NAG Toolbox

nag_lapack_zggrqf (f08zt)

1 Purpose

nag_lapack_zggrqf (f08zt) computes a generalized RQ factorization of a complex matrix pair (A, B), where A is an m by n matrix and B is a p by n matrix.

2 Syntax

```
[a, taua, b, taub, info] = nag_lapack_zggrqf(a, b, 'm', m, 'p', p, 'n', n)
[a, taua, b, taub, info] = f08zt(a, b, 'm', m, 'p', p, 'n', n)
```

3 Description

nag_lapack_zggrqf (f08zt) forms the generalized RQ factorization of an m by n matrix A and a p by n matrix B

$$A = RQ, \quad B = ZTQ,$$

where Q is an n by n unitary matrix, Z is a p by p unitary matrix and R and T are of the form

$$R = \begin{cases} \begin{array}{c} \\ m \end{array} \begin{array}{c} \begin{array}{cc} n-m & m \end{array} \\ \begin{pmatrix} 0 & R_{12} \end{pmatrix} \end{array}; & \text{if } m \leq n, \\ \begin{array}{c} n \\ \\ m-n \\ \\ n \end{array} \begin{pmatrix} \\ R_{11} \\ \\ R_{21} \end{pmatrix}; & \text{if } m > n, \end{cases}$$

with R_{12} or R_{21} upper triangular,

$$T = \begin{cases} \begin{array}{c} \\ p-n \end{array} \begin{array}{c} \begin{array}{c} n \end{array} \\ \begin{pmatrix} T_{11} \\ 0 \end{pmatrix} \end{array}; & \text{if } p \geq n, \\ \begin{array}{c} p \end{array} \begin{array}{c} \begin{array}{cc} p & n-p \end{array} \\ \begin{pmatrix} T_{11} & T_{12} \end{pmatrix} \end{array}; & \text{if } p < n, \end{cases}$$

with T_{11} upper triangular.

In particular, if B is square and nonsingular, the generalized RQ factorization of A and B implicitly gives the RQ factorization of AB^{-1} as

$$AB^{-1} = \left(RT^{-1}\right)Z^{\mathrm{H}}.$$

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

Anderson E, Bai Z and Dongarra J (1992) Generalized *QR* factorization and its applications *Linear Algebra Appl. (Volume 162–164)* 243–271

Hammarling S (1987) The numerical solution of the general Gauss-Markov linear model *Mathematics in Signal Processing* (eds T S Durrani, J B Abbiss, J E Hudson, R N Madan, J G McWhirter and T A Moore) 441–456 Oxford University Press

Paige C C (1990) Some aspects of generalized QR factorizations . *In Reliable Numerical Computation* (eds M G Cox and S Hammarling) 73–91 Oxford University Press

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.

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

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

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

The p by n matrix B.

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: **p** – INTEGER

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

p, the number of rows of the matrix B.

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

3: **n** – INTEGER

Default: the second dimension of the arrays **a**, **b**.

n, the number of columns of the matrices A and B.

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 $m \leq n$, the upper triangle of the subarray $\mathbf{a}(1 : m, n - m + 1 : n)$ contains the m by m upper triangular matrix R_{12}.

If $m \geq n$, the elements on and above the $(m - n)$th subdiagonal contain the m by n upper trapezoidal matrix R; the remaining elements, with the array **taua**, represent the unitary matrix Q as a product of $\min(m, n)$ elementary reflectors (see Section 3.2.6 in the F08 Chapter Introduction).

2: **taua**(**min**(**m**, **n**)) – COMPLEX (KIND=nag_wp) array

The scalar factors of the elementary reflectors which represent the unitary matrix Q.

3: **b**(ldb, :) – COMPLEX (KIND=nag_wp) array

The first dimension of the array **b** will be max(1, **p**).

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

The elements on and above the diagonal of the array contain the min(p, n) by n upper trapezoidal matrix T (T is upper triangular if $p \geq n$); the elements below the diagonal, with the array **taub**, represent the unitary matrix Z as a product of elementary reflectors (see Section 3.2.6 in the F08 Chapter Introduction).

