NAG Library Function Document
nag_dopgtr (f08gfc)

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

2 Specification
#include <nag.h>
#include <nagf08.h>
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_dsptrd (f08gec), which reduces a real symmetric matrix $A$ to symmetric tridiagonal form $T$ by an orthogonal similarity transformation: $A = QTQ^T$. nag_dsptrd (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

5 Arguments
1: order – Nag_OrderType
   On entry: the order argument 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 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.
   Constraint: order = Nag_RowMajor or Nag_ColMajor.

2: uplo – Nag_UploType
   On entry: this must be the same argument uplo as supplied to nag_dsptrd (f08gec).
   Constraint: uplo = Nag_Upper or Nag_Lower.

3: n – Integer
   On entry: $n$, the order of the matrix $Q$.
   Constraint: $n \geq 0$.

4: ap[dim] – const double
   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_dsptrd (f08gec).

5:  \texttt{tau[dim]} – const double

\textbf{Input}

Note: the dimension, \texttt{dim}, of the array \texttt{tau} must be at least \texttt{max(1,n-1)}.

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

6:  \texttt{q[dim]} – double

\textbf{Output}

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

The \((i,j)\)th element of the matrix \(Q\) is stored in

\[
q[j-1] \times \texttt{pdq} + i - 1 \quad \text{when } \texttt{order} = \text{Nag\_ColMajor};
\]

\[
q[i-1] \times \texttt{pdq} + j - 1 \quad \text{when } \texttt{order} = \text{Nag\_RowMajor}.
\]

On exit: the \(n\) by \(n\) orthogonal matrix \(Q\).

7:  \texttt{pdq} – Integer

\textbf{Input}

On entry: the stride separating row or column elements (depending on the value of \texttt{order}) in the array \texttt{q}.

Constraint: \texttt{pdq} \(\geq\) \texttt{max(1,n)}.

8:  \texttt{fail} – NagError*

\textbf{Input/Output}

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 \hspace{1cm} \textbf{Error Indicators and Warnings}

\textbf{NE_ALLOC_FAIL}

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}

On entry, argument \langle\texttt{value}\rangle had an illegal value.

\textbf{NE_INT}

On entry, \texttt{n} = \langle\texttt{value}\rangle.

Constraint: \texttt{n} \(\geq\) 0.

On entry, \texttt{pdq} = \langle\texttt{value}\rangle.

Constraint: \texttt{pdq} \(>\) 0.

\textbf{NE_INT_2}

On entry, \texttt{pdq} = \langle\texttt{value}\rangle and \texttt{n} = \langle\texttt{value}\rangle.

Constraint: \texttt{pdq} \(\geq\) \texttt{max(1,n)}.

\textbf{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 contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.

See Section 3.6.6 in the Essential Introduction for further information.
7 Accuracy

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

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

where $\epsilon$ is the machine precision.

8 Parallelism and Performance

nag_dopgtr (f08gfc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dopgtr (f08gfc) makes calls to BLAS and/or LAPACK routines, which may be threading within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 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).

10 Example

This example computes 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_dsptrd (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 $A$.

10.1 Program Text

```c
/* nag_dopgtr (f08gfc) Example Program. *
 * Copyright 2014 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;
```

