents the matrix Q as a product of $i_{\mathrm{hi}} - i_{\mathrm{lo}}$ elementary reflectors. Here i_{lo} and i_{hi} are values determined by nag_lapack_zgebal (f08nv) when balancing the matrix; if the matrix has not been balanced, $i_{\mathrm{lo}} = 1$ and $i_{\mathrm{hi}} = n$.

This function 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 function 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

5.1 Compulsory Input Parameters

1: **side** – CHARACTER(1)

Indicates how Q or Q^{H} is to be applied to C.

side = 'L'
Q or Q^{H} is applied to C from the left.

side = 'R'
Q or Q^{H} is applied to C from the right.

Constraint: **side** = 'L' or 'R'.

2: **trans** – CHARACTER(1)

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: **ilo** – INTEGER
4: **ihi** – INTEGER

These **must** be the same arguments **ilo** and **ihi**, respectively, as supplied to nag_lapack_zgehrd (f08ns).

Constraints:

if **side** = 'L' and $\mathbf{m} > 0$, $1 \leq \mathbf{ilo} \leq \mathbf{ihi} \leq \mathbf{m}$;
if **side** = 'L' and $\mathbf{m} = 0$, $\mathbf{ilo} = 1$ and $\mathbf{ihi} = 0$;
if **side** = 'R' and $\mathbf{n} > 0$, $1 \leq \mathbf{ilo} \leq \mathbf{ihi} \leq \mathbf{n}$;
if **side** = 'R' and $\mathbf{n} = 0$, $\mathbf{ilo} = 1$ and $\mathbf{ihi} = 0$.

5: $\mathbf{a}(lda,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension, lda, of the array **a** must satisfy

if **side** = 'L', $lda \geq \max(1,\mathbf{m})$;
if **side** = 'R', $lda \geq \max(1,\mathbf{n})$.

The second dimension of the array **a** must be at least $\max(1,\mathbf{m})$ if **side** = 'L' and at least $\max(1,\mathbf{n})$ if **side** = 'R'.

Details of the vectors which define the elementary reflectors, as returned by nag_lapack_zgehrd (f08ns).

6: $\mathbf{tau}(:)$ – COMPLEX (KIND=nag_wp) array

The dimension of the array **tau** must be at least $\max(1,\mathbf{m}-1)$ if **side** = 'L' and at least $\max(1,\mathbf{n}-1)$ if **side** = 'R'

Further details of the elementary reflectors, as returned by nag_lapack_zgehrd (f08ns).

7: $\mathbf{c}(ldc,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **c** must be at least $\max(1,\mathbf{m})$.

The second dimension of the array **c** must be at least $\max(1,\mathbf{n})$.

The m by n matrix C.

5.2 Optional Input Parameters

1: **m** – INTEGER

Default: the first dimension of the array **c**.

m, the number of rows of the matrix C; m is also the order of Q if **side** = 'L'.

Constraint: $\mathbf{m} \geq 0$.

2: **n** – INTEGER

Default: the second dimension of the array **c**.

n, the number of columns of the matrix C; n is also the order of Q if **side** = 'R'.

Constraint: $\mathbf{n} \geq 0$.

5.3 Output Parameters

1: **c**(ldc, :) – COMPLEX (KIND=nag_wp) array

The first dimension of the array **c** will be $\max(1, \mathbf{m})$.

The second dimension of the array **c** will be $\max(1, \mathbf{n})$.

c stores QC or $Q^{\mathrm{H}}C$ or CQ or CQ^{H} as specified by **side** and **trans**.

2: **info** – INTEGER

info $= 0$ unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

info $= -i$

If **info** $= -i$, parameter i had an illegal value on entry. The parameters are numbered as follows:

1: **side**, 2: **trans**, 3: **m**, 4: **n**, 5: **ilo**, 6: **ihi**, 7: **a**, 8: **lda**, 9: **tau**, 10: **c**, 11: **ldc**, 12: **work**, 13: **lwork**, 14: **info**.

It is possible that **info** refers to a parameter that is omitted from the MATLAB interface. This usually indicates that an error in one of the other input parameters has caused an incorrect value to be inferred.

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 $q = i_{\mathrm{hi}} - i_{\mathrm{lo}}$.

The real analogue of this function is nag_lapack_dormhr (f08ng).

9 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 λ such that $\mathrm{Re}(\lambda) < 0$. Here A is general and must first be reduced to upper Hessenberg form H by nag_lapack_zgehrd (f08ns). The program then calls nag_lapack_zhseqr (f08ps) to compute the eigenvalues, and nag_lapack_zhsein (f08px) to compute the required eigenvectors of H by inverse iteration. Finally nag_lapack_zunmhr (f08nu) is called to transform the eigenvectors of H back to eigenvectors of the original matrix A.

9.1 Program Text

```
function f08nu_example

fprintf('f08nu example results\n\n');

n   = nag_int(4);
ilo = nag_int(1);
ihi = n;
a = [ -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];

% Reduce A to upper Hessenberg Form
[H, tau, info] = f08ns(ilo, ihi, a);

% Form Q
[Q, info] = f08nt(ilo, ihi, H, tau);

% Schur factorize H = Y*T*Y^H
job   = 'Schur form';
compz = 'Vectors';
[~, w, ~, info] = f08ps( ...
                         job, compz, ilo, ihi, H, Q);

disp('Eigenvalues of A');
disp(w);

% Calculate eigenvectors of H for negative real part eigenvalues
select = (real(w) < 0);

job = 'Right';
eigsrc = 'QR';
initv = 'No initial vectors';
vl = [];
vr = complex(zeros(n,n));
[~, ~, VR, m, ifaill, ifailr, info] = ...
    f08px(...
          job, eigsrc, initv, select, H, w, vl, vr, n);

% Eigenvectors of A = Q*VR
side = 'Left';
trans = 'No transpose';
[Z, info] = f08nu(side, trans, ilo, ihi, H, tau, VR);

% Normalize Z: largest elements are real
for i = 1:m
  [~,k] = max(abs(real(Z(:,i)))+abs(imag(Z(:,i))));
  Z(:,i) = Z(:,i)*conj(Z(k,i))/abs(Z(k,i));
end
disp('Eigenvectors corresponding to eigenvalues with negative real part');
disp(Z);
```

9.2 Program Results

```
f08nu example results

Eigenvalues of A
  -6.0004 - 6.9998i
  -5.0000 + 2.0060i
   7.9982 - 0.9964i
   3.0023 - 3.9998i
```

```
Eigenvectors corresponding to eigenvalues with negative real part
   0.8079 + 0.0000i  -0.4076 + 0.1827i
  -0.0169 + 0.2900i  -0.3732 + 0.4776i
   0.0836 + 0.2975i   0.6457 + 0.0000i
  -0.0536 - 0.2776i  -0.0906 - 0.3463i
```