nag_complex_qr (f01rcc)

1. Purpose

nag_complex_qr (f01rcc) finds the QR factorization of the complex m by n matrix A, where $m \geq n$.

2. Specification

```
#include <nag.h>
#include <nagf01.h>

void nag_complex_qr(Integer m, Integer n, Complex a[], Integer tda,
     Complex theta[], NagError *fail)
```

3. Description

The m by n matrix A is factorized as

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix} \quad \text{when } m > n$$
$$A = QR \quad \text{when } m = n$$

where Q is an m by m unitary matrix and R is an n by n upper triangular matrix with real diagonal elements.

The factorization is obtained by Householder's method. The kth transformation matrix, Q_k, which is used to introduce zeros into the kth column of A is given in the form

$$Q_k = \begin{pmatrix} I & 0 \\ 0 & T_k \end{pmatrix},$$

where

$$T_k = I - \gamma_k u_k u_k^H$$
$$u_k = \begin{pmatrix} \zeta_k \\ z_k \end{pmatrix},$$

γ_k is a scalar for which $\mathrm{Re}\,\gamma_k = 1.0$, ζ_k is a real scalar and z_k is an $(m-k)$ element vector. γ_k, ζ_k and z_k are chosen to annihilate the elements below the triangular part of A and to make the diagonal elements real.

The scalar γ_k and the vector u_k are returned in the $(k-1)$th element of the array **theta** and in the $(k-1)$th column of **a**, such that θ_k, given by

$$\theta_k = (\zeta_k, \mathrm{Im}\gamma_k),$$

is in **theta**$[k-1]$ and the elements of z_k are in **a**$[k][k+1], \ldots,$**a**$[m-1][k-1]$. The elements of R are returned in the upper triangular part of A.

Q is given by

$$Q = (Q_n Q_{n-1} \cdots Q_1)^H.$$

A good background description to the QR factorization is given in Dongarra *et al*(1979).

4. Parameters

m

Input: m, the number of rows of A.
Constraint: $\mathbf{m} \geq \mathbf{n}$.

n

Input: n, the number of columns of A.
Constraint: $\mathbf{n} \geq 0$.
When $\mathbf{n} = 0$ then an immediate return is effected.

a[m][tda]

Input: the leading m by n part of the array **a** must contain the matrix to be factorized.
Output: the n by n upper triangular part of **a** will contain the upper triangular matrix R, with the imaginary parts of the diagonal elements set to zero, and the m by n strictly lower triangular part of **a** will contain details of the factorization as described above.

tda

Input: the second dimension of the array **a** as declared in the function from which nag_complex_qr is called.
Constraint: $\mathbf{tda} \geq \mathbf{n}$.

theta[n]

Output: the scalar θ_k for the kth transformation. If $T_k = I$ then $\mathbf{theta}[k-1] = 0.0$; if

$$T_k = \begin{pmatrix} \alpha & 0 \\ 0 & I \end{pmatrix} \quad \mathrm{Re}\,\alpha < 0.0$$

then $\mathbf{theta}[k-1] = \alpha$; otherwise $\mathbf{theta}[k-1]$ contains **theta** $[k-1]$ as described in Section 3 and $\mathrm{Re}(\mathbf{theta}[k-1])$ is always in the range $(1.0, \sqrt{2.0})$.

fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_2_INT_ARG_LT

On entry, $\mathbf{m} = \langle \mathit{value} \rangle$ while $\mathbf{n} = \langle \mathit{value} \rangle$. These parameters must satisfy $\mathbf{m} \geq \mathbf{n}$.
On entry, $\mathbf{tda} = \langle \mathit{value} \rangle$ while $\mathbf{n} = \langle \mathit{value} \rangle$. These parameters must satisfy $\mathbf{tda} \geq \mathbf{n}$.

NE_INT_ARG_LT

On entry, **n** must not be less than 0: $\mathbf{n} = \langle \mathit{value} \rangle$.

6. Further Comments

The approximate number of real floating-point operations is given by $8n^2(3m-n)/3$.

Following the use of this function the operations

$$B := QB \quad \text{and} \quad B := Q^H B$$

where B is an m by k matrix, can be performed by calls to nag_complex_apply_q (f01rdc).

The operation $B := QB$ can be obtained by the call:

```
f01rdc(NoTranspose, Nag_ElementsSeparate, m, n, (Complex *) a, tda,
       theta, k, (Complex *) b, tdb, &fail)
```

and $B := Q^H B$ can be obtained by the call:

```
f01rdc(ConjugateTranspose, Nag_ElementsSeparate, m, n, (Complex *) a,
       tda, theta, k, (Complex *) b, tdb, &fail)
```

If B is a one-dimensional array (single column) then the parameter *tdb* can be replaced by 1. See nag_complex_apply_q (f01rdc) for further details.

