NAG Toolbox

nag_lapack_zunmbr (f08ku)

1 Purpose

nag_lapack_zunmbr (f08ku) multiplies an arbitrary complex m by n matrix C by one of the complex unitary matrices Q or P which were determined by nag_lapack_zgebrd (f08ks) when reducing a complex matrix to bidiagonal form.

2 Syntax

```
[c, info] = nag_lapack_zunmbr(vect, side, trans, k, a, tau, c, 'm', m, 'n', n)
[c, info] = f08ku(vect, side, trans, k, a, tau, c, 'm', m, 'n', n)
```

3 Description

nag_lapack_zunmbr (f08ku) is intended to be used after a call to nag_lapack_zgebrd (f08ks), which reduces a complex rectangular matrix A to real bidiagonal form B by a unitary transformation: $A = QBP^{\mathrm{H}}$. nag_lapack_zgebrd (f08ks) represents the matrices Q and P^{H} as products of elementary reflectors.

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

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

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

4 References

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

5 Parameters

Note: in the descriptions below, r denotes the order of Q or P^{H}: if **side** $=$ 'L', $r = \mathbf{m}$ and if **side** $=$ 'R', $r = \mathbf{n}$.

5.1 Compulsory Input Parameters

1: **vect** – CHARACTER(1)

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

vect $=$ 'Q'
Q or Q^{H} is applied to C.

vect $=$ 'P'
P or P^{H} is applied to C.

Constraint: **vect** $=$ 'Q' or 'P'.

2: **side** – CHARACTER(1)

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

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

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

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

3: **trans** – CHARACTER(1)

Indicates whether Q or P or Q^H or P^H is to be applied to C.

trans = 'N'
Q or P is applied to C.

trans = 'C'
Q^H or P^H is applied to C.

Constraint: **trans** = 'N' or 'C'.

4: **k** – INTEGER

If **vect** = 'Q', the number of columns in the original matrix A.

If **vect** = 'P', the number of rows in the original matrix A.

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

5: **a**$(lda,:)$ – COMPLEX (KIND=nag_wp) array

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

if **vect** = 'Q', $lda \geq \max(1, r)$;
if **vect** = 'P', $lda \geq \max(1, \min(r, \mathbf{k}))$.

The second dimension of the array **a** must be at least $\max(1, \min(r, \mathbf{k}))$ if **vect** = 'Q' and at least $\max(1, r)$ if **vect** = 'P'.

Details of the vectors which define the elementary reflectors, as returned by nag_lapack_zgebrd (f08ks).

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

The dimension of the array **tau** must be at least $\max(1, \min(r, \mathbf{k}))$

Further details of the elementary reflectors, as returned by nag_lapack_zgebrd (f08ks) in its argument **tauq** if **vect** = 'Q', or in its argument **taup** if **vect** = 'P'.

7: **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 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.

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.

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

5.3 Output Parameters

1: $\mathbf{c}(ldc,:)$ – 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 $Q^{H}C$ or CQ or $C^{H}Q$ or PC or $P^{H}C$ or CP or $C^{H}P$ as specified by **vect**, **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: **vect**, 2: **side**, 3: **trans**, 4: **m**, 5: **n**, 6: **k**, 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

if **side** = 'L' and $m \geq k$, $8nk(2m-k)$;

if **side** = 'R' and $n \geq k$, $8mk(2n-k)$;

if **side** = 'L' and $m < k$, $8m^2n$;

if **side** = 'R' and $n < k$, $8mn^2$,

where k is the value of the argument **k**.

The real analogue of this function is nag_lapack_dormbr (f08kg).

9 Example

For this function two examples are presented. Both illustrate how the reduction to bidiagonal form of a matrix A may be preceded by a QR or LQ factorization of A.

In the first example, $m > n$, and

$$A = \begin{pmatrix} 0.96 - 0.81i & -0.03 + 0.96i & -0.91 + 2.06i & -0.05 + 0.41i \\ -0.98 + 1.98i & -1.20 + 0.19i & -0.66 + 0.42i & -0.81 + 0.56i \\ 0.62 - 0.46i & 1.01 + 0.02i & 0.63 - 0.17i & -1.11 + 0.60i \\ -0.37 + 0.38i & 0.19 - 0.54i & -0.98 - 0.36i & 0.22 - 0.20i \\ 0.83 + 0.51i & 0.20 + 0.01i & -0.17 - 0.46i & 1.47 + 1.59i \\ 1.08 - 0.28i & 0.20 - 0.12i & -0.07 + 1.23i & 0.26 + 0.26i \end{pmatrix}.$$

The function first performs a QR factorization of A as $A = Q_aR$ and then reduces the factor R to bidiagonal form B: $R = Q_bBP^{\mathrm{H}}$. Finally it forms Q_a and calls nag_lapack_zunmbr (f08ku) to form $Q = Q_aQ_b$.

In the second example, $m < n$, and

$$A = \begin{pmatrix} 0.28 - 0.36i & 0.50 - 0.86i & -0.77 - 0.48i & 1.58 + 0.66i \\ -0.50 - 1.10i & -1.21 + 0.76i & -0.32 - 0.24i & -0.27 - 1.15i \\ 0.36 - 0.51i & -0.07 + 1.33i & -0.75 + 0.47i & -0.08 + 1.01i \end{pmatrix}.$$

The function first performs an LQ factorization of A as $A = LP_a^{\mathrm{H}}$ and then reduces the factor L to bidiagonal form B: $L = QBP_b^{\mathrm{H}}$. Finally it forms P_b^{H} and calls nag_lapack_zunmbr (f08ku) to form $P^{\mathrm{H}} = P_b^{\mathrm{H}}P_a^{\mathrm{H}}$.

