

NAG C Library Function Document

nag_zupgtr (f08gtc)

1 Purpose

nag_zupgtr (f08gtc) generates the complex unitary matrix Q , which was determined by nag_zhptrd (f08gsc) when reducing a Hermitian matrix to tridiagonal form.

2 Specification

```
#include <nag.h>
#include <nagf08.h>

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

3 Description

nag_zupgtr (f08gtc) is intended to be used after a call to nag_zhptrd (f08gsc), which reduces a complex Hermitian matrix A to real symmetric tridiagonal form T by a unitary similarity transformation: $A = QTQ^H$. nag_zhptrd (f08gsc) represents the unitary 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 Arguments

- 1: **order** – Nag_OrderType *Input*
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 2.2.1.4 of 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 *Input*
On entry: this **must** be the same argument **uplo** as supplied to nag_zhptrd (f08gsc).
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 Complex *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_zhptrd (f08gsc).

- 5: **tau**[*dim*] – const Complex *Input*
Note: the dimension, *dim*, of the array **tau** must be at least $\max(1, \mathbf{n} - 1)$.
On entry: further details of the elementary reflectors, as returned by nag_zhptrd (f08gsc).
- 6: **q**[*dim*] – Complex *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].
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* unitary 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 * *Input/Output*
The NAG error argument (see Section 2.6 of the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

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

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_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 unitary matrix by a matrix *E* such that

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

where ϵ is the *machine precision*.

8 Further Comments

The total number of real floating-point operations is approximately $\frac{16}{3}n^3$.

The real analogue of this function is nag_dopgtr (f08gfc).

9 Example

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

$$A = \begin{pmatrix} -2.28 + 0.00i & 1.78 - 2.03i & 2.26 + 0.10i & -0.12 + 2.53i \\ 1.78 + 2.03i & -1.12 + 0.00i & 0.01 + 0.43i & -1.07 + 0.86i \\ 2.26 - 0.10i & 0.01 - 0.43i & -0.37 + 0.00i & 2.31 - 0.92i \\ -0.12 - 2.53i & -1.07 - 0.86i & 2.31 + 0.92i & -0.73 + 0.00i \end{pmatrix},$$

using packed storage. Here A is Hermitian and must first be reduced to tridiagonal form by nag_zhptrd (f08gsc). The program then calls nag_zupgtr (f08gtc) to form Q , and passes this matrix to nag_zsteqr (f08jsc) which computes the eigenvalues and eigenvectors of A .

9.1 Program Text

```
/* nag_zupgtr (f08gtc) 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];
    Complex *ap=0, *tau=0, *z=0;
    double *d=0, *e=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("nag_zupgtr (f08gtc) 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, Complex)) ||
    !(d = NAG_ALLOC(d_len, double)) ||
    !(e = NAG_ALLOC(e_len, double)) ||
    !(tau = NAG_ALLOC(tau_len, Complex)) ||
    !(z = NAG_ALLOC(n * n, Complex)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read A from data file */
Vscanf(" ' %ls '%*[^\\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 , %lf )", &A_UPPER(i,j).re,
                &A_UPPER(i,j).im);
        }
        Vscanf("%*[^\\n] ");
    }
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
        {
            Vscanf(" ( %lf , %lf )", &A_LOWER(i,j).re,
                &A_LOWER(i,j).im);
        }
        Vscanf("%*[^\\n] ");
    }
}

/* Reduce A to tridiagonal form T = (Q**H)*A*Q */
/* nag_zhptrd (f08gsc).
 * Unitary reduction of complex Hermitian matrix to real
 * symmetric tridiagonal form, packed storage
 */
nag_zhptrd(order, uplo, n, ap, d, e, tau, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from nag_zhptrd (f08gsc).\n%s\n", fail.message);
    exit_status = 1;
}

/* Form Q explicitly, storing the result in Z */
/* nag_zupgtr (f08gtc).
 * Generate unitary transformation matrix from reduction to
 * tridiagonal form determined by nag_zhptrd (f08gsc)
 */
nag_zupgtr(order, uplo, n, ap, tau, z, pdz, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from nag_zupgtr (f08gtc).\n%s\n", fail.message);
    exit_status = 1;
}

```

```

        goto END;
    }

/* Calculate all the eigenvalues and eigenvectors of A */
/* nag_zsteqr (f08jsc).
 * All eigenvalues and eigenvectors of real symmetric
 * tridiagonal matrix, reduced from complex Hermitian
 * matrix, using implicit QL or QR
 */
nag_zsteqr(order, Nag_UpdateZ, n, d, e, z, pdz, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from nag_zsteqr (f08jsc).\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");
/* nag_gen_complx_mat_print_comp (x04dbc).
 * Print complex general matrix (comprehensive)
 */
nag_gen_complx_mat_print_comp(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n,
                              z, pdz, Nag_BracketForm, "%7.4f",
                              "Eigenvectors", Nag_IntegerLabels, 0,
                              Nag_IntegerLabels, 0, 80, 0, 0, &fail);

if (fail.code != NE_NOERROR)
{
    Vprintf("Error from nag_gen_complx_mat_print_comp (x04dbc).\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

```

nag_zupgtr (f08gtc) Example Program Data
4                                     :Value of N
'L'                                 :Value of UPLO
(-2.28, 0.00)
( 1.78, 2.03) (-1.12, 0.00)
( 2.26,-0.10) ( 0.01,-0.43) (-0.37, 0.00)
(-0.12,-2.53) (-1.07,-0.86) ( 2.31, 0.92) (-0.73, 0.00) :End of matrix A

```

9.3 Program Results

nag_zupgtr (f08gtc) Example Program Results

```

Eigenvalues
-6.0002                -3.0030                0.5036                3.9996

Eigenvectors

      1      2      3      4
1 ( 0.7299, 0.0000) (-0.2120, 0.1497) ( 0.1000,-0.3570) ( 0.1991, 0.4720)
2 (-0.1663,-0.2061) ( 0.7307, 0.0000) ( 0.2863,-0.3353) (-0.2467, 0.3751)
3 (-0.4165,-0.1417) (-0.3291, 0.0479) ( 0.6890, 0.0000) ( 0.4468, 0.1466)
4 ( 0.1743, 0.4162) ( 0.5200, 0.1329) ( 0.0662, 0.4347) ( 0.5612, 0.0000)

```