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

```c
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


5 Parameters

1: order – Nag_OrderType

   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

   On entry: this must be the same parameter uplo as supplied to nag_zhptrd (f08gsc).

   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 Complex

   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: \( \text{tau}[\text{dim}] \) – const Complex \hspace{1cm} \text{Input}

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

On entry: further details of the elementary reflectors, as returned by nag_zhptrd (f08gsc).

6: \( q[\text{dim}] \) – Complex \hspace{1cm} \text{Output}

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

If \( \text{order} = \text{Nag_ColMajor} \), the \( (i,j) \)th element of the matrix \( Q \) is stored in \( q[(j - 1) \times \text{pdq} + i - 1] \) and if \( \text{order} = \text{Nag_RowMajor} \), the \( (i,j) \)th element of the matrix \( Q \) is stored in \( q[(i - 1) \times \text{pdq} + j - 1] \).

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

7: \( \text{pdq} \) – Integer \hspace{1cm} \text{Input}

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

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

8: \( \text{fail} \) – NagError * \hspace{1cm} \text{Output}

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

**NE_INT**

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 0 \).

On entry, \( \text{pdq} = \langle \text{value} \rangle \).

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

**NE_INT_2**

On entry, \( \text{pdq} = \langle \text{value} \rangle \), \( n = \langle \text{value} \rangle \).

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

**NE_ALLOC_FAIL**

Memory allocation failed.

**NE_BAD_PARAM**

On entry, parameter \( \langle \text{value} \rangle \) 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 unitary matrix by a matrix \( E \) such that

\[ ||E||_2 = O(\epsilon), \]

where \( \epsilon \) is the \textit{machine precision}. 

f08gtc.2 [NP3645/7]
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 (f08fc).

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>

#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
        ap_len = n;
    #else
        ap_len = n;
    #endif
    INIT_FAIL(fail);
    Vprintf("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 */
f08gsc(order, uplo, n, ap, d, e, tau, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08gsc.\n", fail.message);
    exit_status = 1;
}
/* Form Q explicitly, storing the result in Z */
f08gtc(order, uplo, n, ap, tau, z, pdz, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08gtc.\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Calculate all the eigenvalues and eigenvectors of A */
f08jsc(order, Nag_UpdateZ, n, d, e, z, pdz, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08jsc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print eigenvalues and eigenvectors */
Vprintf("Eigenvalues\n");
for (i = 1; i <= n; ++i)
    Vprintf("%8.4f\n", d[i-1], i%8==0 ?"\n": "");
Vprintf("\n\n");
x04dbc(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 x04dbc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}

END:
if (ap) NAG_FREE(ap);
if (d) NAG_FREE(d);
nag_free(e);
nag_free(tau);
if (z) NAG_FREE(z);
return exit_status;

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

9.3 Program Results
f08gtc Example Program Results

Eigenvalues
-6.0002 -3.0030  0.5036  3.9996

Eigenvectors

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Value</th>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.7299</td>
<td>0.0000</td>
<td>-0.2120</td>
<td>0.1497</td>
</tr>
<tr>
<td>2</td>
<td>-0.1663</td>
<td>-0.2061</td>
<td>0.7307</td>
<td>0.0000</td>
</tr>
<tr>
<td>3</td>
<td>-0.4165</td>
<td>-0.1417</td>
<td>-0.3291</td>
<td>0.0479</td>
</tr>
<tr>
<td>4</td>
<td>0.1743</td>
<td>0.4162</td>
<td>0.5200</td>
<td>0.1329</td>
</tr>
</tbody>
</table>