

nag_complex_apply_q (f01rdc)

1. Purpose

nag_complex_apply_q (f01rdc) performs one of the transformations

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

where B is an m by $ncolb$ complex matrix and Q is an m by m unitary matrix, given as the product of Householder transformation matrices.

This function is intended for use following **nag_complex_qr (f01rcc)**.

2. Specification

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

```
void nag_complex_apply_q(MatrixTranspose trans, Nag_WhereElements wheret,
    Integer m, Integer n, Complex a[], Integer tda, Complex theta[],
    Integer ncolb, Complex b[], Integer tdb, NagError *fail)
```

3. Description

The unitary matrix Q is assumed to be given by

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

Q_k being 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 $\text{Re } \gamma_k = 1.0$, ζ_k is a real scalar and z_k is an $(m - k)$ element vector.

z_k must be supplied in the $(k - 1)$ th column of **a** in elements **a**[k][$k - 1$], ..., **a**[$m - 1$][$k - 1$] and θ_k , given by

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

must be supplied either in **a**[$k - 1$][$k - 1$] or in **theta**[$k - 1$], depending upon the parameter **wheret**.

To obtain Q explicitly B may be set to I and premultiplied by Q . This is more efficient than obtaining Q^H . Alternatively, **nag_complex_form_q (f01rec)** may be used to obtain Q overwritten on A .

4. Parameters

trans

Input: the operation to be performed as follows:

trans = **NoTranspose**, perform the operation $B := QB$.

trans = **ConjugateTranspose**, perform the operation $B := Q^H B$.

Constraint: **trans** must be one of **NoTranspose** or **ConjugateTranspose**.

wheret

Input: the elements of θ are to be found as follows:

wheret = **Nag_ElementsIn** The elements of θ are in A .

wheret = **Nag_ElementsSeparate** The elements of θ are separate from A , in **theta**.

Constraint: **wheret** must be one of **Nag_ElementsIn** or **Nag_ElementsSeparate**.

mInput: m , the number of rows of A .Constraint: $\mathbf{m} \geq \mathbf{n}$.**n**Input: n , the number of columns of A .When $\mathbf{n} = 0$ then an immediate return is effected.Constraint: $\mathbf{n} \geq 0$.**a[m][tda]**Input: the leading m by n strictly lower triangular part of the array **a** must contain details of the matrix Q . In addition, when **wheret** = **Nag_ElementsIn**, then the diagonal elements of **a** must contain the elements of θ as described under the parameter **theta** below.When **wheret** = **Nag_ElementsSeparate**, then the diagonal elements of the array **a** are referenced, since they are used temporarily to store the ζ_k , but they contain their original values on return.**tda**Input: the second dimension of the array **a** as declared in the function from which nag_complex_apply_q is called.Constraint: $\mathbf{tda} \geq \mathbf{n}$.**theta[n]**Input: with **wheret** = **Nag_ElementsSeparate**, the array **theta** must contain the elements of θ . If **theta**[$k-1$] = 0.0 then T_k is assumed to be I ; if **theta**[$k-1$] = α , with $\text{Re } \alpha < 0.0$, then T_k is assumed to be of the form

$$T_k = \begin{pmatrix} \alpha & 0 \\ 0 & I \end{pmatrix};$$

otherwise **theta**[$k-1$] is assumed to contain θ_k given by $\theta_k = (\zeta_k, \text{Im}\gamma_k)$.When **wheret** = **Nag_ElementsIn**, the array **theta** is not referenced, and may be set to the null pointer, i.e., (Complex *)0.**ncolb**Input: ncolb , the number of columns of B .When **ncolb** = 0 then an immediate return is effected.Constraint: $\mathbf{ncolb} \geq 0$.**b[m][tdb]**Input: the leading m by ncolb part of the array **b** must contain the matrix to be transformed.Output: **b** is overwritten by the transformed matrix.**tdb**Input: the second dimension of the array **b** as declared in the function from which nag_complex_apply_q is called.Constraint: $\mathbf{tdb} \geq \mathbf{ncolb}$.**fail**

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

5. Error Indications and Warnings

NE_BAD_PARAM

On entry, parameter **trans** had an illegal value.On entry, parameter **wheret** had an illegal value.

NE_2_ARG_LT

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

NE_INT_ARG_LT

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

NE_ALLOC_FAIL

Memory allocation failed.