The first k columns of the unitary matrix Q can either be obtained by setting B to the first k columns of the unit matrix and using the first of the above two calls, or by calling nag_complex_form_q (f01rec), which overwrites the k columns of Q on the first k columns of the array **a**. Q is obtained by the call:

```
f01rec(Nag_ElementsSeparate, m, n, k, (Complex *) a, tda, theta, &fail)
```

If k is larger than n, then A must have been declared to have at least k columns.

6.1. Accuracy

The computed factors Q and R satisfy the relation

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

where $\|E\| \leq c\epsilon\|A\|$, ϵ being the ***machine precision***, c is a modest function of m and n and $\|.\|$ denotes the spectral (two) norm.

6.2. References

Dongarra J J, Moler C B, Bunch J R and Stewart G W (1979) *LINPACK Users' Guide* SIAM, Philadelphia.
Wilkinson J H (1965) *The Algebraic Eigenvalue Problem* Clarendon Press, Oxford.

7. See Also

nag_complex_apply_q (f01rdc)
nag_complex_form_q (f01rec)

8. Example

To obtain the QR factorization of the 5 by 3 matrix

$$A = \begin{pmatrix} 0.5i & -0.5+1.5i & -1.0+1.0i \\ 0.4+0.3i & 0.9+1.3i & 0.2+1.4i \\ 0.4 & -0.4+0.4i & 1.8 \\ 0.3-0.4i & 0.1+0.7i & 0.0 \\ -0.3i & 0.3+0.3i & 2.4i \end{pmatrix}$$

8.1. Program Text

```
/* nag_complex_qr(f01rcc) Example Program
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 1, 1990.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagf01.h>

#define MMAX 20
#define NMAX 10
#define TDA NMAX
#define COMPLEX(A) A.re, A.im

main()
{
  Integer i, j, m, n;
  static NagError fail;
  Complex a[MMAX][TDA], theta[NMAX];

  /* Skip heading in data file */
  Vscanf("%*[^\n]");
  Vprintf("f01rcc Example Program Results\n");
  Vscanf("%ld%ld", &m, &n);
  Vprintf("\n");
  if (m>MMAX || n>NMAX)
    {
      Vfprintf(stderr, "\n m or n is out of range.\n");
      Vfprintf(stderr, "m = %ld    n = %ld\n", m, n);
      exit(EXIT_FAILURE);
```

```
    }
  for (i=0; i<m; ++i)
    for (j=0; j<n; ++j)
      Vscanf(" ( %lf , %lf ) ", COMPLEX(&a[i][j]));
  /* Find the QR factorization of A. */
  fail.print = TRUE;
  f01rcc(m, n, (Complex *)a, (Integer)TDA, theta, &fail);
  if (fail.code != NE_NOERROR)
    exit(EXIT_FAILURE);
  Vprintf("QR factorization of A\n");
  Vprintf("Vector THETA\n");
  for (i=0; i<n; ++i)
    Vprintf(" (%7.4f,%8.4f)%s", COMPLEX(theta[i]),
            (i%3==2 || i==n-1) ? "\n" : " ");
  Vprintf("\nMatrix A after factorization (upper triangular part is R)\n");
  for (i=0; i<m; ++i)
    {
      for (j=0; j<n; ++j)
        Vprintf(" (%7.4f,%8.4f)%s", COMPLEX(a[i][j]),
                (j%3==2 || j==n-1) ? "\n" : " ");
    }
  exit(EXIT_SUCCESS);
}
```

8.2. Program Data

```
f01rcc Example Program Data

   5     3

  ( 0.0,  0.5 )   (-0.5,  1.5)   (-1.0,  1.0)
  ( 0.4,  0.3 )   ( 0.9,  1.3)   ( 0.2,  1.4)
  ( 0.4,  0.0 )   (-0.4,  0.4)   ( 1.8,  0.0)
  ( 0.3, -0.4 )   ( 0.1,  0.7)   ( 0.0,  0.0)
  ( 0.0, -0.3 )   ( 0.3,  0.3)   ( 0.0,  2.4)
```

8.3. Program Results

```
f01rcc Example Program Results

QR factorization of A
Vector THETA
 ( 1.0000,  0.5000)  ( 1.0954, -0.3333)  ( 1.2649,  0.0000)

Matrix A after factorization (upper triangular part is R)
 ( 1.0000,  0.0000)  ( 1.0000,  1.0000)  ( 1.0000,  1.0000)
 (-0.2000, -0.4000)  (-2.0000,  0.0000)  (-1.0000, -1.0000)
 (-0.3200, -0.1600)  (-0.3505,  0.2629)  (-3.0000,  0.0000)
 (-0.4000,  0.2000)  ( 0.0000,  0.5477)  ( 0.0000,  0.0000)
 (-0.1200,  0.2400)  ( 0.1972,  0.2629)  ( 0.0000,  0.6325)
```