NAG Library Function Document

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

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

$$T_k = I - u_k u_k^{\mathrm{T}},$$

$$u_k = \begin{pmatrix} \zeta_k \\ z_k \end{pmatrix},$$

γ_k is a scalar for which $\operatorname{Re}\gamma_k = 1.0$, 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

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

is in **theta**$[k-1]$ and the elements of z_k are in **a**$[(k) \times$ **tda** $+ k + 1], \ldots,$ **a**$[(m-1) \times$ **tda** $+ k - 1]$. 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 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* Oxford University Press, Oxford

5 Arguments

1: **m** – Integer *Input*

On entry: m, the number of rows of A.

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

2: **n** – Integer *Input*

On entry: n, the number of columns of A.

Constraints:

$\mathbf{n} \geq 0$;
when $\mathbf{n} = 0$ then an immediate return is effected.

3: **a**[**m** × **tda**] – Complex *Input/Output*

On entry: the leading m by n part of the array **a** must contain the matrix to be factorized.

On exit: 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.

4: **tda** – Integer *Input*

On entry: the stride separating matrix column elements in the array **a**.

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

5: **theta**[**n**] – Complex *Output*

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

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

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

6: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_2_INT_ARG_LT

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

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

NE_INT_ARG_LT

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 0$.

7 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.

8 Parallelism and Performance

Not applicable.

9 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:

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

If B is a one-dimensional array (single column) then the argument *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:

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

10 Example

To obtain the QR factorization of the 5 by 3 matrix

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

10.1 Program Text

```
/* nag_complex_qr (f01rcc) Example Program.
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 1, 1990.
 * Mark 8 revised, 2004.
 */

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

#define COMPLEX(A) A.re, A.im
#define A(I, J)    a[(I) *tda + J]

int main(void)
```

```
{
  Complex  *a = 0, *theta = 0;
  Integer  exit_status = 0, i, j, m, n, tda;
  NagError fail;

  INIT_FAIL(fail);

  /* Skip heading in data file */
  scanf("%*[^\n]");
  printf("nag_complex_qr (f01rcc) Example Program Results\n");
  scanf("%ld%ld", &m, &n);
  printf("\n");
  if (n >= 0 && m >= n)
    {
      if (!(a = NAG_ALLOC(m*n, Complex)) ||
          !(theta = NAG_ALLOC(n, Complex)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tda = n;
    }
  else
    {
      printf("Invalid n or m.\n");
      exit_status = 1;
      return exit_status;
    }
  for (i = 0; i < m; ++i)
    for (j = 0; j < n; ++j)
      scanf(" ( %lf,  %lf ) ", COMPLEX(&A(i, j)));
  /* Find the QR factorization of A. */
  /* nag_complex_qr (f01rcc).
   * QR factorization of complex m by n matrix (m >= n)
   */
  nag_complex_qr(m, n, a, tda, theta, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_complex_qr (f01rcc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  printf("QR factorization of A\n");
  printf("Vector THETA\n");
  for (i = 0; i < n; ++i)
    printf(" (%7.4f,%8.4f)%s", COMPLEX(theta[i]),
            (i%3 == 2 || i == n-1)?"\n":" ");
  printf(
          "\nMatrix A after factorization (upper triangular part is R)\n");
  for (i = 0; i < m; ++i)
    {
      for (j = 0; j < n; ++j)
        printf(" (%7.4f,%8.4f)%s", COMPLEX(A(i, j)),
                (j%3 == 2 || j == n-1)?"\n":" ");
    }
 END:
  NAG_FREE(a);
  NAG_FREE(theta);
  return exit_status;
}
```

10.2 Program Data

```
nag_complex_qr (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)
```

10.3 Program Results

```
nag_complex_qr (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)
```