6. Further Comments

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

6.1. Accuracy

Letting C denote the computed matrix $Q^H B$, C satisfies the relation

$$QC = B + E$$

where $\|E\| \leq c\epsilon\|B\|$, ϵ being the **machine precision**, c is a modest function of m and $\|\cdot\|$ denotes the spectral (two) norm. An equivalent result holds for the computed matrix QB . See also Section 6.1 of nag_complex_qr (f01rcc).

6.2. References

Wilkinson J H (1965) *The Algebraic Eigenvalue Problem* Clarendon Press, Oxford.

7. See Also

nag_complex_form_q (f01rec)
nag_complex_qr (f01rcc)

8. Example

To obtain the matrix $Q^H B$ for the matrix B given by

$$B = \begin{pmatrix} -0.55 + 1.05i & 0.45 + 1.05i \\ 0.49 + 0.93i & 1.09 + 0.13i \\ 0.56 - 0.16i & 0.64 + 0.16i \\ 0.39 + 0.23i & -0.39 - 0.23i \\ 1.13 + 0.83i & -1.13 + 0.77i \end{pmatrix}$$

following the QR factorization of the 5 by 3 matrix A given by

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

8.1. Program Text

```
/* nag_complex_apply_q(f01rdc) 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 NCBMAX 5
#define TDA NMAX
#define TDB NCBMAX
#define COMPLEX(A) A.re, A.im
```

```

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

    Vprintf("f01rdc Example Program Results\n");
    /* Skip heading in data file */
    Vscanf("%*[^\\n]");
    Vscanf("%ld%ld", &m, &n);
    if (m>MMAX || n>NMAX)
    {
        Vfprintf(stderr, "\\n m or n is out of range.\\n");
        Vfprintf(stderr, "m = %ld      n = %ld", m, n);
        exit(EXIT_FAILURE);
    }
    for (i=0; i<m; ++i)
        for (j=0; j<n; ++j)
            Vscanf(" ( %lf , %lf ) ", COMPLEX(&a[i][j]));
    Vscanf("%ld", &ncolb);
    if (ncolb>NCBMAX)
    {
        Vprintf("\\n ncolb is out of range.\\n ncolb = %ld\\n", ncolb);
        exit(EXIT_FAILURE);
    }
    for (i=0; i<m; ++i)
        for (j=0; j<ncolb; ++j)
            Vscanf(" ( %lf , %lf ) ", COMPLEX(&b[i][j]));
    /* Find the QR factorization of A. */
    fail.print = TRUE;
    f01rcc(m, n, (Complex *)a, (Integer)TDA, theta, &fail);

    /* Form conjg( Q' ) * B. */

    f01rdc(ConjugateTranspose, Nag_ElementsSeparate, m, n, (Complex *)a, (Integer)
        theta, ncolb, (Complex *)b, (Integer)TDB, &fail);

    if (fail.code != NE_NOERROR)
        exit(EXIT_FAILURE);
    Vprintf("\\nMatrix conjg( Q' ) * B\\n");
    for (i=0; i<m; ++i)
    {
        for (j=0; j<ncolb; ++j)
            Vprintf(" (%7.4f, %8.4f)%s", COMPLEX(b[i][j]),
                (j%2==1 || j==n-1) ? "\\n" : " ");
    }
    exit (EXIT_SUCCESS);
}

```

8.2. Program Data

f01rdc Example Program Data

```

5      3

(0.00, 0.50)  (-0.50, 1.50)  (-1.00, 1.00)
(0.40, 0.30)  ( 0.90, 1.30)  ( 0.20, 1.40)
(0.40, 0.00)  (-0.40, 0.40)  ( 1.80, 0.00)
(0.30, -0.40) ( 0.10, 0.70)  ( 0.00, 0.00)
(0.00, -0.30) ( 0.30, 0.30)  ( 0.00, 2.40)

2

(-0.55, 1.05) ( 0.45, 1.05)
(0.49, 0.93)  ( 1.09, 0.13)
(0.56, -0.16) ( 0.64, 0.16)
(0.39, 0.23)  (-0.39, -0.23)
(1.13, 0.83)  (-1.13, 0.77)

```

8.3. Program Results

f01rdc Example Program Results

```
Matrix  conjg( Q' )*B
( 1.0000,  1.0000)  ( 1.0000, -1.0000)
(-1.0000,  0.0000)  (-1.0000,  0.0000)
(-1.0000,  1.0000)  (-1.0000, -1.0000)
(-0.0600, -0.0200)  (-0.0400,  0.1200)
( 0.0400,  0.1200)  (-0.0600,  0.0200)
```
