NAG C Library Function Document

nag_dsptrd (f08gec)

1 Purpose

nag_dsptrd (f08gec) reduces a real symmetric matrix to tridiagonal form, using packed storage.

2 Specification

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

3 Description

nag_dsptrd (f08gec) reduces a real symmetric matrix \( A \), held in packed storage, to symmetric tridiagonal form \( T \) by an orthogonal similarity transformation: \( A = QTQ^T \).

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] – double

On entry: the symmetric matrix \( A \), packed by rows or columns. The storage of elements \( a_{ij} \) depends on the order and uplo parameters as follows:

Note: the dimension, dim, of the array ap must be at least max(1, n × (n + 1)/2).

Input/Output

Input
if order = Nag_ColMajor and uplo = Nag_Upper,  
\( a_{ij} \) is stored in \( \text{ap}[(j-1) \times j/2 + i - 1] \), for \( i \leq j \);

if order = Nag_ColMajor and uplo = Nag_Lower,  
\( a_{ij} \) is stored in \( \text{ap}[(2n - j) \times (j-1)/2 + i - 1] \), for \( i \geq j \);

if order = Nag_RowMajor and uplo = Nag_Upper,  
\( a_{ij} \) is stored in \( \text{ap}[(2 - n) \times i/2 + j - 1] \), for \( i \leq j \);

if order = Nag_RowMajor and uplo = Nag_Lower,  
\( a_{ij} \) is stored in \( \text{ap}[(i - 1) \times i/2 + j - 1] \), for \( i \geq j \).

On exit: \( A \) is overwritten by the tridiagonal matrix \( T \) and details of the orthogonal matrix \( Q \).

5: \( \mathbf{d}[\text{dim}] \) – double  
\text{Output}  

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

On exit: the diagonal elements of the tridiagonal matrix \( T \).

6: \( \mathbf{e}[\text{dim}] \) – double  
\text{Output}  

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

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

7: \( \mathbf{tau}[\text{dim}] \) – double  
\text{Output}  

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

On exit: further details of the orthogonal matrix \( Q \).

8: \( \text{fail} \) – NagError *  
\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: \( n \geq 0 \).

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 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 floating-point operations is approximately $\frac{4}{3}n^3$.

To form the orthogonal matrix $Q$ this function may be followed by a call to nag_dopgr (f08gfc):

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

To apply $Q$ to an $n$ by $p$ real matrix $C$ this function may be followed by a call to nag_dopmtr (f08ggc). For example,

\[ \text{nag_dopmtr (order, Nag_LeftSide, uplo, Nag_NoTrans, n, p, ap, tau, &c, pdc, &fail)} \]

forms the matrix product $QC$.

The complex analogue of this function is nag_zhptrd (f08gsc).

9 Example

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

\[
A = \begin{pmatrix}
2.07 & 3.87 & 4.20 & -1.15 \\
3.87 & -0.21 & 1.87 & 0.63 \\
4.20 & 1.87 & 1.15 & 2.06 \\
-1.15 & 0.63 & 2.06 & -1.81
\end{pmatrix},
\]

using packed storage.

9.1 Program Text

/* nag_dsptrd (f08gec) 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, 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];
    double *ap=0, *d=0, *e=0, *tau=0;

    ifdef NAG_COLUMN_MAJOR
    #define A_UPPER(I,J) ap[(J-I)*(J-1)/2 + I - 1]
    #define A_LOWER(I,J) ap[(2*n-I)*(I-1)/2 + J - 1]
    order = Nag_ColMajor;
    #else
    #define A_UPPER(I,J) ap[(I-J)*(J-1)/2 + I - 1]
    #define A_LOWER(I,J) ap[(2*n-J)*(J-1)/2 + I - 1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    Vprintf("f08gec 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, double)) ||
   !(d = NAG_ALLOC(d_len, double)) ||
   !(e = NAG_ALLOC(e_len, double)) ||
   !(tau = NAG_ALLOC(tau_len, double)) )
{
    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", &A_UPPER(i,j));
    }
    Vscanf("%*[\n] ");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf("%lf", &A_LOWER(i,j));
    }
    Vscanf("%*[\n] ");
}
/* Reduce A to tridiagonal form */
f08gec(order, uplo, n, ap, d, e, tau, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08gec.\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 (ap) NAG_FREE(ap);
if (d) NAG_FREE(d);
if (e) NAG_FREE(e);
if (tau) NAG_FREE(tau);
return exit_status;
}

9.2 Program Data

f08gec Example Program Data

4 :Value of N
'U' :Value of UPLO
2.07  3.87  4.20  -1.15
   -0.21  1.87   0.63
   1.15   2.06
   -1.81 :End of matrix A

9.3 Program Results

f08gec Example Program Results

Diagonal
2.0700  1.4741  -0.6492  -1.6949

Off-diagonal
-5.8258  2.6240   0.9163