NAG Library Function Document

nag_zungtr (f08ftc)

1 Purpose

nag_zungtr (f08ftc) generates the complex unitary matrix Q, which was determined by nag_zhetrd (f08fsc) when reducing a Hermitian matrix to tridiagonal form.

2 Specification

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

```
void nag_zungtr (Nag_OrderType order, Nag_UploType uplo, Integer n,
     Complex a[], Integer pda, const Complex tau[], NagError *fail)
```

3 Description

nag_zungtr (f08ftc) is intended to be used after a call to nag_zhetrd (f08fsc), which reduces a complex Hermitian matrix A to real symmetric tridiagonal form T by a unitary similarity transformation: $A = QTQ^{\mathrm{H}}$. nag_zhetrd (f08fsc) 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 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 *Input*

On entry: this **must** be the same argument **uplo** as supplied to nag_zhetrd (f08fsc).

Constraint: **uplo** = Nag_Upper or Nag_Lower.

3: **n** – Integer *Input*

On entry: n, the order of the matrix Q.

Constraint: **n** ≥ 0.

4: **a**[*dim*] – Complex *Input/Output*

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

On entry: details of the vectors which define the elementary reflectors, as returned by nag_zhetrd (f08fsc).

On exit: the n by n unitary matrix Q.

If **order** = 'Nag_ColMajor', the (i, j)th element of the matrix is stored in $\mathbf{a}[(j-1) \times \mathbf{pda} + i - 1]$.

If **order** = 'Nag_RowMajor', the (i, j)th element of the matrix is stored in $\mathbf{a}[(i-1) \times \mathbf{pda} + j - 1]$.

5: **pda** – Integer *Input*

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

Constraint: $\mathbf{pda} \geq \max(1, \mathbf{n})$.

6: **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_zhetrd (f08fsc).

7: **fail** – NagError * *Input/Output*

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

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_INT

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 0$.

On entry, $\mathbf{pda} = \langle value \rangle$.
Constraint: $\mathbf{pda} > 0$.

NE_INT_2

On entry, $\mathbf{pda} = \langle value \rangle$ and $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{pda} \geq \max(1, \mathbf{n})$.

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 contact 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 Parallelism and Performance

nag_zungtr (f08ftc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_zungtr (f08ftc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the Users' Note for your implementation for any additional implementation-specific information.

9 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_dorgtr (f08ffc).

10 Example

This example computes all the eigenvalues and eigenvectors of the matrix A, where

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

Here A is Hermitian and must first be reduced to tridiagonal form by nag_zhetrd (f08fsc). The program then calls nag_zungtr (f08ftc) to form Q, and passes this matrix to nag_zsteqr (f08jsc) which computes the eigenvalues and eigenvectors of A.