4: **taub**(**min**(**p**, **n**)) – COMPLEX (KIND=nag_wp) array

The scalar factors of the elementary reflectors which represent the unitary matrix Z.

5: **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: **p**, 3: **n**, 4: **a**, 5: **lda**, 6: **taua**, 7: **b**, 8: **ldb**, 9: **taub**, 10: **work**, 11: **lwork**, 12: **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 generalized RQ factorization is the exact factorization for nearby matrices $(A + E)$ and $(B + F)$, where

$$\|E\|_2 = O\epsilon\|A\|_2 \quad \text{and} \quad \|F\|_2 = O\epsilon\|B\|_2,$$

and ϵ is the ***machine precision***.

8 Further Comments

The unitary matrices Q and Z may be formed explicitly by calls to nag_lapack_zungrq (f08cw) and nag_lapack_zungqr (f08at) respectively. nag_lapack_zunmrq (f08cx) may be used to multiply Q by another matrix and nag_lapack_zunmqr (f08au) may be used to multiply Z by another matrix.

The real analogue of this function is nag_lapack_dggrqf (f08zf).

9 Example

This example solves the least squares problem

$$\underset{x}{\text{minimize}} \|c - Ax\|_2 \quad \text{subject to} \quad Bx = d$$

where

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

$$B = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}, \quad c = \begin{pmatrix} -2.54+0.09i \\ 1.65-2.26i \\ -2.11-3.96i \\ 1.82+3.30i \\ -6.41+3.77i \\ 2.07+0.66i \end{pmatrix} \quad \text{and} \quad d = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

The constraints $Bx = d$ correspond to $x_1 = x_3$ and $x_2 = x_4$.

The solution is obtained by first obtaining a generalized RQ factorization of the matrix pair (A, B). The example illustrates the general solution process, although the above data corresponds to a simple weighted least squares problem.

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 f08zt_example

fprintf('f08zt example results\n\n');

% Find x that minimizes norm(c-Ax) subject to Bx = d .
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 = complex([ 1 0 -1  0;
              0 1  0 -1]);
c = [-2.54+0.09i;
      1.65-2.26i;
     -2.11-3.96i;
      1.82+3.30i;
     -6.41+3.77i;
      2.07+0.66i];
d = complex([0;0]);

% Compute the generalized RQ factorization of (B,A) as
% A = ZRQ, B = TQ
[TQ, taub, ZR, taua, info] = f08zt(b, a);

% Set Qx = y. The problem reduces to:
% minimize (Ry - Z'Hc) subject to Ty = d

% Update c = Z'H*c -> minimize (Ry-c)
[cup, info] = f08au( ...
                    'Left','Conjugate Transpose',ZR,taua,c);

% Solve Ty = d for last p elements
T12 = complex(triu(TQ(1:p,n-p+1:n)));

[y2, info] = f07ts( ...
                   'Upper', 'No transpose', 'Non-unit', T12, d);

% (from Ry-c) R11*y1 + R12*y2 = c1 --> R11*y1 = c1 - R12*y2
```

```
% Update c1
c1 = cup(1:n-p) - ZR(1:n-p,n-p+1:n)*y2;

% Solve R11*y1 = c1 for y1
  R11 = complex(triu(ZR(1:n-p,1:n-p)));
[y1, info] = f07ts( ...
                    'Upper', 'No transpose', 'Non-unit', R11, c1);

% Contruct y and backtransform for x = Q^Hy
y = [y1;y2];
[~, x, info] = f08cx( ...
                     'Left', 'Conjugate Transpose', TQ, taub, y);

disp('Constrained least squares solution');
disp(x);

res = a*x - c;
fprintf('Square root of the residual sum of squares\n%11.2e\n', ...
        norm(res));
```

9.2 Program Results

```
     f08zt example results

Constrained least squares solution
   1.0874 - 1.9621i
  -0.7409 + 3.7297i
   1.0874 - 1.9621i
  -0.7409 + 3.7297i

Square root of the residual sum of squares
   1.59e-01
```