NAG Library Function Document

nag_zhptrd (f08gsc)

1 Purpose

nag_zhptrd (f08gsc) reduces a complex Hermitian matrix to tridiagonal form, using packed storage.

2 Specification

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

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

3 Description

nag_zhptrd (f08gsc) reduces a complex Hermitian matrix A, held in packed storage, to real symmetric tridiagonal form T by a unitary similarity transformation: $A = QTQ^{\mathrm{H}}$.

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 9).

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

uplo = Nag_Upper
The upper triangular part of A is stored.

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: **ap**[dim] – Complex *Input/Output*

Note: the dimension, *dim*, of the array **ap** must be at least $\max(1, \mathbf{n} \times (\mathbf{n}+1)/2)$.

On entry: the upper or lower triangle of the n by n Hermitian matrix A, packed by rows or columns.

The storage of elements A_{ij} depends on the **order** and **uplo** arguments as follows:

if **order** = 'Nag_ColMajor' and **uplo** = 'Nag_Upper',
A_{ij} is stored in **ap**$[(j-1) \times j/2 + i - 1]$, for $i \leq j$;
if **order** = 'Nag_ColMajor' and **uplo** = 'Nag_Lower',
A_{ij} is stored in **ap**$[(2n-j) \times (j-1)/2 + i - 1]$, for $i \geq j$;
if **order** = 'Nag_RowMajor' and **uplo** = 'Nag_Upper',
A_{ij} is stored in **ap**$[(2n-i) \times (i-1)/2 + j - 1]$, for $i \leq j$;
if **order** = 'Nag_RowMajor' and **uplo** = 'Nag_Lower',
A_{ij} is stored in **ap**$[(i-1) \times i/2 + j - 1]$, for $i \geq j$.

On exit: **ap** is overwritten by the tridiagonal matrix T and details of the unitary matrix Q.

5: **d**[**n**] – double *Output*

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

6: **e**[**n** − **1**] – double *Output*

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

7: **tau**[**n** − **1**] – Complex *Output*

On exit: further details of the unitary matrix Q.

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

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

6 Error Indicators and Warnings

NE_BAD_PARAM

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

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

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 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 Parallelism and Performance

nag_zhptrd (f08gsc) is not threaded by NAG in any implementation.

nag_zhptrd (f08gsc) 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 $\frac{16}{3}n^3$.

To form the unitary matrix Q nag_zhptrd (f08gsc) may be followed by a call to nag_zupgtr (f08gtc):

```
nag_zupgtr(order,uplo,n,ap,tau,&q,pdq,&fail)
```

To apply Q to an n by p complex matrix C nag_zhptrd (f08gsc) may be followed by a call to nag_zupmtr (f08guc). For example,

```
nag_zupmtr(order,Nag_LeftSide,uplo,Nag_NoTrans,n,p,ap,tau,&c,
  pdc,&fail)
```

forms the matrix product QC.

The real analogue of this function is nag_dsptrd (f08gec).

10 Example

This example reduces the matrix A to tridiagonal form, where

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

using packed storage.

10.1 Program Text

```
/* nag_zhptrd (f08gsc) 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 <naga02.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          nag_enum_arg[40];
  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]
#define Z(I, J) z[(J - 1) * pdz + 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]
#define Z(I, J) z[(I - 1) * pdz + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  printf("nag_zhptrd (f08gsc) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%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)))
    {
      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_UPPER(i, j).re,
                     &A_UPPER(i, j).im);
            }
        }
      scanf("%*[^\n] ");
    }
  else
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = 1; j <= i; ++j)
            {
              scanf(" ( %lf , %lf )", &A_LOWER(i, j).re,
                     &A_LOWER(i, j).im);
            }
        }
      scanf("%*[^\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)
  {
    printf("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)
  {
    printf("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)
  {
    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("Eigenvalues\n");
for (i = 1; i <= n; ++i)
  printf("%8.4f%s", d[i-1], i%8 == 0?"\n":"            ");
printf("\n\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_NoLabels, 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(ap);
NAG_FREE(d);
```

```
  NAG_FREE(e);
  NAG_FREE(tau);
  NAG_FREE(z);

  return exit_status;
}
```

10.2 Program Data

```
nag_zhptrd (f08gsc) 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_zhptrd (f08gsc) Example Program Results

Eigenvalues
 -6.0002           -3.0030             0.5036            3.9996

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