

NAG Toolbox

nag_lapack_zgeqrt (f08ap)

1 Purpose

nag_lapack_zgeqrt (f08ap) recursively computes, with explicit blocking, the QR factorization of a complex m by n matrix.

2 Syntax

```
[a, t, info] = nag_lapack_zgeqrt(nb, a, 'm', m, 'n', n)
[a, t, info] = f08ap(nb, a, 'm', m, 'n', n)
```

3 Description

nag_lapack_zgeqrt (f08ap) forms the QR factorization of an arbitrary rectangular complex m by n matrix. No pivoting is performed.

It differs from nag_lapack_zgeqrf (f08as) in that it: requires an explicit block size; stores reflector factors that are upper triangular matrices of the chosen block size (rather than scalars); and recursively computes the QR factorization based on the algorithm of Elmroth and Gustavson (2000).

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

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

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

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

which reduces to

$$A = Q_1 R,$$

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

If $m < n$, R is upper trapezoidal, and the factorization can be written

$$A = Q \begin{pmatrix} R_1 & R_2 \end{pmatrix},$$

where R_1 is upper triangular and R_2 is rectangular.

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 $k < n$, the information returned represents a QR factorization of the first k columns of the original matrix A .

4 References

Elmroth E and Gustavson F (2000) Applying Recursion to Serial and Parallel QR Factorization Leads to Better Performance *IBM Journal of Research and Development*. (Volume 44) **4** 605–624

Golub G H and Van Loan C F (2012) *Matrix Computations* (4th Edition) Johns Hopkins University Press, Baltimore

5 Parameters

5.1 Compulsory Input Parameters

- 1: **nb** – INTEGER

The explicitly chosen block size to be used in computing the QR factorization. See Section 9 for details.

Constraints:

$$\mathbf{nb} \geq 1;$$

$$\text{if } \min(\mathbf{m}, \mathbf{n}) > 0, \mathbf{nb} \leq \min(\mathbf{m}, \mathbf{n}).$$

- 2: **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 elements below the diagonal store details of the unitary matrix Q and the upper triangle stores the corresponding elements of the n by n upper triangular matrix R .

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

The diagonal elements of R are real.

- 2: **t**(*ldt*, :) – COMPLEX (KIND=nag_wp) array

The first dimension of the array **t** will be **nb**.

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

Further details of the unitary matrix Q . The number of blocks is $b = \lceil \frac{k}{\mathbf{nb}} \rceil$, where $k = \min(m, n)$ and each block is of order **nb** except for the last block, which is of order $k - (b - 1) \times \mathbf{nb}$. For each of the blocks, an upper triangular block reflector factor is computed: T_1, T_2, \dots, T_b . These are stored in the **nb** by n matrix T as $T = [T_1 | T_2 | \dots | T_b]$.

3: **info** – INTEGER

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

6 Error Indicators and Warnings

info < 0

If **info** = $-i$, argument i had an illegal value. An explanatory message is output, and execution of the program is terminated.

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 apply Q to an arbitrary complex rectangular matrix C , nag_lapack_zgemqrt (f08ap) may be followed by a call to nag_lapack_zgemqrt (f08aq). For example,

```
[t, c, info] = f08aq('Left', 'Conjugate Transpose', nb, a, t, c);
```

forms $C = Q^H C$, where C is m by p .

To form the unitary matrix Q explicitly, simply initialize the m by m matrix C to the identity matrix and form $C = QC$ using nag_lapack_zgemqrt (f08aq) as above.

The block size, **nb**, used by nag_lapack_zgemqrt (f08ap) is supplied explicitly through the interface. For moderate and large sizes of matrix, the block size can have a marked effect on the efficiency of the algorithm with the optimal value being dependent on problem size and platform. A value of **nb** = 64 \ll $\min(m, n)$ is likely to achieve good efficiency and it is unlikely that an optimal value would exceed 340.

To compute a QR factorization with column pivoting, use nag_lapack_ztpqrt (f08bp) or nag_lapack_zgeqpf (f08bs).

The real analogue of this function is nag_lapack_dgemqrt (f08ab).

9 Example

This example solves the linear least squares problems

$$\text{minimize } \|Ax_i - b_i\|_2, \quad i = 1, 2$$

where b_1 and b_2 are the columns 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

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

9.1 Program Text

```
function f08ap_example

fprintf('f08ap example results\n\n');

% Minimize ||Ax - b|| using recursive QR for m-by-n A and m-by-p B

m = nag_int(6);
n = nag_int(4);
p = nag_int(2);

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 QR Factorisation of A
[QR, T, info] = f08ap(n,a);

% Compute C = (C1) = (Q'H)*B
[c1, info] = f08aq(...
    'Left', 'Conjugate Transpose', QR, T, b);

% Compute least-squares solutions by backsubstitution in R*X = C1
[x, info] = f07ts(...
    'Upper', 'No Transpose', 'Non-Unit', QR, c1, 'n', n);

% Print least-squares solutions
disp('Least-squares solutions');
disp(x(1:n,:));

% Compute and print estimates of the square roots of the residual
% sums of squares
for j=1:p
    rnorm(j) = norm(x(n+1:m,j));
end
fprintf('\nSquare roots of the residual sums of squares\n');
fprintf('%12.2e', rnorm);
fprintf('\n');
```

9.2 Program Results

```
f08ap example results

Least-squares solutions
-0.5044 - 1.2179i   0.7629 + 1.4529i
-2.4281 + 2.8574i   5.1570 - 3.6089i
 1.4872 - 2.1955i  -2.6518 + 2.1203i
 0.4537 + 2.6904i  -2.7606 + 0.3318i

Square roots of the residual sums of squares
 6.88e-02   1.87e-01
```
