NAG C Library Function Document

nag_dsygst (f08sec)

1 Purpose

nag_dsygst (f08sec) reduces a real symmetric-definite generalized eigenproblem \(Az = \lambda Bz\), \(ABz = \lambda z\) or \(BAz = \lambda z\) to the standard form \(Cy = \lambda y\), where \(A\) is a real symmetric matrix and \(B\) has been factorized by nag_dpotrf (f07fdc).

2 Specification

```c
void nag_dsygst (Nag_OrderType order, Nag_ComputeType comp_type,
                Nag_UploType uplo, Integer n, double a[], Integer pda,
                const double b[], Integer pdb, NagError *fail)
```

3 Description

To reduce the real symmetric-definite generalized eigenproblem \(Az = \lambda Bz\), \(ABz = \lambda z\) or \(BAz = \lambda z\) to the standard form \(Cy = \lambda y\), this function must be preceded by a call to nag_dpotrf (f07fdc) which computes the Cholesky factorization of \(B\); \(B\) must be positive-definite.

The different problem types are specified by the parameter \(\text{comp_type}\), as indicated in the table below. The table shows how \(C\) is computed by the function, and also how the eigenvectors \(z\) of the original problem can be recovered from the eigenvectors of the standard form.

<table>
<thead>
<tr>
<th>(\text{comp_type})</th>
<th>Problem</th>
<th>(B)</th>
<th>(C)</th>
<th>(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Az = \lambda Bz)</td>
<td>(U^T U) (L L^T)</td>
<td>(U^{-T} A U^{-1}) (L^{-1} A L^{-T})</td>
<td>(U^{-1} y) (L^{-T} y)</td>
</tr>
<tr>
<td>2</td>
<td>(ABz = \lambda z)</td>
<td>(U^T U) (L L^T)</td>
<td>(U A U^T) (L^T A L)</td>
<td>(U^{-1} y) (L^{-T} y)</td>
</tr>
<tr>
<td>3</td>
<td>(BAz = \lambda z)</td>
<td>(U^T U) (L L^T)</td>
<td>(U A U^T) (L^T A L)</td>
<td>(U^T y) (L y)</td>
</tr>
</tbody>
</table>

4 References


5 Parameters

1: \(\text{order}\) – Nag_OrderType
   
   \(\text{Input}\)
   
   \(\text{On entry:}\) 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 \(\text{order} = \text{Nag_RowMajor}\). See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

   \(\text{Constraint:}\) \(\text{order} = \text{Nag_RowMajor}\) or \(\text{Nag_ColMajor}\).

2: \(\text{comp_type}\) – Nag_ComputeType
   
   \(\text{Input}\)
   
   \(\text{On entry:}\) indicates how the standard form is computed as follows:
if comp_type = Nag_Compute_1,
    if uplo = Nag_Upper, $C = U^{-T}AU^{-1}$;
    if uplo = Nag_Lower, $C = L^{-1}AL^{-T}$;
if comp_type = Nag_Compute_2 or Nag_Compute_3,
    if uplo = Nag_Upper, $C = UAU^T$;
    if uplo = Nag_Lower, $C = L^TAL$.

Constraint: comp_type = Nag_Compute_1, Nag_Compute_2 or Nag_Compute_3.

3: uplo – Nag_UploType

On entry: indicates whether the upper or lower triangular part of $A$ is stored and how $B$ has been factorized, as follows:
    if uplo = Nag_Upper, the upper triangular part of $A$ is stored and $B = U^TU$;
    if uplo = Nag_Lower, the lower triangular part of $A$ is stored and $B = LL^T$.

Constraint: uplo = Nag_Upper or Nag_Lower.

4: n – Integer

On entry: $n$, the order of the matrices $A$ and $B$.

Constraint: $n \geq 0$.

5: $a[dim]$ – double

Input/Output

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

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

On entry: the $n$ by $n$ symmetric 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: the upper or lower triangle of $A$ is overwritten by the corresponding upper or lower triangle of $C$ as specified by comp_type and uplo.

6: pdav – Integer

Input

On entry: the stride separating matrix row or column elements (depending on the value of order) in the array $a$.

Constraint: pdav $\geq$ max$(1, n)$.

7: $b[dim]$ – const double

Input

Note: the dimension, $dim$, of the array $b$ must be at least max$(1, \text{pdb} \times n)$.

If order = Nag_ColMajor, the $(i,j)$th element of the matrix $B$ is stored in $b[(j-1) \times \text{pdb} + i - 1]$ and if order = Nag_RowMajor, the $(i,j)$th element of the matrix $B$ is stored in $b[(i-1) \times \text{pdb} + j - 1]$.

On entry: the Cholesky factor of $B$ as specified by uplo and returned by nag_dpotrf (f07fdc).

8: pdbv – Integer

Input

On entry: the stride separating matrix row or column elements (depending on the value of order) in the array $b$.

Constraint: pdbv $\geq$ max$(1, n)$.
6 Error Indicators and Warnings

**NE_INT**

On entry, \( n \) = \(<\text{value}>\).
Constraint: \( n \geq 0 \).
On entry, \( pda = \langle\text{value}>\).
Constraint: \( pda > 0 \).
On entry, \( pdb = \langle\text{value}>\).
Constraint: \( pdb > 0 \).

