NAG C Library Function Document
nag_zhptrd (f08gsc)

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

2 Specification

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: \( A = QTQ^H \).

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

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: 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
   
   On entry: \( n \), the order of the matrix \( A \).
   
   Constraint: \( n \geq 0 \).

4: ap[dim] – Complex
   
   Note: the dimension, \( dim \), of the array ap must be at least \( \max(1,n \times (n + 1)/2) \).
   
   On entry: the Hermitian matrix \( A \), packed by rows or columns. The storage of elements \( a_{ij} \) depends on the order and uplo parameters 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: A is overwritten by the tridiagonal matrix \( T \) and details of the unitary matrix \( Q \).

5: \( \text{d}[\text{dim}] \) – double  
   Output
   Note: the dimension, \( \text{dim} \), of the array \( \text{d} \) must be at least \( \max(1, n) \).
   On exit: the diagonal elements of the tridiagonal matrix \( T \).

6: \( \text{e}[\text{dim}] \) – double  
   Output
   Note: the dimension, \( \text{dim} \), of the array \( \text{e} \) must be at least \( \max(1, n - 1) \).
   On exit: the off-diagonal elements of the tridiagonal matrix \( T \).

7: \( \text{tau}[\text{dim}] \) – Complex  
   Output
   Note: the dimension, \( \text{dim} \), of the array \( \text{tau} \) must be at least \( \max(1, n - 1) \).
   On exit: further details of the unitary matrix \( Q \).

8: \( \text{fail} \) – NagError *  
   Output
   The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT  
On entry, \( n = \langle \text{value} \rangle \).
Constraint: \( n \geq 0 \).

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 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 \( \epsilon \) is the \textit{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_zupgtr (f08gtc):

\[
\text{nag\_zupgtr \ (order,uplo,n,ap,tau,&q,pdq,&fail)}
\]

To apply $Q$ to an $n \times p$ complex matrix $C$ this function may be followed by a call to nag_zupmtr (f08guc). For example,

\[
\text{nag\_zupmtr \ (order,Nag\_LeftSide,uplo,Nag\_NoTrans,n,p,ap,tau,&c, pdc,&fail)}
\]

forms the matrix product $QC$.

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

9 Example

To reduce the matrix $A$ to tridiagonal form, where

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

using packed storage.

9.1 Program Text

/* nag_zhptrd (f08gsc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 * *
 */

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

int main(void)
{
    /* Scalars */
    Integer i, j, n, ap_len, 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;
    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("f08gsc Example Program Results\n");
/* Skip heading in data file */
Vscanf("%*[\n] ");
Vscanf("%ld%*[\n] ", &n);
ap_len = n*(n+1)/2;
d_len = n;
e_len = n-1;
tau_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)) )
{
    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_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 */
f08gsc(order, uplo, n, ap, d, e, tau, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08gsc.\n");
    exit_status = 1;
    goto END;
}

/* Print tridiagonal form */
Vprintf("%nDiagonal\n");
for (i = 1; i <= n; ++i)
Vprintf("%9.4f\n", d[i-1], i%8==0 ?"\n" : "");
Vprintf("%nOff-diagonal\n");
for (i = 1; i <= n - 1; ++i)
9.2 Program Data

f08gsc Example Program Data

4

'U'

(-2.28, 0.00) (1.78, -2.03) (2.26, 0.10) (-0.12, 2.53)
(-1.12, 0.00) (0.01, 0.43) (-1.07, 0.86)
(-0.37, 0.00) (2.31, -0.92)
(-0.73, 0.00)

9.3 Program Results

f08gsc Example Program Results

Diagonal

-2.2800  -0.1285  -0.1666  -1.9249

Off-diagonal

-4.3385  -2.0226  -1.8023