NAG C Library Function Document

nag_dopmtr (f08ggc)

1 Purpose

nag_dopmtr (f08ggc) multiplies an arbitrary real matrix \( C \) by the real orthogonal matrix \( Q \) which was determined by nag_dsptrd (f08gec) when reducing a real symmetric matrix to tridiagonal form.

2 Specification

```c
void nag_dopmtr (Nag_OrderType order, Nag_SideType side, Nag_UploType uplo,
                Nag_TransType trans, Integer m, Integer n, double ap[],
                const double tau[], double c[], Integer pdc, NagError *fail)
```

3 Description

nag_dopmtr (f08ggc) is intended to be used after a call to nag_dsptrd (f08gec), which reduces a real symmetric matrix \( A \) to symmetric tridiagonal form \( T \) by an orthogonal similarity transformation: \( A = QTQ^T \). nag_dsptrd (f08gec) represents the orthogonal matrix \( Q \) as a product of elementary reflectors.

This function may be used to form one of the matrix products

\[
QC, \quad QT C, \quad CQ \text{ or } CQT,
\]

overwriting the result on \( C \) (which may be any real rectangular matrix).

A common application of this function is to transform a matrix \( Z \) of eigenvectors of \( T \) to the matrix \( QZ \) of eigenvectors of \( A \).

4 References


5 Parameters

1: \( \text{order} \) – Nag_OrderType

\( \text{Input} \)

On entry: the \text{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 \text{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: \text{order} = Nag_RowMajor or Nag_ColMajor.

2: \( \text{side} \) – Nag_SideType

\( \text{Input} \)

On entry: indicates how \( Q \) or \( Q^T \) is to be applied to \( C \) as follows:

\( \text{if } \text{side} = \text{Nag_LeftSide}, \text{Q or Q}^T \text{ is applied to C from the left}; \)

\( \text{if } \text{side} = \text{Nag_RightSide}, \text{Q or Q}^T \text{ is applied to C from the right}. \)

Constraint: \text{side} = Nag_LeftSide or Nag_RightSide.
3:  **uplo** – Nag_UploType  
   *Input*  
   On entry: this *must* be the same parameter *uplo* as supplied to nag_dsptrd (f08gec).  
   *Constraint:* *uplo* = Nag_Upper or Nag_Lower.

4:  **trans** – Nag_TransType  
   *Input*  
   On entry: indicates whether $Q$ or $Q^T$ is to be applied to $C$ as follows:  
   - if *trans* = Nag_NoTrans, $Q$ is applied to $C$;  
   - if *trans* = Nag_Trans, $Q^T$ is applied to $C$.  
   *Constraint:* *trans* = Nag_NoTrans or Nag_Trans.

5:  **m** – Integer  
   *Input*  
   On entry: $m$, the number of rows of the matrix $C$; $m$ is also the order of $Q$ if *side* = Nag_LeftSide.  
   *Constraint:* $m \geq 0$.

6:  **n** – Integer  
   *Input*  
   On entry: $n$, the number of columns of the matrix $C$; $n$ is also the order of $Q$ if *side* = Nag_RightSide.  
   *Constraint:* $n \geq 0$.

7:  **ap[dim]** – double  
   *Input/Output*  
   *Note:* the dimension, dim, of the array *ap* must be at least max(1,$m \times (m + 1)/2$) when *side* = Nag_LeftSide and at least max(1,$n \times (n + 1)/2$) when *side* = Nag_RightSide.  
   On entry: details of the vectors which define the elementary reflectors, as returned by nag_dsptrd (f08gec).  
   On exit: *ap* is used as internal workspace prior to being restored and hence is unchanged.

8:  **tau[dim]** – const double  
   *Input*  
   *Note:* the dimension, dim, of the array *tau* must be at least max(1,$m - 1$) when *side* = Nag_LeftSide and at least max(1,$n - 1$) when *side* = Nag_RightSide.  
   On entry: further details of the elementary reflectors, as returned by nag_dsptrd (f08gec).

9:  **c[dim]** – double  
   *Input/Output*  
   *Note:* the dimension, dim, of the array *c* must be at least max(1,$pdc \times n$) when *order* = Nag_ColMajor and at least max(1,$pdc \times m$) when *order* = Nag_RowMajor.  
   If *order* = Nag_ColMajor, the $(i,j)$th element of the matrix $C$ is stored in $c[(j-1) \times pdc + i - 1]$ and if *order* = Nag_RowMajor, the $(i,j)$th element of the matrix $C$ is stored in $c[(i-1) \times pdc + j - 1]$.  
   On entry: the $m$ by $n$ matrix $C$.  
   On exit: *c* is overwritten by $QC$ or $Q^T C$ or $CQ$ or $CQ^T$ as specified by *side* and *trans*.

10:  **pdc** – Integer  
   *Input*  
   On entry: the stride separating matrix row or column elements (depending on the value of *order*) in the array *c*.  
   *Constraints:*  
   - if *order* = Nag_ColMajor, $pdc \geq \max(1,m)$;  
   - if *order* = Nag_RowMajor, $pdc \geq \max(1,n)$.
6 Error Indicators and Warnings

**NE_INT**

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

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

On entry, \( pdc = \langle \text{value} \rangle \).
Constraint: \( pdc > 0 \).

