# NAG Toolbox

# nag_lapack_zgelqf (f08av)

## 1 Purpose

nag_lapack_zgelqf (f08av) computes the $LQ$ factorization of a complex $m$ by $n$ matrix.

## 2 Syntax

```
[a, tau, info] = nag_lapack_zgelqf(a, 'm', m, 'n', n)
[a, tau, info] = f08av(a, 'm', m, 'n', n)
```

## 3 Description

nag_lapack_zgelqf (f08av) forms the $LQ$ factorization of an arbitrary rectangular complex $m$ by $n$ matrix. No pivoting is performed.

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

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

where $L$ is an $m$ by $m$ lower triangular matrix (with real diagonal elements) and $Q$ is an $n$ by $n$ unitary matrix. It is sometimes more convenient to write the factorization as

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

which reduces to

$$A = LQ_1,$$

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

If $m > n$, $L$ is trapezoidal, and the factorization can be written

$$A = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} Q$$

where $L_1$ is lower triangular and $L_2$ is rectangular.

The $LQ$ factorization of $A$ is essentially the same as the $QR$ factorization of $A^{\mathrm{H}}$, since

$$A = \begin{pmatrix} L & 0 \end{pmatrix} Q \Leftrightarrow A^{\mathrm{H}} = Q^{\mathrm{H}} \begin{pmatrix} L^{\mathrm{H}} \\ 0 \end{pmatrix}.$$

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

Note also that for any $k < m$, the information returned in the first $k$ rows of the array **a** represents an $LQ$ factorization of the first $k$ rows of the original matrix $A$.

## 4 References

None.

# 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, **m**).

The second dimension of the array **a** must be at least max(1, **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*: **m** $\geq 0$.

2: **n** – INTEGER

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

$n$, the number of columns of the matrix $A$.

*Constraint*: **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, **m**).

The second dimension of the array **a** will be max(1, **n**).

If $m \leq n$, the elements above the diagonal store details of the unitary matrix $Q$ and the lower triangle stores the corresponding elements of the $m$ by $m$ lower triangular matrix $L$.

If $m > n$, the strictly upper triangular part stores details of the unitary matrix $Q$ and the remaining elements store the corresponding elements of the $m$ by $n$ lower trapezoidal matrix $L$.

The diagonal elements of $L$ are real.

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

The dimension of the array **tau** will be max(1, min(**m**, **n**))

Further details of the unitary matrix $Q$.

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 $\epsilon$ is the ***machine precision***.

## 8 Further Comments

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

To form the unitary matrix $Q$ nag_lapack_zgelqf (f08av) may be followed by a call to nag_lapack_zunglq (f08aw):

```
[a, info] = f08aw(a(1:n,:), tau);
```

but note that the first dimension of the array **a**, specified by the argument $lda$, must be at least **n**, which may be larger than was required by nag_lapack_zgelqf (f08av).

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

```
[a, info] = f08aw(a, tau, 'k', m);
```

To apply $Q$ to an arbitrary complex rectangular matrix $C$, nag_lapack_zgelqf (f08av) may be followed by a call to nag_lapack_zunmlq (f08ax). For example,

```
[c, info] = f08ax('Left', 'Conjugate Transpose', a(:,1:p), tau, c);
```

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

The real analogue of this function is nag_lapack_dgelqf (f08ah).

## 9 Example

This example finds the minimum norm solutions of the under-determined systems of linear equations

$$Ax_1 = b_1 \quad \text{and} \quad Ax_2 = b_2$$

where $b_1$ and $b_2$ are the columns of the matrix $B$,

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

and

$$B = \begin{pmatrix} -1.35 + 0.19i & 4.83 - 2.67i \\ 9.41 - 3.56i & -7.28 + 3.34i \\ -7.57 + 6.93i & 0.62 + 4.53i \end{pmatrix}.$$

### 9.1 Program Text

```
     function f08av_example

fprintf('f08av example results\n\n');

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];
b = [-1.35 + 0.19i,  4.83 - 2.67i;
      9.41 - 3.56i, -7.28 + 3.34i;
     -7.57 + 6.93i,  0.62 + 4.53i;
      0,             0];
[m,n] = size(a);
```

```
% Compute the LQ factorization of a
[lq, tau, info] = f08av(a);

% Solve l*y = b
l = tril(lq(:, 1:m));
y = [inv(l)*b(1:m,:); b(m+1:n,:)];

% Compute minimum-norm solution x = (qˆh)*y
[x, info] = f08ax( ...
    'Left', 'Conjugate Transpose', lq, tau, y);

disp('Minimum-norm solution(s)');
disp(x);
```

## 9.2 Program Results

```
     f08av example results

Minimum-norm solution(s)
  -2.8501 + 6.4683i  -1.1682 - 1.8886i
   1.6264 - 0.7799i   2.8377 + 0.7654i
   6.9290 + 4.6481i  -1.7610 - 0.7041i
   1.4048 + 3.2400i   1.0518 - 1.6365i
```