NAG Library Function Document

nag_zstegr (f08jyc)

1 Purpose

nag_zstegr (f08jyc) computes all the eigenvalues and, optionally, all the eigenvectors of a real $n$ by $n$ symmetric tridiagonal matrix.

2 Specification

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

void nag_zstegr (Nag_OrderType order, Nag_JobType job, Nag_RangeType range,
                 Integer n, double d[], double e[], double vl, double vu, Integer il,
                 Integer iu, Integer *m, double w[], Complex z[], Integer pdz,
                 Integer isuppz[], NagError *fail)
```

3 Description

nag_zstegr (f08jyc) computes all the eigenvalues and, optionally, the eigenvectors, of a real symmetric tridiagonal matrix $T$. That is, the function computes the spectral factorization of $T$ given by

$$
T = Z \Lambda Z^T,
$$

where $\Lambda$ is a diagonal matrix whose diagonal elements are the eigenvalues, $\lambda_i$, of $T$ and $Z$ is an orthogonal matrix whose columns are the eigenvectors, $z_i$, of $T$. Thus

$$
T z_i = \lambda_i z_i, \quad i = 1, 2, \ldots, n.
$$

The function stores the real orthogonal matrix $Z$ in a complex array, so that it may also be used to compute all the eigenvalues and eigenvectors of a complex Hermitian matrix $A$ which has been reduced to tridiagonal form $T$:

$$
A = Q T Q^H, \text{ where } Q \text{ is unitary}
$$

$$
= (Q Z) \Lambda (Q Z)^H.
$$

In this case, the matrix $Q$ must be explicitly applied to the output matrix $Z$. The functions which must be called to perform the reduction to tridiagonal form and apply $Q$ are:

- full matrix: nag_zhetrd (f08fsc) and nag_zunmtr (f08fuc)
- full matrix, packed storage: nag_zhptrd (f08gsc) and nag_zupmtr (f08guc)
- band matrix: nag_zhbtrd (f08hsc) with vect = Nag_FormQ and nag_zgemm (f16zac).

This function uses the dqds and the Relatively Robust Representation algorithms to compute the eigenvalues and eigenvectors respectively; see for example Parlett and Dhillon (2000) and Dhillon and Parlett (2004) for further details. nag_zstegr (f08jyc) can usually compute all the eigenvalues and eigenvectors in $O(n^2)$ floating-point operations and so, for large matrices, is often considerably faster than the other symmetric tridiagonal functions in this chapter when all the eigenvectors are required, particularly so compared to those functions that are based on the QR algorithm.
4 References


5 Arguments

1: order – Nag_OrderType

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: job – Nag_JobType

On entry: indicates whether eigenvectors are computed.

job = Nag_EigVals

Only eigenvalues are computed.

job = Nag_DoBoth

Eigenvalues and eigenvectors are computed.

Constraint: job = Nag_EigVals or Nag_DoBoth.

3: range – Nag_RangeType

On entry: indicates which eigenvalues should be returned.

range = Nag_AllValues

All eigenvalues will be found.

range = Nag_Interval

All eigenvalues in the half-open interval (vl, vu] will be found.

range = Nag_Indices

The ilth through iu th eigenvectors will be found.

Constraint: range = Nag_AllValues, Nag_Interval or Nag_Indices.

4: n – Integer

On entry: n, the order of the matrix T.

Constraint: n ≥ 0.

5: d[dim] – double

Note: the dimension, dim, of the array d must be at least max(1, n).

On entry: the n diagonal elements of the tridiagonal matrix T.

On exit: d is overwritten.
6: \( e[\text{dim}] \) – double

*Input/Output*

**Note:** the dimension, \( \dim \), of the array \( e \) must be at least \( \max(1, n) \).

**On entry:** \( e[0] \) to \( e[n-2] \) are the subdiagonal elements of the tridiagonal matrix \( T \). \( e[n-1] \) need not be set.

**On exit:** \( e \) is overwritten.

7: \( vl \) – double

*Input*

8: \( vu \) – double

*Input*

**On entry:** if \( \text{range} = \text{Nag\_Interval} \), \( vl \) and \( vu \) contain the lower and upper bounds respectively of the interval to be searched for eigenvalues.

If \( \text{range} = \text{Nag\_AllValues} \) or \( \text{Nag\_Indices} \), \( vl \) and \( vu \) are not referenced.

**Constraint:** if \( \text{range} = \text{Nag\_Interval} \), \( vl < vu \).

9: \( il \) – Integer

*Input*