**NE_INT_2**

On entry, \( pdc = \langle \text{value} \rangle, m = \langle \text{value} \rangle \).
Constraint: \( pdc \geq \max(1, m) \).

On entry, \( pdc = \langle \text{value} \rangle, n = \langle \text{value} \rangle \).
Constraint: \( pdc \geq \max(1, n) \).

**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 result differs from the exact result by a matrix \( E \) such that

\[ ||E||_2 = O(\epsilon)||C||_2, \]

where \( \epsilon \) is the *machine precision*.

8 Further Comments

The total number of floating-point operations is approximately \( 2m^2n \) if \( \text{side} = \text{Nag LeftSide} \) and \( 2mn^2 \) if \( \text{side} = \text{Nag RightSide} \).

The complex analogue of this function is nag_zupmtr (f08guc).

9 Example

To compute the two smallest eigenvalues, and the associated eigenvectors, of the matrix \( A \), 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. Here \( A \) is symmetric and must first be reduced to tridiagonal form \( T \) by nag_dsptrd (f08gec). The program then calls nag_dstebz (f08jjc) to compute the requested eigenvalues and nag_dstein...
(f08jkc) to compute the associated eigenvectors of $T$. Finally nag_dopmtr (f08ggc) is called to transform the eigenvectors to those of $A$.

9.1 Program Text

/* nag_dopmtr (f08ggc) 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>

int main(void)
{
    /* Scalars */
    Integer ap_len, i, j, m, n, nsplit, pdz, d_len, e_len;
    Integer tau_len;
    Integer exit_status=0;
    double vl=0.0, vu=0.0;
    NagError fail;
    Nag_UploType uplo;
    Nag_OrderType order;
    /* Arrays */
    char uplo_char[2];
    Integer *iblock=0, *ifailv=0, *isplit=0;
    double *ap=0, *d=0, *e=0, *tau=0, *w=0, *z=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
    \n    INIT_FAIL(fail);
    Vprintf("f08ggc Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%[^\n] ");
    Vscanf("%d%[^\n] ", &n);
    pdz = n;
    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, double)) ||
        !(d = NAG_ALLOC(d_len, double)) ||
        !(e = NAG_ALLOC(e_len, double)) ||
        !(iblock = NAG_ALLOC(n, Integer)) ||
        !(ifailv = NAG_ALLOC(n, Integer)) ||
        !(isplit = NAG_ALLOC(n, Integer)) ||
        !(w = NAG_ALLOC(n, double)) ||
        !(tau = NAG_ALLOC(tau_len, double)) ||
        !(z = NAG_ALLOC(n * n, double)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    f08ggc
    NAG C Library Manual
    f08ggc.4 [NP3645/7]
/* 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", &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 T = (Q**T)*A*Q */
f08gec(order, uplo, n, ap, d, e, tau, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f08gec.\n\n", fail.message);
  exit_status = 1;
  goto END;
}
/* Calculate the two smallest eigenvalues of T (same as A) */
f08jjc(Nag_Indices, Nag_ByBlock, n, vl, vu, 1, 2, 0.0,
d, e, &m, &nsplit, w, iblock, isplit, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f08jjc.\n\n", fail.message);
  exit_status = 1;
  goto END;
}
/* Print eigenvalues */
Vprintf("Eigenvalues\n");
for (i = 0; i < m; ++i)
  Vprintf("%8.4f%s", w[i], (i+1)%8==0 ?"\n": "");
Vprintf("\n\n");
/* Calculate the eigenvectors of T storing the result in Z */
f08jkc(order, n, d, e, m, w, iblock, isplit, z, pdz, ifailv, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f08jkc.\n\n", fail.message);
  exit_status = 1;
  goto END;
}
/* Calculate all the eigenvectors of A = Q*(eigenvectors of T) */
f08ggc(order, Nag_LeftSide, uplo, Nag_NoTrans, n, m, ap,
tau, z, pdz, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f08ggc.\n\n", fail.message);
  exit_status = 1;
goto END;
}
/* Print eigenvectors */
x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, m, z, pdz,
"Eigenvectors", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04cac.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
if (ap) NAG_FREE(ap);
if (d) NAG_FREE(d);
if (e) NAG_FREE(e);
if (iblock) NAG_FREE(iblock);
if (ifailv) NAG_FREE(ifailv);
if (isplit) NAG_FREE(isplit);
if (tau) NAG_FREE(tau);
if (w) NAG_FREE(w);
if (z) NAG_FREE(z);
return exit_status;
}

9.2 Program Data

f08ggc 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

f08ggc Example Program Results

Eigenvalues
-5.0034 -1.9987

Eigenvalues
-5.0034 -1.9987

Eigen vectors
1
1 0.5658 -0.2328
2 -0.3478 0.7994
3 -0.4740 -0.4087
4 0.5781 0.3737