NAG C Library Function Document

nag_zhetrd (f08fsc)

1 Purpose

nag_zhetrd (f08fsc) reduces a complex Hermitian matrix to tridiagonal form.

2 Specification

```
void nag_zhetrd (Nag_OrderType order, Nag_UploType uplo, Integer n, Complex a[],
      Integer pda, double d[], double e[], Complex tau[], NagError *fail)
```

3 Description

nag_zhetrd (f08fsc) reduces a complex Hermitian matrix A to real symmetric tridiagonal form T by a unitary similarity transformation: $A = QTQ^H$.

The matrix Q is not formed explicitly but is represented as a product of $n-1$ elementary reflectors (see the f08 Chapter Introduction for details). Functions are provided to work with Q in this representation (see Section 8).

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: indicates whether the upper or lower triangular part of A is stored as follows:

if **uplo** = **Nag_Upper**, the upper triangular part of A is stored;

if **uplo** = **Nag_Lower**, the lower triangular part of A is stored.

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

3: **n** – Integer *Input*

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

Constraint: **n** ≥ 0.

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

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

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

On entry: the n by n Hermitian matrix A. If **uplo** $=$ **Nag_Upper**, the upper triangle of A must be stored and the elements of the array below the diagonal are not referenced; if **uplo** $=$ **Nag_Lower**, the lower triangle of A must be stored and the elements of the array above the diagonal are not referenced.

On exit: **a** is overwritten by the tridiagonal matrix T and details of the unitary matrix Q as specified by **uplo**.

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

6: **d**[dim] – double *Output*

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

On exit: the diagonal elements of the tridiagonal matrix T.

7: **e**[dim] – double *Output*

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

On exit: the off-diagonal elements of the tridiagonal matrix T.

8: **tau**[dim] – Complex *Output*

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

On exit: further details of the unitary matrix Q.

9: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = $\langle$*value*$\rangle$.
Constraint: **n** ≥ 0.

On entry, **pda** $=$ $\langle$*value*$\rangle$.
Constraint: **pda** > 0.

NE_INT_2

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

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter $\langle$*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 tridiagonal matrix T is exactly similar to a nearby matrix $A+E$, where

$$\|E\|_2 \leq c(n)\epsilon\|A\|_2,$$

$c(n)$ is a modestly increasing function of n, and ϵ is the ***machine precision***.

The elements of T themselves may be sensitive to small perturbations in A or to rounding errors in the computation, but this does not affect the stability of the eigenvalues and eigenvectors.

8 Further Comments

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

To form the unitary matrix Q this function may be followed by a call to nag_zungtr (f08ftc):

```
nag_zungtr (order,uplo,n,&a,pda,tau,&fail)
```

To apply Q to an n by p complex matrix C this function may be followed by a call to nag_zunmtr (f08fuc). For example,

```
nag_zunmtr (order,Nag_LeftSide,uplo,Nag_NoTrans,n,p,&a,pda,
tau,&c,pdc,&fail)
```

forms the matrix product QC.

The real analogue of this function is nag_dsytrd (f08fec).

9 Example

To reduce the matrix A to tridiagonal form, where

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

9.1 Program Text

```
/* nag_zhetrd (f08fsc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

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

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

#ifdef NAG_COLUMN_MAJOR
#define A(I,J) a[(J-1)*pda + I - 1]
  order = Nag_ColMajor;
#else
```

```
#define A(I,J) a[(I-1)*pda + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);
  Vprintf("f08fsc Example Program Results\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n] ");
  Vscanf("%ld%*[^\n] ", &n);
  pda = n;
  d_len = n;
  e_len = n-1;
  tau_len = n-1;

  /* Allocate memory */
  if ( !(a = NAG_ALLOC(n * n, Complex)) ||
       !(d = NAG_ALLOC(d_len, double)) ||
       !(e = NAG_ALLOC(e_len, double)) ||
       !(tau = NAG_ALLOC(tau_len, Complex)) )
    {
      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 , %lf )", &A(i,j).re, &A(i,j).im);
        }
      Vscanf("%*[^\n] ");
    }
  else
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = 1; j <= i; ++j)
            Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
        }
      Vscanf("%*[^\n] ");
    }

  /* Reduce A to tridiagonal form */
  f08fsc(order, uplo, n, a, pda, d, e, tau, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f08fsc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Print tridiagonal form */
  Vprintf("\nDiagonal\n");
  for (i = 1; i <= n; ++i)
    Vprintf("%9.4f%s", d[i-1], i%8==0 ?"\n":" ");
  Vprintf("\nOff-diagonal\n");
```

```
  for (i = 1; i <= n - 1; ++i)
    Vprintf("%9.4f%s", e[i-1], i%8==0 ?"\n":" ");
  Vprintf("\n");
 END:
  if (a) NAG_FREE(a);
  if (d) NAG_FREE(d);
  if (e) NAG_FREE(e);
  if (tau) NAG_FREE(tau);

  return exit_status;
}
```

9.2 Program Data

```
f08fsc 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

```
f08fsc Example Program Results

Diagonal
  -2.2800   -0.1285   -0.1666   -1.9249
Off-diagonal
  -4.3385   -2.0226   -1.8023
```