NAG Library Function Document

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, const double zeta[],
     NagError *fail)
```

3 Description

Q is assumed to be given by

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

Q_k being given in the form

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

where

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

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

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 $\mathbf{a}[(k) \times \mathbf{tda} + k - 1], \ldots, \mathbf{a}[(m-1) \times \mathbf{tda} + k - 1]$ and ζ_k must be supplied either in $\mathbf{a}[(k-1) \times \mathbf{tda} + k - 1]$ or in $\mathbf{zeta}[k-1]$, depending upon the argument **wheret**.

4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

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

5 Arguments

1: **wheret** – Nag_WhereElements *Input*

On entry: 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** = Nag_ElementsIn or Nag_ElementsSeparate.

2: **m** – Integer *Input*

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

Constraint: **m** $\geq$ **n**.

3: **n** – Integer *Input*

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

Constraint: **n** ≥ 0.

4: **ncolq** – Integer *Input*

On entry: $ncolq$, the required number of columns of Q. When **ncolq** $= 0$ then an immediate return is effected.

Constraint: $0 \leq$ **ncolq** $\leq$ **m**.

5: **a**[**m** $\times$ **tda**] – double *Input/Output*

On entry: 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 argument **zeta**.

On exit: 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.

6: **tda** – Integer *Input*

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

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

7: **zeta**[**n**] – const double *Input*

On entry: 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 **NULL**.

8: **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_GT

On entry, **ncolq** $= \langle value \rangle$ while **m** $= \langle value \rangle$. These arguments must satisfy **ncolq** $\leq$ **m**.

NE_2_INT_ARG_LT

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

On entry, **tda** $= \langle value \rangle$ while $\max($**n**, **ncolq**$) = \langle value \rangle$. These arguments must satisfy **tda** $\geq \max($**n**, **ncolq**$)$.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument **wheret** had an illegal value.

NE_INT_ARG_LT

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

On entry, **ncolq** $= \langle value \rangle$.
Constraint: **ncolq** ≥ 0.

7 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 9.

8 Parallelism and Performance

Not applicable.

9 Further Comments

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

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

10 Example

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

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

10.1 Program Text

```
/* nag_real_form_q (f01qec) 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 A(I, J) a[(I) *tda + J]
#define Q(I, J) q[(I) *tdq + J]
int main(void)
{
  Integer  exit_status = 0, i, j, m, n, ncolq, tda, tdq;
  double   *a = 0, *q = 0, *zeta = 0;
  NagError fail;

  INIT_FAIL(fail);
```

```
  printf("nag_real_form_q (f01qec) Example Program Results\n");
  scanf(" %*[^\n]"); /* skip headings in data file */
  scanf(" %*[^\n]");
  scanf("%ld%ld", &m, &n);
  if (n >= 0 && m >= n)
    {
      if (!(zeta = NAG_ALLOC(n, double)) ||
          !(a = NAG_ALLOC(m*n, double)) ||
          !(q = NAG_ALLOC(m*m, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tda = n;
      tdq = m;
    }
  else
    {
      printf("Invalid n or m.\n");
      exit_status = 1;
      return exit_status;
    }
  scanf(" %*[^\n]");
  for (i = 0; i < m; ++i)   /* Read matrix data */
    for (j = 0; j < n; ++j)
      scanf("%lf", &A(i, j));

  /* Find the QR factorization of A */
  /* nag_real_qr (f01qcc).
   * QR factorization of real m by n matrix (m >= n)
   */
  nag_real_qr(m, n, a, tda, zeta, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_real_qr (f01qcc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* 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;
  /* nag_real_form_q (f01qec).
   * Form columns of Q after factorization by nag_real_qr
   * (f01qcc)
   */
  nag_real_form_q(Nag_ElementsSeparate, m, n, ncolq, q, tdq,
                  zeta, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_real_form_q (f01qec).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  printf("Matrix  Q\n");
  for (i = 0; i < m; ++i)
    {
      for (j = 0; j < ncolq; ++j)
        printf(" %8.4f", Q(i, j));
      printf("\n");
    }

 END:
  NAG_FREE(zeta);
  NAG_FREE(a);
  NAG_FREE(q);
```

```
  return exit_status;
}
```

10.2 Program Data

```
nag_real_form_q (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
```

10.3 Program Results

```
nag_real_form_q (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
```