10: \( iu \) – Integer

*Input*

**On entry:** if \( \text{range} = \text{Nag\_Indices} \), \( il \) and \( iu \) contains the indices (in ascending order) of the smallest and largest eigenvalues to be returned, respectively.

If \( \text{range} = \text{Nag\_AllValues} \) or \( \text{Nag\_Interval} \), \( il \) and \( iu \) are not referenced.

**Constraints:**

- if \( \text{range} = \text{Nag\_Indices} \) and \( n > 0 \), \( 1 \leq il \leq iu \leq n \);
- if \( \text{range} = \text{Nag\_Indices} \) and \( n = 0 \), \( il = 1 \) and \( iu = 0 \).

11: \( m \) – Integer *

*Output*

**On exit:** the total number of eigenvalues found. \( 0 \leq m \leq n \).

If \( \text{range} = \text{Nag\_AllValues} \), \( m = n \).

If \( \text{range} = \text{Nag\_Indices} \), \( m = iu - il + 1 \).

12: \( w[\text{dim}] \) – double

*Output*

**Note:** the dimension, \( \dim \), of the array \( w \) must be at least \( \max(1, n) \).

**On exit:** the eigenvalues in ascending order.

13: \( z[\text{dim}] \) – Complex

*Output*

**Note:** the dimension, \( \dim \), of the array \( z \) must be at least \( \max(1, pdz \times n) \) when \( \text{job} = \text{Nag\_DoBoth} \);

1 otherwise.

The \( (i, j) \)th element of the matrix \( Z \) is stored in

\[
Z[(j - 1) \times \text{pdz} + i - 1] \quad \text{when} \quad \text{order} = \text{Nag\_ColMajor};
\]

\[
Z[(i - 1) \times \text{pdz} + j - 1] \quad \text{when} \quad \text{order} = \text{Nag\_RowMajor}.
\]

**On exit:** if \( \text{job} = \text{Nag\_DoBoth} \), then if \( \text{fail\_code} = \text{NE\_NOERROR} \), the columns of \( z \) contain the orthonormal eigenvectors of the matrix \( T \), with the \( i \)th column of \( Z \) holding the eigenvector associated with \( w[i - 1] \).

If \( \text{job} = \text{Nag\_EigVals} \), \( z \) is not referenced.

14: \( pdz \) – Integer

*Input*

**On entry:** the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( z \).
Constraints:
if \( \text{job} = \text{Nag}_\text{DoBoth} \), \( \text{pdz} \geq \max(1, n) \);
otherwise \( \text{pdz} \geq 1 \).

15: \( \text{isuppz}[\text{dim}] \) – Integer
Output

Note: the dimension, \( \text{dim} \), of the array \( \text{isuppz} \) must be at least \( \max(1, 2 \times m) \).

On exit: the support of the eigenvectors in \( Z \), i.e., the indices indicating the nonzero elements in \( Z \). The \( i \)th eigenvector is nonzero only in elements \( \text{isuppz}[2 \times i - 2] \) through \( \text{isuppz}[2 \times i - 1] \).

16: \( \text{fail} = \text{NagError} \) * 
Input/Output

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

6 Error Indicators and Warnings

**NE_ALLOC_FAIL**
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**
On entry, argument \( \langle \text{value} \rangle \) had an illegal value.

**NE_CONVERGENCE**
Inverse iteration failed to converge.
The dqds algorithm failed to converge.

**NE_ENUM_INT_2**
On entry, \( \text{job} = \langle \text{value} \rangle \), \( \text{pdz} = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: if \( \text{job} = \text{Nag}_\text{DoBoth} \), \( \text{pdz} \geq \max(1, n) \);
otherwise \( \text{pdz} \geq 1 \).

**NE_ENUM_INT_3**
On entry, \( \text{range} = \langle \text{value} \rangle \), \( \text{il} = \langle \text{value} \rangle \), \( \text{iu} = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: if \( \text{range} = \text{Nag}_\text{Indices} \) and \( n > 0 \), \( 1 \leq \text{il} \leq \text{iu} \leq n \);
if \( \text{range} = \text{Nag}_\text{Indices} \) and \( n = 0 \), \( \text{il} = 1 \) and \( \text{iu} = 0 \).

**NE_ENUM_REAL_2**
On entry, \( \text{range} = \langle \text{value} \rangle \), \( \text{vl} = \langle \text{value} \rangle \) and \( \text{vu} = \langle \text{value} \rangle \).
Constraint: if \( \text{range} = \text{Nag}_\text{Interval} \), \( \text{vl} < \text{vu} \).

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

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

**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.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

f08jyc.4   Mark 25
7 Accuracy
See Section 4.7 of Anderson et al. (1999) and Barlow and Demmel (1990) for further details.

