nag_real_form_q (f01qec)

1. Purpose

nag_real_form_q (f01qec) returns the first *ncolq* columns of the real m by m orthogonal matrix Q, where Q is given as the product of Householder transformation matrices.
This function is intended for use following nag_real_qr (f01qcc).

2. Specification

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

void nag_real_form_q(Nag_WhereElements wheret, Integer m, Integer n,
     Integer ncolq, double a[], Integer tda, double zeta[], NagError *fail)
```

3. Description

Q is assumed to be given by

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

Q_k being given in the form

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

where

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

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

4. Parameters

wheret

Input: indicates where the elements of ζ are to be found as follows:
wheret = **Nag_ElementsIn**, the elements of ζ are in **a**.
wheret = **Nag_ElementsSeparate**, the elements of ζ are separate from **a**, in **zeta**.
Constraint: **wheret** must be **Nag_ElementsIn** or **Nag_ElementsSeparate**.

m

Input: n, the number of rows of A.
Constraint: **m** $\geq$ n.

n

Input: n, the number of columns of A.
Constraint: **n** ≥ 0.

ncolq

Input: *ncolq*, the required number of columns of Q.
When **ncolq** $= 0$ then an immediate return is effected.
Constraint: $0 \leq$ **ncolq** $\leq$ m.

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 ζ as described under the parameter **zeta** below.
Output: the first **ncolq** columns of the array **a** are overwritten by the first **ncolq** columns of the m by m orthogonal matrix Q. When **n** $= 0$ then the first **ncolq** columns of **a** are overwritten by the first **ncolq** columns of the identity matrix.

tda

Input: the second dimension of the array **a** as declared in the function from which nag_real_form_q is called.

Constraint: **tda** $\geq$ max(**n**,**ncolq**).

zeta[n]

Input: if **wheret** = **Nag_ElementsSeparate**, the array **zeta** must contain the elements of ζ. If **zeta**$[k-1] = 0.0$ then T_k is assumed to be I otherwise **zeta**$[k-1]$ is assumed to contain ζ_k. When **wheret** = **Nag_ElementsIn**, **zeta** is not referenced and may be set to the null pointer, i.e., (double $*$)0.

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 **wheret** had an illegal value.

NE_2_INT_ARG_LT

On entry, **m** = ⟨*value*⟩ while **n** = ⟨*value*⟩. These parameters must satisfy **m** $\geq$ **n**.

On entry, **tda** = ⟨*value*⟩ while max(**n,ncolq**) = ⟨*value*⟩. These parameters must satisfy **tda** $\geq$ max(**n,ncolq**).

NE_2_INT_ARG_GT

On entry, **ncolq** = ⟨*value*⟩ while **m** = ⟨*value*⟩. These parameters must satisfy **ncolq** $\leq$ **m**.

NE_INT_ARG_LT

On entry, **n** must not be less than 0: **n** = ⟨*value*⟩.

On entry, **ncolq** must not be less than 0: **ncolq** = ⟨*value*⟩.

NE_ALLOC_FAIL

Memory allocation failed.

6. Further Comments

The approximate number of floating-point operations required is given by

$$\begin{array}{ll} \frac{2}{3}n(3m-n)(2ncolq-n)-n(ncolq-n) & ncolq > n \\ \frac{2}{3}ncolq^2(3m-ncolq) & ncolq \leq n. \end{array}$$

6.1. Accuracy

The computed matrix Q satisfies the relation

$$Q = P + E$$

where P is an exactly orthogonal matrix and $\|E\| \leq c\epsilon$, ϵ is the ***machine precision***, c is a modest function of m and $\|.\|$ denotes the spectral (two) norm. See also Section 6.1 of nag_real_qr (f01qcc).

6.2. References

Golub G H and Van Loan C F (1989) *Matrix Computations* (2nd Edn) Johns Hopkins University Press, Baltimore.

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

7. See Also

nag_real_qr (f01qcc)

8. Example

To obtain the 5 by 5 orthogonal matrix Q following the QR factorization of the 5 by 3 matrix A given by

$$A = \begin{pmatrix} 2.0 & 2.5 & 2.5 \\ 2.0 & 2.5 & 2.5 \\ 1.6 & -0.4 & 2.8 \\ 2.0 & -0.5 & 0.5 \\ 1.2 & -0.3 & -2.9 \end{pmatrix}.$$

8.1. Program Text

```
/* nag_real_form_q(f01qec) 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

main()
{
  Integer tda = NMAX;
  Integer tdq = MMAX;

  double zeta[NMAX], a[MMAX][NMAX], q[MMAX][MMAX];
  Integer i, j, m, n, ncolq;

  Vprintf("f01qec Example Program Results\n");
  Vscanf(" %*[^\n]"); /* skip headings in data file */
  Vscanf(" %*[^\n]");

  Vscanf("%ld%ld", &m, &n);
  if (m > MMAX || n > NMAX)
    {
      Vprintf("m or n is out of range.\n");
      Vprintf("m = %2ld, n = %2ld\n", m, n);
    }
  else
    {
      Vscanf(" %*[^\n]");
      for (i = 0; i < m; ++i)   /* Read matrix data */
        for (j = 0; j < n; ++j)
          Vscanf("%lf", &a[i][j]);

      /* Find the QR factorization of A */
      f01qcc(m, n, (double *)a, tda, zeta, NAGERR_DEFAULT);

      /* Copy the array a into q and form the m by m matrix Q */
      for (j = 0; j < n; ++j)
        for (i = 0; i < m; ++i)
          q[i][j] = a[i][j];
      ncolq = m;
      f01qec(Nag_ElementsSeparate, m, n, ncolq, (double *)q, tdq,
             zeta, NAGERR_DEFAULT);

      Vprintf("Matrix  Q\n");
      for (i = 0; i < m; ++i)
        {
          for (j = 0; j < ncolq; ++j)
            Vprintf(" %8.4f", q[i][j]);
```

```
          Vprintf("\n");
        }
    }
  exit(EXIT_SUCCESS);
}
```

8.2. Program Data

```
f01qec Example Program Data
Values of m and n.
  5     3
Matrix A
  2.0   2.5   2.5
  2.0   2.5   2.5
  1.6  -0.4   2.8
  2.0  -0.5   0.5
  1.2  -0.3  -2.9
```

8.3. Program Results

```
f01qec Example Program Results
Matrix  Q
  -0.5000  -0.5000   0.0000  -0.5000  -0.5000
  -0.5000  -0.5000   0.0000   0.5000   0.5000
  -0.4000   0.4000  -0.6000  -0.4000   0.4000
  -0.5000   0.5000   0.0000   0.5000  -0.5000
  -0.3000   0.3000   0.8000  -0.3000   0.3000
```