NAG Toolbox

nag_lapack_zgeqlf (f08cs)

1 Purpose

nag_lapack_zgeqlf (f08cs) computes a QL factorization of a complex m by n matrix A.

2 Syntax

```
[a, tau, info] = nag_lapack_zgeqlf(a, 'm', m, 'n', n)
[a, tau, info] = f08cs(a, 'm', m, 'n', n)
```

3 Description

nag_lapack_zgeqlf (f08cs) forms the QL factorization of an arbitrary rectangular complex m by n matrix.

If $m \geq n$, the factorization is given by:

$$A = Q\begin{pmatrix} 0 \\ L \end{pmatrix},$$

where L is an n by n lower triangular matrix and Q is an m by m unitary matrix. If $m < n$ the factorization is given by

$$A = QL,$$

where L is an m by n lower trapezoidal matrix and Q is again an m by m unitary matrix. In the case where $m > n$ the factorization can be expressed as

$$A = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}\begin{pmatrix} 0 \\ L \end{pmatrix} = Q_2 L,$$

where Q_1 consists of the first $m - n$ columns of Q, and Q_2 the remaining n columns.

The matrix Q is not formed explicitly but is represented as a product of $\min(m, n)$ elementary reflectors (see Section 3.2.6 in the F08 Chapter Introduction for details). Functions are provided to work with Q in this representation (see Section 9).

Note also that for any $k < n$, the information returned in the last k columns of the array **a** represents a QL factorization of the last kendgroup columns of the original matrix A.

4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia http://www.netlib.org/lapack/lug

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

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

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

The m by n matrix A.

5.2 Optional Input Parameters

1: **m** – INTEGER

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

m, the number of rows of the matrix A.

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

2: **n** – INTEGER

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

n, the number of columns of the matrix A.

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

5.3 Output Parameters

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

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

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

If $m \geq n$, the lower triangle of the subarray $\mathbf{a}(m - n + 1 : m, 1 : n)$ contains the n by n lower triangular matrix L.

If $m \leq n$, the elements on and below the $(n - m)$th superdiagonal contain the m by n lower trapezoidal matrix L. The remaining elements, with the array **tau**, represent the unitary matrix Q as a product of elementary reflectors (see Section 3.2.6 in the F08 Chapter Introduction).

2: **tau**(:) – COMPLEX (KIND=nag_wp) array

The dimension of the array **tau** will be $\max(1, \min(\mathbf{m}, \mathbf{n}))$

The scalar factors of the elementary reflectors (see Section 9).

3: **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: **m**, 2: **n**, 3: **a**, 4: **lda**, 5: **tau**, 6: **work**, 7: **lwork**, 8: **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 factorization is the exact factorization of a nearby matrix $(A + E)$, where

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

and ϵ is the ***machine precision***.

8 Further Comments

The total number of real floating-point operations is approximately $\frac{8}{3}n^2(3m-n)$ if $m \geq n$ or $\frac{8}{3}m^2(3n-m)$ if $m < n$.

To form the unitary matrix Q nag_lapack_zgeqlf (f08cs) may be followed by a call to nag_lapack_zungql (f08ct):

```
[a, info] = f08ct(a, tau);
```

but note that the second dimension of the array **a** must be at least **m**, which may be larger than was required by nag_lapack_zgeqlf (f08cs).

When $m \geq n$, it is often only the first n columns of Q that are required, and they may be formed by the call:

```
[a, info] = f08ct(a, tau, 'k', n);
```

To apply Q to an arbitrary complex rectangular matrix C, nag_lapack_zgeqlf (f08cs) may be followed by a call to nag_lapack_zunmql (f08cu). For example,

```
[c, info] = f08cu('Left','Conjugate Transpose', a, tau, c);
```

forms $C = Q^{\mathrm{H}}C$, where C is m by p.

The real analogue of this function is nag_lapack_dgeqlf (f08ce).

9 Example

This example solves the linear least squares problems

$$\min_x \left\| b_j - Ax_j \right\|_2, \; j = 1, 2$$

for x_1 and x_2, where b_j is the jth column of the matrix B,

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

and

$$B = \begin{pmatrix} -2.09 + 1.93i & 3.26 - 2.70i \\ 3.34 - 3.53i & -6.22 + 1.16i \\ -4.94 - 2.04i & 7.94 - 3.13i \\ 0.17 + 4.23i & 1.04 - 4.26i \\ -5.19 + 3.63i & -2.31 - 2.12i \\ 0.98 + 2.53i & -1.39 - 4.05i \end{pmatrix}.$$

The solution is obtained by first obtaining a QL factorization of the matrix A.

Note that the block size (NB) of 64 assumed in this example is not realistic for such a small problem, but should be suitable for large problems.

9.1 Program Text

```
function f08cs_example

fprintf('f08cs example results\n\n');

% Find least squares solution of Ax=B (m>n) via QL factorization.
m = 6;
n = 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];
b = [-2.09 + 1.93i,  3.26 - 2.70i;
      3.34 - 3.53i, -6.22 + 1.16i;
     -4.94 - 2.04i,  7.94 - 3.13i;
      0.17 + 4.23i,  1.04 - 4.26i;
     -5.19 + 3.63i, -2.31 - 2.12i;
      0.98 + 2.53i, -1.39 - 4.05i];

% Compute the QL factorization of A
[ql, tau, info] = f08cs(a);

% LX = Q^H B = C; compute C = (Q^H)*B
[c, info] = f08cu( ...
    'Left', 'ConjTrans', ql, tau, b);

% Least-squares solution X = L^-1 C (lower n part)
il = m-n+1;
[x, info] = f07ts( ...
    'Lower', 'Notrans','Non-Unit', ql(il:m,:), c(il:m,:));

% Print least-squares solutions
ncols  = nag_int(80);
indent = nag_int(0);
[ifail] = x04db( ...
  'General', ' ', x, 'Bracketed', 'F7.4', ...
  'Least-squares solution(s)', 'Integer', 'Integer', ...
  ncols, indent);

% Compute estimates of the square roots of the residual sums of squares.
rnorm = zeros(2,1);
for j=1:2
  rnorm(j) = norm(c(1:m-n,j));
end
fprintf('\nSquare root(s) of the residual sum(s) of squares\n');
fprintf('\t%11.2e    %11.2e\n', rnorm(1), rnorm(2));
```

9.2 Program Results

```
     f08cs example results

 Least-squares solution(s)
                      1                 2
 1  (-0.5044,-1.2179) ( 0.7629, 1.4529)
 2  (-2.4281, 2.8574) ( 5.1570,-3.6089)
 3  ( 1.4872,-2.1955) (-2.6518, 2.1203)
 4  ( 0.4537, 2.6904) (-2.7606, 0.3318)

Square root(s) of the residual sum(s) of squares
    6.88e-02       1.87e-01
```