NAG Toolbox

nag_lapack_dtpqrt (f08bb)

1 Purpose

nag_lapack_dtpqrt (f08bb) computes the QR factorization of a real $(m+n)$ by n triangular-pentagonal matrix.

2 Syntax

```
[a, b, t, info] = nag_lapack_dtpqrt(l, nb, a, b, 'm', m, 'n', n)
[a, b, t, info] = f08bb(l, nb, a, b, 'm', m, 'n', n)
```

3 Description

nag_lapack_dtpqrt (f08bb) forms the QR factorization of a real $(m+n)$ by n triangular-pentagonal matrix C,

$$C = \begin{pmatrix} A \\ B \end{pmatrix}$$

where A is an upper triangular n by n matrix and B is an m by n pentagonal matrix consisting of an $(m-l)$ by n rectangular matrix B_1 on top of an l by n upper trapezoidal matrix B_2:

$$B = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}.$$

The upper trapezoidal matrix B_2 consists of the first l rows of an n by n upper triangular matrix, where $0 \leq l \leq \min(m,n)$. If $l=0$, B is m by n rectangular; if $l=n$ and $m=n$, B is upper triangular.

A recursive, explicitly blocked, QR factorization (see nag_lapack_dgeqrt (f08ab)) is performed on the matrix C. The upper triangular matrix R, details of the orthogonal matrix Q, and further details (the block reflector factors) of Q are returned.

Typically the matrix A or B_2 contains the matrix R from the QR factorization of a subproblem and nag_lapack_dtpqrt (f08bb) performs the QR update operation from the inclusion of matrix B_1.

For example, consider the QR factorization of an l by n matrix $\hat{B}$ with $l<n$: $\hat{B} = \hat{Q}\hat{R}$, $\hat{R} = \begin{pmatrix} \hat{R}_1 & \hat{R}_2 \end{pmatrix}$, where $\hat{R}_1$ is l by l upper triangular and $\hat{R}_2$ is $(n-l)$ by n rectangular (this can be performed by nag_lapack_dgeqrt (f08ab)). Given an initial least-squares problem $\hat{B}\hat{X} = \hat{Y}$ where X and Y are l by $nrhs$ matrices, we have $\hat{R}\hat{X} = \hat{Q}^{\mathrm{T}}\hat{Y}$.

Now, adding an additional $m-l$ rows to the original system gives the augmented least squares problem

$$BX = Y$$

where B is an m by n matrix formed by adding $m-l$ rows on top of $\hat{R}$ and Y is an m by $nrhs$ matrix formed by adding $m-l$ rows on top of $\hat{Q}^{\mathrm{T}}\hat{Y}$.

nag_lapack_dtpqrt (f08bb) can then be used to perform the QR factorization of the pentagonal matrix B; the n by n matrix A will be zero on input and contain R on output.

In the case where $\hat{B}$ is r by n, $r \geq n$, $\hat{R}$ is n by n upper triangular (forming A) on top of $r-n$ rows of zeros (forming first $r-n$ rows of B). Augmentation is then performed by adding rows to the bottom of B with $l=0$.

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: **l** – INTEGER

l, the number of rows of the trapezoidal part of B (i.e., B_2).

Constraint: $0 \leq \mathbf{l} \leq \min(\mathbf{m}, \mathbf{n})$.

2: **nb** – INTEGER

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

Constraints:

$\mathbf{nb} \geq 1$;
if $\mathbf{n} > 0$, $\mathbf{nb} \leq \mathbf{n}$.

3: $\mathbf{a}(lda, :)$ – REAL (KIND=nag_wp) array

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

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

The n by n upper triangular matrix A.

4: $\mathbf{b}(ldb, :)$ – REAL (KIND=nag_wp) array

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

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

The m by n pentagonal matrix B composed of an $(m - l)$ by n rectangular matrix B_1 above an l by n upper trapezoidal matrix B_2.

5.2 Optional Input Parameters

1: **m** – INTEGER

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

m, the number of rows of the matrix B.

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

2: **n** – INTEGER

Default: the first dimension of the array **a** and the second dimension of the arrays **a**, **b**. (An error is raised if these dimensions are not equal.)

n, the number of columns of the matrix B and the order of the upper triangular matrix A.

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

5.3 Output Parameters

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

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

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

The upper triangle stores the corresponding elements of the n by n upper triangular matrix R.

2: **b**(ldb, :) – REAL (KIND=nag_wp) array

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

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

Details of the orthogonal matrix Q.

3: **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 **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 $\boldsymbol{T} = [\boldsymbol{T}_1 | \boldsymbol{T}_2 | \ldots | \boldsymbol{T}_b]$.

4: **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 $m < n$.

The block size, **nb**, used by nag_lapack_dtpqrt (f08bb) 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 $\mathbf{nb} = 64 \ll \min(m, n)$ is likely to achieve good efficiency and it is unlikely that an optimal value would exceed 340.

To apply Q to an arbitrary real rectangular matrix C, nag_lapack_dtpqrt (f08bb) may be followed by a call to nag_lapack_dtpmqrt (f08bc). For example,

```
[t, c, info] = f08bc('Left','Transpose', nb, a(:,1:min(m,n)), t, c);
```

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

To form the orthogonal matrix Q explicitly set $p = m + n$, initialize C to the identity matrix and make a call to nag_lapack_dtpmqrt (f08bc) as above.

9 Example

This example finds the basic solutions for 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,

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

A QR factorization is performed on the first 4 rows of A using nag_lapack_dgeqrt (f08ab) after which the first 4 rows of B are updated by applying Q^T using nag_lapack_dgemqrt (f08ac). The remaining row is added by performing a QR update using nag_lapack_dtpqrt (f08bb); B is updated by applying the new Q^T using nag_lapack_dtpmqrt (f08bc); the solution is finally obtained by triangular solve using R from the updated QR.

9.1 Program Text

```
     function f08bb_example

fprintf('f08bb 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];

nb = n;
% Compute the QR Factorisation of first n rows of A
[QRn, Tn, info] = f08ab( ...
  nb,a(1:n,:));

% Compute C = (C1) = (Q^T)*B
[c1, info] = f08ac( ...
     'Left', 'Transpose', QRn, Tn, b(1:n,:));

% Compute least-squares solutions by backsubstitution in R*X = C1
[x, info] = f07te( ...
    'Upper', 'No Transpose', 'Non-Unit', QRn, c1);

% Print first n-row solutions
disp('Solution for n rows');
disp(x(1:n,:));

% Add the remaining rows and perform QR update
nb2 = m-n;
```

```
l = nag_int(0);
[R, Q, T, info] = f08bb( ...
  l, nb2, QRn, a(n+1:m,:));

% Apply orthogonal transformations to C
[c1,c2,info] = f08bc( ...
       'Left','Transpose', l, Q, T, c1, b(n+1:m,:));

% Compute least-squares solutions for first n rows: R*X = C1
[x, info] = f07te( ...
    'Upper', 'No transpose', 'Non-Unit', R, c1);
% Print least-squares solutions for all m rows
disp('Least squares solution');
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(c2(:,j));
end
fprintf('Square roots of the residual sums of squares\n');
fprintf('%12.2e', rnorm);
fprintf('\n');
```

9.2 Program Results

```
     f08bb example results

Solution for n rows
    1.5179   -1.5850
    1.8629    0.5531
   -1.4608    1.3485
    0.0398    2.9619

Least squares solution
    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
```