9.1 Program Text

```
     function f08ku_example

fprintf('f08ku example results\n\n');

% Two cases of preceding reduction to bidiagonal form by QR or LQ
% Case 1: m > n, precede by QR
ex1;
% Case 2: m < n, precede by LQ
ex2;

function ex1
  m = nag_int(6);
  n = nag_int(4);
  a = [ 0.96 - 0.81i, -0.03 + 0.96i, -0.91 + 2.06i, -0.05 + 0.41i;
       -0.98 + 1.98i, -1.20 + 0.19i, -0.66 + 0.42i, -0.81 + 0.56i;
        0.62 - 0.46i,  1.01 + 0.02i,  0.63 - 0.17i, -1.11 + 0.60i;
       -0.37 + 0.38i,  0.19 - 0.54i, -0.98 - 0.36i,  0.22 - 0.20i;
        0.83 + 0.51i,  0.20 + 0.01i, -0.17 - 0.46i,  1.47 + 1.59i;
        1.08 - 0.28i,  0.20 - 0.12i, -0.07 + 1.23i,  0.26 + 0.26i];

  % Factorize A = QR
  [QR, tau, info] = f08as(a);

  % Generate Q from QR
  [Q, info] = f08at(QR, tau);

  % Extract R from QR
  R = triu(QR(1:n,1:n));

  % Bidiagonalize R = Q1 B P^H
  [B, d, e, tauq, taup, info] = ...
  f08ks(R);

  % Update Q: Q2 = Q*Q1 (so A = QR = Q2 B P^H)
  vect = 'Q';
  side = 'Right';
  trans = 'No transpose';
  [Q2, info] = f08ku( ...
                      vect, side, trans, n, B, tauq, Q);

  fprintf('Example 1: bidiagonal matrix B\n   Main diagonal  ');
  fprintf(' %7.3f',d);
  fprintf('\n   super-diagonal ');
```

```
  fprintf(' %7.3f',e);
  fprintf('\n\n');
  disp('Example 1: Orthogonal matrix Q');
  disp(Q2);

function ex2
  m = nag_int(3);
  n = nag_int(4);
  a = [0.28 - 0.36i  0.50 - 0.86i -0.77 - 0.48i  1.58 + 0.66i;
      -0.50 - 1.10i -1.21 + 0.76i -0.32 - 0.24i -0.27 - 1.15i;
       0.36 - 0.51i -0.07 + 1.33i -0.75 + 0.47i -0.08 + 1.01i];

  % Factorize A = LQ
  [LQ, tau, info] = f08av(a);

  % Generate Q from LQ
  [Q, info] = f08aw(LQ, tau);

  % Extract L from LQ
  L = tril(LQ(1:m,1:m));

  % Bidiagonalize L = Q1 B P^H
  [B, d, e, tauq, taup, info] = ...
  f08ks(L);

  % Update Q: P2 = P^H*Q (so A = LQ = Q1 B P2)
  vect = 'P';
  side = 'Left';
  trans = 'Conjugate Transpose';
  [P2, info] = f08ku( ...
                    vect, side, trans, n, B, taup, Q);

  fprintf('Example 2: bidiagonal matrix B\n   Main diagonal  ');
  fprintf(' %7.3f',d);
  fprintf('\n   super-diagonal ');
  fprintf(' %7.3f',e);
  fprintf('\n\n');
  disp('Example 2: Orthogonal matrix P^H');
  disp(P2);
```

9.2 Program Results

```
     f08ku example results

Example 1: bidiagonal matrix B
   Main diagonal    -3.087  -2.066  -1.873  -2.002
   super-diagonal    2.113  -1.263   1.613

Example 1: Orthogonal matrix Q
  -0.3110 + 0.2624i   0.6521 + 0.5532i   0.0427 + 0.0361i  -0.2634 - 0.0741i
   0.3175 - 0.6414i   0.3488 + 0.0721i   0.2287 + 0.0069i   0.1101 - 0.0326i
  -0.2008 + 0.1490i  -0.3103 + 0.0230i   0.1855 - 0.1817i  -0.2956 + 0.5648i
   0.1199 - 0.1231i  -0.0046 - 0.0005i  -0.3305 + 0.4821i  -0.0675 + 0.3464i
  -0.2689 - 0.1652i   0.1794 - 0.0586i  -0.5235 - 0.2580i   0.3927 + 0.1450i
  -0.3499 + 0.0907i   0.0829 - 0.0506i   0.3202 + 0.3038i   0.3174 + 0.3241i

Example 2: bidiagonal matrix B
   Main diagonal     2.761   1.630  -1.327
   super-diagonal   -0.950  -1.018

Example 2: Orthogonal matrix P^H
  -0.1258 + 0.1618i  -0.2247 + 0.3864i   0.3460 + 0.2157i  -0.7099 - 0.2966i
   0.4148 + 0.1795i   0.1368 - 0.3976i   0.6885 + 0.3386i   0.1667 - 0.0494i
   0.4575 - 0.4807i  -0.2733 + 0.4981i  -0.0230 + 0.3861i   0.1730 + 0.2395i
```