Mark 25

f08gfc.3
Integer exit_status = 0;
NagError fail;
Nag_UploType uplo;
Nag_OrderType order;
/* Arrays */
char nag_enum_arg[40];
double *ap = 0, *d = 0, *e = 0, *tau = 0, *z = 0;
#endif
#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]
#define Z(I, J) z[(J - 1) * pdz + I - 1]
#endif
#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]
#define Z(I, J) z[(I - 1) * pdz + J - 1]
#endif
INIT_FAIL(fail);
printf("nag_dopgtr (f08gfc) Example Program Results\n\n");
#ifdef _WIN32
scanf_s("%*[^
"]");
#else
scanf("%*[^
"]");
#endif
#ifdef _WIN32
scanf_s("%"NAG_IFMT"%*[^
"]", &n);
#else
scanf("%"NAG_IFMT"%*[^
"]", &n);
#endif
#define 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]
#define Z(I, J) z[(J - 1) * pdz + I - 1]
#endif
#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]
#define Z(I, J) z[(I - 1) * pdz + J - 1]
#endif
INIT_FAIL(fail);
printf("nag_dopgtr (f08gfc) Example Program Results\n\n");
#ifdef _WIN32
scanf_s("%*[^
"]");
#else
scanf("%*[^
"]");
#endif
#ifdef _WIN32
scanf_s("%"NAG_IFMT"%*[^
"]", &n);
#else
scanf("%"NAG_IFMT"%*[^
"]", &n);
#endif
#ifdef NAG_COLUMN_MAJOR
pdz = n;
#else
pdz = n;
#endif
/* 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)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
/* Read A from data file */
#ifdef _WIN32
 scanf_s("%39s%*[^
"]", nag_enum_arg, _countof(nag_enum_arg));
#else
 scanf("%39s%*[^
"]", nag_enum_arg);
#endif
#define nag_enum_name_to_value (x04nac)
 * Converts NAG enum member name to value
 * uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
 if (uplo == Nag_Upper)
 {
     for (i = 1; i <= n; ++i)
     {
         for (j = i; j <= n; ++j)
```c

#ifdef _WIN32
    scanf_s("%lf", &A_UPPER(i, j));
#else
    scanf("%lf", &A_UPPER(i, j));
#endif

#endif _WIN32

else
    for (i = 1; i <= n; ++i)
        for (j = 1; j <= i; ++j)
            #ifdef _WIN32
                scanf_s("%lf", &A_LOWER(i, j));
            #else
                scanf("%lf", &A_LOWER(i, j));
            #endif

#endif _WIN32

else
    scanf("%*\n ");
#endif

else
    printf("Error from nag_dsptrd (f08gec).\n%s\n", fail.message);
    exit_status = 1;
#endif

/* Calculate all the eigenvalues and eigenvectors of A */
/* nag_dsteqr (f08jec).
* All eigenvalues and eigenvectors of real symmetric
* tridiagonal matrix, reduced from real symmetric matrix
* using implicit QL or QR
*/
    nag_dsteqr(order, Nag_UpdateZ, n, d, e, pdz, &fail);
    if (fail.code != NE_NOERROR)
        printf("Error from nag_dsteqr (f08jec).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
#endif

/* Normalize the eigenvectors */
    for(j=1; j<=n; j++)
        
```
for(i=n; i>=1; i--)
{
    Z(i, j) = Z(i, j) / Z(1,j);
}

/* Print eigenvalues and eigenvectors */
printf("Eigenvalues\n");
for (i = 1; i <= n; ++i)
    printf("%8.4f%s", d[i-1], i%8 == 0?"\n": "");
printf("\n\n");
/* nag_gen_real_mat_print (x04cac).
* Print real general matrix (easy-to-use)
*/
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n, z, 
    pdz, "Eigenvectors", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(ap);
NAG_FREE(d);
NAG_FREE(e);
NAG_FREE(tau);
NAG_FREE(z);
return exit_status;

10.2 Program Data

nag_dopgtr (f08gfc) Example Program Data
4 :Value of n
Nag_Lower :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

10.3 Program Results

nag_dopgtr (f08gfc) Example Program Results

Eigenvalues
-5.0034 -1.9987 0.2013 8.0008

Eigenvectors
1 2 3 4
1 1.0000 1.0000 1.0000 1.0000
2 -0.6148 -3.4333 0.4489 0.6668
3 -0.8378 1.7553 -1.3572 0.8248
4 1.0219 -1.6052 -1.8213 0.0988

f08gfc
NAG Library Manual
f08gfc.6 (last) Mark 25