8 Parallelism and Performance
nag_zstegr (f08jyc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.
nag_zstegr (f08jyc) 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 X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments
The total number of floating-point operations required to compute all the eigenvalues and eigenvectors is approximately proportional to $n^2$.
The real analogue of this function is nag_dstegr (f08jlc).

10 Example
This example finds all the eigenvalues and eigenvectors of the symmetric tridiagonal matrix

$$T = \begin{pmatrix}
1.0 & 1.0 & 0 & 0 \\
1.0 & 4.0 & 2.0 & 0 \\
0 & 2.0 & 9.0 & 3.0 \\
0 & 0 & 3.0 & 16.0
\end{pmatrix}.$$

10.1 Program Text
/* nag_zstegr (f08jyc) Example Program. *
 * Copyright 2014 Numerical Algorithms Group. *
 * Mark 23, 2011. */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    double vl = 0.0, vu = 0.0;
    Integer i, j, il = 0, iu = 0, m, n, pdz;
    Integer exit_status = 0;
    /* Arrays */
    char nag_enum_arg[40];
    Complex *z = 0;
    double *d = 0, *e = 0, *w = 0;
Integer *isuppz = 0;
/* Nag Types */
Nag_OrderType order;
Nag_JobType job;
Nag_RangeType range;
NagError fail;
#endif
#define Z(I, J) z[(J - 1) * pdz + I - 1]
order = Nag_ColMajor;
#else
#define Z(I, J) z[(I - 1) * pdz + J - 1]
order = Nag_RowMajor;
#endif
INIT_FAIL(fail);
printf("nag_zstegr (f08jyc) Example Program Results\n\n");
/* Skip heading in data file */
#endif
scanf_s("%*[\n]");
#else
scanf("%*[\n]");
#endif
#ifdef _WIN32
scanf_s("%"NAG_IFMT"%*[\n]", \n);
#else
scanf("%"NAG_IFMT"%*[\n]", \n);
#endif
m = n;
#ifdef _WIN32
scanf_s("%39s%*[\n]", nag_enum_arg, _countof(nag_enum_arg));
#else
scanf("%39s%*[\n]", nag_enum_arg);
#endif
job = (Nag_JobType) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
scanf_s("%39s%*[\n]", nag_enum_arg, _countof(nag_enum_arg));
#else
scanf("%39s%*[\n]", nag_enum_arg);
#endif
range = (Nag_RangeType) nag_enum_name_to_value(nag_enum_arg);
#ifdef NAG_COLUMN_MAJOR
pdz = n;
#else
pdz = n;
#endif
/* Allocate memory */
if (!(z = NAG_ALLOC(n*m, Complex)) ||
(d = NAG_ALLOC(n, double)) ||
(e = NAG_ALLOC(n, double)) ||
(w = NAG_ALLOC(n, double)) ||
(!isuppz = NAG_ALLOC(2*m, Integer)))
{
printf("Allocation failure\n");
exit_status = -1;
goto END;
}
/* Read the symmetric tridiagonal matrix T from data file, first
* the diagonal elements, then the off diagonal elements. */

f08jyc.6 Mark 25
for (i = 0; i < n; ++i)
#ifdef _WIN32
    scanf_s("%lf", &d[i]);
#else
    scanf("%lf", &d[i]);
#endif
#endif
for (i = 0; i < n-1; ++i)
#ifdef _WIN32
    scanf_s("%lf", &e[i]);
#else
    scanf("%lf", &e[i]);
#endif
#endif
/* nag_zstegr (f08jyc).
 * Calculate all the eigenvalues of T.
 */
ag_zstegr(order, job, range, n, d, e, vl, vu, il, iu, &m, w, z, pdz, isuppz, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zstegr (f08jyc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* nag_complex_divide (a02cdc).
 * Normalize the eigenvectors.
 */
for(j=1; j<=m; j++)
    for(i=n; i>=1; i--)
        Z(i, j) = nag_complex_divide(Z(i, j),Z(1, j));
/* Print eigenvalues and eigenvectors */
printf("%s\n", "Eigenvalues");
for (i = 0; i < m; ++i)
    printf("%8.4f\n", (i+1)%8 == 0?"\n":" ");
printf("\n\n");
/* nag_gen_complx_mat_print (x04dac).
 * Print eigenvectors.
 */
fflush(stdout);
nag_gen_complx_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, m, z, pdz, "Eigenvectors", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_complx_mat_print (x04dac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(z);
NAG_FREE(d);
NAG_FREE(e);
NAG_FREE(w);
NAG_FREE(isuppz);
return exit_status;
};

#define Z

10.2 Program Data

nag_zstegr (f08jyc) Example Program Data

4
Nag_DoBoth
Nag_AllValues
1.0 4.0 9.0 16.0
1.0 2.0 3.0

10.3 Program Results

nag_zstegr (f08jyc) Example Program Results

Eigenvalues
0.6476 3.5470 8.6578 17.1477

Eigenvectors

1 2 3 4
1 1.0000 1.0000 1.0000 1.0000
0.0000 0.0000 0.0000 0.0000
2 -0.3524 2.5470 7.6578 16.1477
0.0000 0.0000 0.0000 0.0000
3 0.0908 -1.0769 17.3340 105.6521
0.0000 0.0000 0.0000 0.0000
4 -0.0177 0.2594 -7.0826 276.1742
0.0000 0.0000 0.0000 0.0000