NAG Toolbox

nag_lapack_dgeqrt (f08ab)

1 Purpose

nag_lapack_dgeqrt (f08ab) recursively computes, with explicit blocking, the QR factorization of a real m by n matrix.

2 Syntax

```
[a, t, info] = nag_lapack_dgeqrt(nb, a, 'm', m, 'n', n)
[a, t, info] = f08ab(nb, a, 'm', m, 'n', n)
```

3 Description

nag_lapack_dgeqrt (f08ab) forms the QR factorization of an arbitrary rectangular real m by n matrix. No pivoting is performed.

It differs from nag_lapack_dgeqrf (f08ae) 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 and Q is an m by m orthogonal 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:

nb ≥ 1;
if $\min(\mathbf{m},\mathbf{n}) > 0$, **nb** $\leq \min(\mathbf{m},\mathbf{n})$.

2: **a**$(lda,:)$ – REAL (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: **m** ≥ 0.

2: **n** – INTEGER

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

n, the number of columns of the matrix A.

Constraint: **n** ≥ 0.

5.3 Output Parameters

1: **a**$(lda,:)$ – REAL (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 orthogonal 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 orthogonal matrix Q and the remaining elements store the corresponding elements of the m by n upper trapezoidal matrix R.

2: **t**$(ldt,:)$ – REAL (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 orthogonal matrix Q. The number of blocks is $b = \left\lceil \frac{k}{\mathbf{nb}} \right\rceil$, 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: $\boldsymbol{T}_1,\boldsymbol{T}_2,\ldots,\boldsymbol{T}_b$. These are stored in the **nb** by n matrix T as $\boldsymbol{T} = [\boldsymbol{T}_1|\boldsymbol{T}_2|\ldots|\boldsymbol{T}_b]$.

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 floating-point operations is approximately $\frac{2}{3}n^2(3m - n)$ if $m \geq n$ or $\frac{2}{3}m^2(3n - m)$ if $m < n$.

To apply Q to an arbitrary real rectangular matrix C, nag_lapack_dgeqrt (f08ab) may be followed by a call to nag_lapack_dgemqrt (f08ac). For example,

```
[t, c, info] = f08ac('Left', 'Transpose', nb, a, t, c, 'k', min(m,n));
```

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

To form the orthogonal matrix Q explicitly, simply initialize the m by m matrix C to the identity matrix and form $C = QC$ using nag_lapack_dgemqrt (f08ac) as above.

The block size, **nb**, used by nag_lapack_dgeqrt (f08ab) 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_dtpqrt (f08bb) or nag_lapack_dgeqpf (f08be).

The complex analogue of this function is nag_lapack_zgeqrt (f08ap).

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,

$$A = \begin{pmatrix} -0.57 & -1.28 & -0.39 & 0.25 \\ -1.93 & 1.08 & -0.31 & -2.14 \\ 2.30 & 0.24 & 0.40 & -0.35 \\ -1.93 & 0.64 & -0.66 & 0.08 \\ 0.15 & 0.30 & 0.15 & -2.13 \\ -0.02 & 1.03 & -1.43 & 0.50 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -2.67 & 0.41 \\ -0.55 & -3.10 \\ 3.34 & -4.01 \\ -0.77 & 2.76 \\ 0.48 & -6.17 \\ 4.10 & 0.21 \end{pmatrix}.$$

9.1 Program Text

```
function f08ab_example

fprintf('f08ab 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 = [-0.57, -1.28, -0.39,  0.25;
     -1.93,  1.08, -0.31, -2.14;
      2.30,  0.24,  0.40, -0.35;
     -1.93,  0.64, -0.66,  0.08;
      0.15,  0.30,  0.15, -2.13;
     -0.02,  1.03, -1.43,  0.50];
b = [-2.67,  0.41;
     -0.55, -3.10;
      3.34, -4.01;
     -0.77,  2.76;
      0.48, -6.17;
      4.10,  0.21];

% Compute the QR Factorisation of A
[QR, T, info] = f08ab(n,a);

% Compute C = (C1) = (Q^T)*B
[c1, info] = f08ac(...
                  'Left', 'Transpose', QR, T, b);

% Compute least-squares solutions by backsubstitution in R*X = C1
[x, info] = f07te(...
                  '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('Square roots of the residual sums of squares\n');
fprintf('%12.2e', rnorm);
fprintf('\n');
```

9.2 Program Results

```
     f08ab example results

Least-squares solutions
    1.5339   -1.5753
    1.8707    0.5559
   -1.5241    1.3119
    0.0392    2.9585

Square roots of the residual sums of squares
    2.22e-02    1.38e-02
```