NAG Library Function Document

nag_real_apply_q (f01qdc)

1 Purpose

nag_real_apply_q (f01qdc) performs one of the transformations

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

where B is an m by $ncolb$ real matrix and Q is an m by m orthogonal matrix, 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_apply_q (MatrixTranspose trans, Nag_WhereElements wheret,
     Integer m, Integer n, double a[], Integer tda, const double zeta[],
     Integer ncolb, double b[], Integer tdb, 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**.

To obtain Q explicitly B may be set to I and premultiplied by Q. This is more efficient than obtaining Q^{T}.

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: **trans** – MatrixTranspose *Input*

On entry: the operation to be performed as follows:

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

trans = Transpose or ConjugateTranspose, perform the operation $B := Q^{\mathrm{T}}B$.

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

2: **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.

3: **m** – Integer *Input*

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

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

4: **n** – Integer *Input*

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

When $\mathbf{n} = 0$ then an immediate return is effected.

Constraint: $\mathbf{n} \geq 0$.

5: **a**[**m** × **tda**] – double *Input*

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**. When **wheret** = Nag_ElementsSeparate, 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.

6: **tda** – Integer *Input*

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

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

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

On entry: if **wheret** = Nag_ElementsSeparate, the array **zeta** must contain the elements of ζ. If $\mathbf{zeta}[k-1] = 0.0$ then T_k is assumed to be I otherwise $\mathbf{zeta}[k-1]$ is assumed to contain ζ_k. When **wheret** = Nag_ElementsIn, **zeta** is not referenced and may be **NULL**

8: **ncolb** – Integer *Input*

On entry: $ncolb$, the number of columns of B.

When $\mathbf{ncolb} = 0$ then an immediate return is effected.

Constraint: $\mathbf{ncolb} \geq 0$.

9: **b**[**m** × **tdb**] – double *Input/Output*

Note: the (i,j)th element of the matrix B is stored in $\mathbf{b}[(i-1) \times \mathbf{tdb} + j - 1]$.

On entry: the leading m by $ncolb$ part of the array **b** must contain the matrix to be transformed.

On exit: **b** is overwritten by the transformed matrix.

10: **tdb** – Integer *Input*

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

Constraint: **tdb** $\geq$ **ncolb**.

11: **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, **m** = ⟨*value*⟩ while **n** = ⟨*value*⟩. These arguments must satisfy **m** $\geq$ **n**.

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

On entry, **tdb** = ⟨*value*⟩ while **ncolb** = ⟨*value*⟩. These arguments must satisfy **tdb** $\geq$ **ncolb**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

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

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

NE_INT_ARG_LT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

On entry, **ncolb** = ⟨*value*⟩.
Constraint: **ncolb** ≥ 0.

7 Accuracy

Letting C denote the computed matrix $Q^{\mathrm{T}}B$, C satisfies the relation

$$QC = B + E$$

where $\|E\| \leq c\epsilon\|B\|$, ϵ is the ***machine precision***, c is a modest function of m and . denotes the spectral (two) norm. An equivalent result holds for the computed matrix QB. See also Section 9.

8 Parallelism and Performance

Not applicable.

9 Further Comments

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

10 Example

To obtain the matrix $Q^{\mathrm{T}}B$ for the matrix B given by

$$B = \begin{pmatrix} 1.10 & 0.00 \\ 0.90 & 0.00 \\ 0.60 & 1.32 \\ 0.00 & 1.10 \\ -0.80 & -0.26 \end{pmatrix}$$

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_apply_q (f01qdc) 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 B(I, J) b[(I) *tdb + J]
int main(void)
{
  Integer  exit_status = 0, i, j, m, n, ncolb, tda, tdb;
  NagError fail;
  double   *a = 0, *b = 0, *zeta = 0;

  INIT_FAIL(fail);

  printf("nag_real_apply_q (f01qdc) Example Program Results\n");
  scanf(" %*[^\n]"); /* skip headings in data file */
  scanf(" %*[^\n]");
  scanf("%ld%ld", &m, &n);
  if (n > 0 && m >= n)
    {
      if (!(a = NAG_ALLOC(m*n, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tda = n;
    }
  else
    {
      printf("Invalid n or m.\n");
      exit_status = 1;
      return exit_status;
    }
  scanf(" %*[^\n]");
  for (i = 0; i < m; ++i)
    for (j = 0; j < n; ++j)
      scanf("%lf", &A(i, j));
  scanf(" %*[^\n]");
  scanf("%ld", &ncolb);
```

```
  if (ncolb >= 0)
    {
      if (!(zeta = NAG_ALLOC(n, double)) ||
          !(b = NAG_ALLOC(m*ncolb, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tdb = ncolb;
    }
  else
    {
      printf("Invalid ncolb.\n");
      exit_status = 1;
      return exit_status;
    }
  scanf(" %*[^\n]");
  for (i = 0; i < m; ++i)
    for (j = 0; j < ncolb; ++j)
      scanf("%lf", &B(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;
    }

  /* Form  Q'*B */
  /* nag_real_apply_q (f01qdc).
   * Compute QB or Q^TB after factorization by nag_real_qr
   * (f01qcc)
   */
  nag_real_apply_q(Transpose, Nag_ElementsSeparate, m, n, a, tda, zeta,
                   ncolb, b, tdb, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_real_apply_q (f01qdc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }


  printf("Matrix  Q'*B\n");
  for (i = 0; i < m; ++i)
    {
      for (j = 0; j < ncolb; ++j)
        printf(" %8.4f", B(i, j));
      printf("\n");
    }
 END:
  NAG_FREE(a);
  NAG_FREE(zeta);
  NAG_FREE(b);
  return exit_status;
}
```

10.2 Program Data

```
nag_real_apply_q (f01qdc) 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
Value of ncolb
  2
Matrix B
  1.1   0.0
  0.9   0.0
  0.6   1.32
  0.0   1.1
 -0.8  -0.26
```

10.3 Program Results

```
nag_real_apply_q (f01qdc) Example Program Results
Matrix  Q'*B
  -1.0000  -1.0000
  -1.0000   1.0000
  -1.0000  -1.0000
  -0.1000   0.1000
  -0.1000  -0.1000
```