

NAG C Library Function Document

nag_dopgtr (f08gfc)

1 Purpose

nag_dopgtr (f08gfc) generates the real orthogonal matrix Q , which was determined by nag_dsptd (f08gec) when reducing a symmetric matrix to tridiagonal form.

2 Specification

```
void nag_dopgtr (Nag_OrderType order, Nag_UploType uplo, Integer n,
                const double ap[], const double tau[], double q[], Integer pdq, NagError *fail)
```

3 Description

nag_dopgtr (f08gfc) is intended to be used after a call to nag_dsptd (f08gec), which reduces a real symmetric matrix A to symmetric tridiagonal form T by an orthogonal similarity transformation: $A = QTQ^T$. nag_dsptd (f08gec) represents the orthogonal matrix Q as a product of $n - 1$ elementary reflectors.

This function may be used to generate Q explicitly as a square matrix.

4 References

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

5 Parameters

- 1: **order** – Nag_OrderType *Input*
On entry: the **order** parameter specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order = Nag_RowMajor**. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.
Constraint: **order = Nag_RowMajor** or **Nag_ColMajor**.
- 2: **uplo** – Nag_UploType *Input*
On entry: this **must** be the same parameter **uplo** as supplied to nag_dsptd (f08gec).
Constraint: **uplo = Nag_Upper** or **Nag_Lower**.
- 3: **n** – Integer *Input*
On entry: n , the order of the matrix Q .
Constraint: $n \geq 0$.
- 4: **ap**[*dim*] – const double *Input*
Note: the dimension, *dim*, of the array **ap** must be at least $\max(1, n \times (n + 1)/2)$.
On entry: details of the vectors which define the elementary reflectors, as returned by nag_dsptd (f08gec).
- 5: **tau**[*dim*] – const double *Input*
Note: the dimension, *dim*, of the array **tau** must be at least $\max(1, n - 1)$.

On entry: further details of the elementary reflectors, as returned by nag_dsprtd (f08gec).

6: **q**[*dim*] – double *Output*

Note: the dimension, *dim*, of the array **q** must be at least $\max(1, \mathbf{pdq} \times \mathbf{n})$.

If **order** = **Nag_ColMajor**, the (*i*, *j*)th element of the matrix *Q* is stored in **q**[(*j* – 1) × **pdq** + *i* – 1] and if **order** = **Nag_RowMajor**, the (*i*, *j*)th element of the matrix *Q* is stored in **q**[(*i* – 1) × **pdq** + *j* – 1].

On exit: the *n* by *n* orthogonal matrix *Q*.

7: **pdq** – Integer *Input*

On entry: the stride separating matrix row or column elements (depending on the value of **order**) in the array **q**.

Constraint: **pdq** ≥ $\max(1, \mathbf{n})$.

8: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = *⟨value⟩*.

Constraint: **n** ≥ 0.

On entry, **pdq** = *⟨value⟩*.

Constraint: **pdq** > 0.

NE_INT_2

On entry, **pdq** = *⟨value⟩*, **n** = *⟨value⟩*.

Constraint: **pdq** ≥ $\max(1, \mathbf{n})$.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter *⟨value⟩* had an illegal value.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

7 Accuracy

The computed matrix *Q* differs from an exactly orthogonal matrix by a matrix *E* such that

$$\|E\|_2 = O(\epsilon),$$

where ϵ is the *machine precision*.

8 Further Comments

The total number of floating-point operations is approximately $\frac{4}{3}n^3$.

The complex analogue of this function is nag_zupgtr (f08gtc).

9 Example

To compute all the eigenvalues and eigenvectors of the matrix A , where

$$A = \begin{pmatrix} 2.07 & 3.87 & 4.20 & -1.15 \\ 3.87 & -0.21 & 1.87 & 0.63 \\ 4.20 & 1.87 & 1.15 & 2.06 \\ -1.15 & 0.63 & 2.06 & -1.81 \end{pmatrix},$$

using packed storage. Here A is symmetric and must first be reduced to tridiagonal form by `nag_dsprtd` (`f08gec`). The program then calls `nag_dopgtr` (`f08gfc`) to form Q , and passes this matrix to `nag_dsteqr` (`f08jec`) which computes the eigenvalues and eigenvectors of A .

