NAG Toolbox

nag_lapack_dormhr (f08ng)

1 Purpose

nag_lapack_dormhr (f08ng) multiplies an arbitrary real matrix C by the real orthogonal matrix Q which was determined by nag_lapack_dgehrd (f08ne) when reducing a real general matrix to Hessenberg form.

2 Syntax

```
[c, info] = nag_lapack_dormhr(side, trans, ilo, ihi, a, tau, c, 'm', m, 'n', n)
[c, info] = f08ng(side, trans, ilo, ihi, a, tau, c, 'm', m, 'n', n)
```

3 Description

nag_lapack_dormhr (f08ng) is intended to be used following a call to nag_lapack_dgehrd (f08ne), which reduces a real general matrix A to upper Hessenberg form H by an orthogonal similarity transformation: $A = QHQ^{\mathrm{T}}$. nag_lapack_dgehrd (f08ne) 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_dgebal (f08nh) 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

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

overwriting the result on C (which may be any real 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^{T} is to be applied to C.

side = 'L'
Q or Q^{T} is applied to C from the left.

side = 'R'
Q or Q^{T} is applied to C from the right.

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

2: **trans** – CHARACTER(1)

Indicates whether Q or Q^{T} is to be applied to C.

trans = 'N'
Q is applied to C.

trans = 'T'
Q^{T} is applied to C.

Constraint: **trans** = 'N' or 'T'.

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

These **must** be the same arguments **ilo** and **ihi**, respectively, as supplied to nag_lapack_dgehrd (f08ne).

Constraints:

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

5: **a**$(lda, :)$ – REAL (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_dgehrd (f08ne).

6: **tau**$(:)$ – REAL (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_dgehrd (f08ne).

7: **c**$(ldc, :)$ – REAL (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: **m** ≥ 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: **n** ≥ 0.

5.3 Output Parameters

1: **c**$(ldc,:)$ – REAL (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 $Q^{\mathrm{T}}C$ or CQ or CQ^{T} 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 floating-point operations is approximately $2nq^2$ if **side** = 'L' and $2mq^2$ if **side** = 'R', where $q = i_{\mathrm{hi}} - i_{\mathrm{lo}}$.

The complex analogue of this function is nag_lapack_zunmhr (f08nu).

9 Example

This example computes all the eigenvalues of the matrix A, where

$$A = \begin{pmatrix} 0.35 & 0.45 & -0.14 & -0.17 \\ 0.09 & 0.07 & -0.54 & 0.35 \\ -0.44 & -0.33 & -0.03 & 0.17 \\ 0.25 & -0.32 & -0.13 & 0.11 \end{pmatrix},$$

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_dgehrd (f08ne). The program then calls nag_lapack_dhseqr (f08pe) to compute the eigenvalues, and nag_lapack_dhsein (f08pk) to compute the required eigenvectors of H by inverse iteration. Finally nag_lapack_dormhr (f08ng) is called to transform the eigenvectors of H back to eigenvectors of the original matrix A.

9.1 Program Text

```
     function f08ng_example

fprintf('f08ng example results\n\n');

n   = nag_int(4);
ilo = nag_int(1);
ihi = n;
a = [ 0.35,  0.45, -0.14, -0.17;
      0.09,  0.07, -0.54,  0.35;
     -0.44, -0.33, -0.03,  0.17;
      0.25, -0.32, -0.13,  0.11];

% Reduce A to upper Hessenberg Form A = QHQ^T
[H, tau, info] = f08ne( ...
                        ilo, ihi, a);

% Form Q
[Q, info] = f08nf( ...
                   ilo, ihi, H, tau);

% Schur factorize H = Y*T*Y^T
job   = 'Schur form';
compz = 'No Vectors';
[~, wr, wi, ~, info] = f08pe( ...
                              job, compz, ilo, ihi, H, Q);

w = wr + i*wi;
disp('Eigenvalues of A');
disp(w);

% Calculate eigenvetors of H corresponding to negative real part eigenvalues
select = (wr < 0);

job = 'Right';
eigsrc = 'QR';
initv = 'No initial vectors';
vl = [];
vr = zeros(n,n);
[~, ~, ~, VR, m, ifaill, ifailr, info] = ...
f08pk( ...
       job, eigsrc, initv, select, H, wr, wi, vl, vr, n);

% Eigenvectors of A = Q*VR
side = 'Left';
trans = 'No transpose';
[V, info] = f08ng( ...
                   side, trans, ilo, ihi, H, tau, VR);

% Combine columns of V into complex eigenvectors Z
j = 0;
for k = 1:n
  if select(k)
    j = j+1;
    if (wi(k)==0)
      % Normalize eigenvector: largest element positive
      [~,l] = max(abs(V(:,j)));
      if V(l,j) < 0;
        V(:,j) = -V(:,j);
      end
      Z(:,j) = complex(V(:,j));
    else
      Z(:,j)   = V(:,j) + i*V(:,j+1);
      Z(:,j+1) = V(:,j) - i*V(:,j+1);
      % Normalize: largest element is real
      [~,l] = max(abs(real(Z(:,j)))+abs(imag(Z(:,j))));
      Z(:,j) = Z(:,j)*conj(Z(l,j))/abs(Z(l,j));
      Z(:,j+1) = Z(:,j+1)*conj(Z(l,j+1))/abs(Z(l,j+1));
      j = j+1;
      select(k+1) = false;
```

```
    end
  end
end
disp('Eigenvectors corresponding to eigenvalues with negative real part');
disp(Z);
```

9.2 Program Results

```
     f08ng example results

Eigenvalues of A
   0.7995 + 0.0000i
  -0.0994 + 0.4008i
  -0.0994 - 0.4008i
  -0.1007 + 0.0000i

Eigenvectors corresponding to eigenvalues with negative real part
  -0.2379 + 0.3134i  -0.2379 - 0.3134i   0.1493 + 0.0000i
   0.3100 - 0.6430i   0.3100 + 0.6430i   0.3956 + 0.0000i
   0.1196 - 0.3795i   0.1196 + 0.3795i   0.7075 + 0.0000i
   0.8319 + 0.0000i   0.8319 + 0.0000i   0.8603 + 0.0000i
```