**NE_INT_2**

On entry, \( pda = \langle\text{value}>\), \( n = \langle\text{value}>\).
Constraint: \( pda \geq \max(1, n) \).
On entry, \( pdb = \langle\text{value}>\), \( n = \langle\text{value}>\).
Constraint: \( pdb \geq \max(1, n) \).

**NE_ALLOC_FAIL**

Memory allocation failed.

**NE_BAD_PARAM**

On entry, parameter \(<\text{value}>\) 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

Forming the reduced matrix \( C \) is a stable procedure. However it involves implicit multiplication by \( B^{-1} \) (if \( \text{comp_type} = \text{Nag_Compute}_1 \)) or \( B \) (if \( \text{comp_type} = \text{Nag_Compute}_2 \) or \( \text{Nag_Compute}_3 \)). When the function is used as a step in the computation of eigenvalues and eigenvectors of the original problem, there may be a significant loss of accuracy if \( B \) is ill-conditioned with respect to inversion.

8 Further Comments

The total number of floating-point operations is approximately \( n^3 \).

The complex analogue of this function is \( \text{nag_zhegst (f08ssc)} \).

9 Example

To compute all the eigenvalues of \( Az = \lambda Bz \), where

\[
A = \begin{pmatrix}
0.24 & 0.39 & 0.42 & -0.16 \\
0.39 & -0.11 & 0.79 & 0.63 \\
0.42 & 0.79 & -0.25 & 0.48 \\
-0.16 & 0.63 & 0.48 & -0.03
\end{pmatrix}
\quad \text{and} \quad
B = \begin{pmatrix}
4.16 & -3.12 & 0.56 & -0.10 \\
-3.12 & 5.03 & -0.83 & 1.18 \\
0.56 & -0.83 & 0.76 & 0.34 \\
-0.10 & 1.18 & 0.34 & 1.18
\end{pmatrix}.
\]

Here \( B \) is symmetric positive-definite and must first be factorized by \( \text{nag_dpotrf (f07fdc)} \). The program calls \( \text{nag_dsygst (f08sec)} \) to reduce the problem to the standard form \( Cy = \lambda y \); then \( \text{nag_dsytrd (f08fec)} \) to reduce \( C \) to tridiagonal form, and \( \text{nag_dstерf (f08jfc)} \) to compute the eigenvalues.
9.1 Program Text

/* nag_dsygst (f08sec) Example Program. */
* Copyright 2001 Numerical Algorithms Group.
* Mark 7, 2001. */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagf08.h>

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

    #ifdef NAG_COLUMN_MAJOR
    #define A(I,J) a[(J-1)*pda+I-1]
    #define B(I,J) b[(J-1)*pdb+I-1]
    order = Nag_ColMajor;
    #else
    #define A(I,J) a[(I-1)*pda+J-1]
    #define B(I,J) b[(I-1)*pdb+J-1]
    order = Nag_RowMajor;
    #endif
    INIT_FAIL(fail);
    Vprintf("f08sec Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[\n]");
    Vscanf("%ld%*[\n]", &n);

    #ifdef NAG_COLUMN_MAJOR
    pda = n;
    pdb = n;
    #else
    pda = n;
    pdb = n;
    #endif
    d_len = n;
    e_len = n-1;
    tau_len = n-1;

    /* Allocate memory */
    if ( !(a = NAG_ALLOC(n * n, double)) ||
        !(b = NAG_ALLOC(n * n, 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 and B from data file */
    Vscanf("%*[\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(i,j));
    }
    Vscanf("%*[\n]");
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
            Vscanf("%lf", &B(i,j));
    }
    Vscanf("%*[\n]");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf("%lf", &A(i,j));
    }
    Vscanf("%*[\n]");
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf("%lf", &B(i,j));
    }
    Vscanf("%*[\n]");
}

/* Compute the Cholesky factorization of B */
f07fdc(order, uplo, n, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07fdc.\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Reduce the problem to standard form C*y = lambda*y, storing */
/* the result in A */
f08sec(order, Nag_Compute_1, uplo, n, a, pda, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08sec.\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Reduce C to tridiagonal form T = (Q**T)*C*Q */
f08fec(order, uplo, n, a, pda, d, e, tau, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08fec.\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Calculate the eigenvalues of T (same as C) */
f08jfc(n, d, e, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08jfc.\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print eigenvalues */
Vprintf("Eigenvalues\n");
for (i = 1; i <= n; ++i)
    Vprintf("%8.4f%s", d[i-1], i%9==0 ?"\n":" ");
Vprintf("\n");
END:
    if (a) NAG_FREE(a);
    if (b) NAG_FREE(b);
    if (d) NAG_FREE(d);
    if (e) NAG_FREE(e);
    if (tau) NAG_FREE(tau);
    return exit_status;
}

9.2 Program Data
f08sec Example Program Data
4 :Value of N
   'L' :Value of UPLO
  0.24
  0.39 -0.11
  0.42  0.79 -0.25
-0.16  0.63  0.48 -0.03 :End of matrix A
  4.16
-3.12  5.03
  0.56 -0.83  0.76
-0.10  1.09  0.34  1.18 :End of matrix B

9.3 Program Results
f08sec Example Program Results

Eigenvalues
-2.2254 -0.4548  0.1001  1.1270