9.1 Program Text

```

/* nag_dopgtr (f08gfc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer ap_len, i, j, n, pdz, d_len, e_len, tau_len;
    Integer exit_status=0;
    NagError fail;
    Nag_UploType uplo;
    Nag_OrderType order;
    /* Arrays */
    char uplo_char[2];
    double *ap=0, *d=0, *e=0, *tau=0, *z=0;

#ifdef NAG_COLUMN_MAJOR
#define A_UPPER(I,J) ap[J*(J-1)/2 + I - 1]
#define A_LOWER(I,J) ap[(2*n-J)*(J-1)/2 + I - 1]
    order = Nag_ColMajor;
#else
#define A_LOWER(I,J) ap[I*(I-1)/2 + J - 1]
#define A_UPPER(I,J) ap[(2*n-I)*(I-1)/2 + J - 1]
    order = Nag_RowMajor;
#endif

    INIT_FAIL(fail);
    Vprintf("f08gfc Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[\n] ");
    Vscanf("%ld%*[\n] ", &n);
#ifdef NAG_COLUMN_MAJOR
    pdz = n;
#else
    pdz = n;
#endif
    ap_len = n*(n+1)/2;
    tau_len = n-1;
    d_len = n;
    e_len = n-1;
    /* Allocate memory */
    if ( !(ap = NAG_ALLOC(ap_len, double)) ||
        !(d = NAG_ALLOC(d_len, double)) ||
        !(e = NAG_ALLOC(e_len, double)) ||

```

```

        !(tau = NAG_ALLOC(tau_len, double)) ||
        !(z = NAG_ALLOC(n * n, double)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

/* Read A from data file */
Vscanf(" ' %1s '%*[\n] ", uplo_char);
if (*(unsigned char *)uplo_char == 'L')
    uplo = Nag_Lower;
else if (*(unsigned char *)uplo_char == 'U')
    uplo = Nag_Upper;
else
    {
        Vprintf("Unrecognised character for Nag_UploType type\n");
        exit_status = -1;
        goto END;
    }
if (uplo == Nag_Upper)
    {
        for (i = 1; i <= n; ++i)
            {
                for (j = i; j <= n; ++j)
                    Vscanf("%lf", &A_UPPER(i,j));
            }
        Vscanf("%*[\n] ");
    }
else
    {
        for (i = 1; i <= n; ++i)
            {
                for (j = 1; j <= i; ++j)
                    Vscanf("%lf", &A_LOWER(i,j));
            }
        Vscanf("%*[\n] ");
    }

/* Reduce A to tridiagonal form T = (Q**T)*A*Q */
f08gec(order, uplo, n, ap, d, e, tau, &fail);
if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f08gec.\n%s\n", fail.message);
        exit_status = 1;
    }

/* Form Q explicitly, storing the result in Z */
f08gfc(order, uplo, n, ap, tau, z, pdz, &fail);
if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f08gfc.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

/* Calculate all the eigenvalues and eigenvectors of A */
f08jec(order, Nag_UpdateZ, n, d, e, z, pdz, &fail);
if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f08jec.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

/* Print eigenvalues and eigenvectors */
Vprintf("Eigenvalues\n");
for (i = 1; i <= n; ++i)
    Vprintf("%8.4f%s", d[i-1], i%8==0 ? "\n": " ");
Vprintf("\n\n");
x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n, z, pdz,
        "Eigenvectors", 0, &fail);

```

```

if (fail.code != NE_NOERROR)
{
  Vprintf("Error from x04cac.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}
END:
if (ap) NAG_FREE(ap);
if (d) NAG_FREE(d);
if (e) NAG_FREE(e);
if (tau) NAG_FREE(tau);
if (z) NAG_FREE(z);

return exit_status;
}

```

9.2 Program Data

```

f08gfc Example Program Data
  4                               :Value of N
  'L'                             :Value of UPLO
  2.07
  3.87  -0.21
  4.20  1.87  1.15
 -1.15  0.63  2.06  -1.81  :End of matrix A

```

9.3 Program Results

f08gfc Example Program Results

```

Eigenvalues
-5.0034  -1.9987   0.2013   8.0008

```

```

Eigenvectors
      1      2      3      4
1  0.5658  -0.2328  -0.3965  0.6845
2  -0.3478   0.7994  -0.1780  0.4564
3  -0.4740  -0.4087   0.5381  0.5645
4   0.5781   0.3737   0.7221  0.0676

```