10.1 Program Text

```
/* nag_zungtr (f08ftc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  Integer       i, j, n, pda, pdz, d_len, e_len, tau_len;
  Integer       exit_status = 0;
  NagError      fail;
  Nag_UploType  uplo;
  Nag_OrderType order;
  /* Arrays */
  char          nag_enum_arg[40];
  Complex       *a = 0, *tau = 0, *z = 0;
  double        *d = 0, *e = 0;

#ifdef NAG_COLUMN_MAJOR
#define A(I, J) a[(J - 1) * pda + I - 1]
#define Z(I, J) z[(J - 1) * pdz + I - 1]
  order = Nag_ColMajor;
#else
#define A(I, J) a[(I - 1) * pda + J - 1]
#define Z(I, J) z[(I - 1) * pdz + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);
```

```
  printf("nag_zungtr (f08ftc) Example Program Results\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%ld%*[^\n] ", &n);
#ifdef NAG_COLUMN_MAJOR
  pda = n;
  pdz = n;
#else
  pda = n;
  pdz = n;
#endif

  tau_len = n-1;
  d_len = n;
  e_len = n-1;
  /* Allocate memory */
  if (!(a = NAG_ALLOC(n * n, Complex)) ||
      !(tau = NAG_ALLOC(tau_len, Complex)) ||
      !(z = NAG_ALLOC(n * n, Complex)) ||
      !(d = NAG_ALLOC(d_len, double)) ||
      !(e = NAG_ALLOC(e_len, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read A from data file */
  scanf("%39s%*[^\n] ", nag_enum_arg);
  /* 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)
            scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
        }
      scanf("%*[^\n] ");
    }
  else
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = 1; j <= i; ++j)
            scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
        }
      scanf("%*[^\n] ");
    }

  /* Reduce A to tridiagonal form T = (Q**H)*A*Q */
  /* nag_zhetrd (f08fsc).
   * Unitary reduction of complex Hermitian matrix to real
   * symmetric tridiagonal form
   */
  nag_zhetrd(order, uplo, n, a, pda, d, e, tau, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zhetrd (f08fsc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Copy A into Z */
  if (uplo == Nag_Upper)
    {
      for (i = 1; i <= n; ++i)
```

```
      {
        for (j = i; j <= n; ++j)
          {
            Z(i, j).re = A(i, j).re;
            Z(i, j).im = A(i, j).im;
          }
      }
  }
else
  {
    for (i = 1; i <= n; ++i)
      {
        for (j = 1; j <= i; ++j)
          {
            Z(i, j).re = A(i, j).re;
            Z(i, j).im = A(i, j).im;
          }
      }
  }
/* Form Q explicitly, storing the result in Z */
/* nag_zungtr (f08ftc).
 * Generate unitary transformation matrix from reduction to
 * tridiagonal form determined by nag_zhetrd (f08fsc)
 */
nag_zungtr(order, uplo, n, z, pdz, tau, &fail);
if (fail.code != NE_NOERROR)
  {
    printf("Error from nag_zungtr (f08ftc).\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)
  {
    printf("Error from nag_zsteqr (f08jsc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
/* Normalize the eigenvectors */
for(j=1; j<=n; j++)
  {
    for(i=n; i>=1; i--)
      {
        Z(i, j) = nag_complex_divide(Z(i, j),Z(1, j));
      }
  }
/* Print eigenvalues and eigenvectors */
printf("\nEigenvalues\n");
for (i = 1; i <= n; ++i)
  printf("%9.4f%s", d[i-1], i%4 == 0?"\n":" ");
printf("\n");
/* nag_gen_complx_mat_print_comp (x04dbc).
 * Print complex general matrix (comprehensive)
 */
fflush(stdout);
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)
  {
    printf(
           "Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n",
```

```
            fail.message);
      exit_status = 1;
      goto END;
    }
 END:
  NAG_FREE(a);
  NAG_FREE(tau);
  NAG_FREE(z);
  NAG_FREE(d);
  NAG_FREE(e);

  return exit_status;
}
```

10.2 Program Data

```
nag_zungtr (f08ftc) Example Program Data
  4                                                          :Value of n
  Nag_Lower                                                  :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
```

10.3 Program Results

```
nag_zungtr (f08ftc) Example Program Results

Eigenvalues
  -6.0002   -3.0030    0.5036    3.9996

 Eigenvectors
                    1                   2                   3                   4
 1  ( 1.0000, 0.0000)  ( 1.0000,-0.0000)  ( 1.0000,-0.0000)  ( 1.0000, 0.0000)
 2  (-0.2278,-0.2824)  (-2.2999,-1.6237)  ( 1.0792, 0.4997)  ( 0.4876, 0.7282)
 3  (-0.5706,-0.1941)  ( 1.1424, 0.5807)  ( 0.5013, 1.7896)  ( 0.6025,-0.6924)
 4  ( 0.2388, 0.5702)  (-1.3415,-1.5739)  (-1.0810, 0.4883)  ( 0.4257,-1.0093)
```