NAG Library Function Document

nag_zheevr (f08frc)

1 Purpose

nag_zheevr (f08frc) computes selected eigenvalues and, optionally, eigenvectors of a complex \( n \) by \( n \) Hermitian matrix \( A \). Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues.

2 Specification

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

void nag_zheevr (Nag_OrderType order, Nag_JobType job, Nag_RangeType range,
                 Nag_UploType uplo, Integer n, Complex a[], Integer pda,
                 double vl, double vu, Integer il, Integer iu, double abstol,
                 Integer *m, double w[], Complex z[], Integer pdz,
                 Integer isuppz[], NagError *fail)
```

3 Description

The Hermitian matrix is first reduced to a real tridiagonal matrix \( T \), using unitary similarity transformations. Then whenever possible, nag_zheevr (f08frc) computes the eigenspectrum using Relatively Robust Representations. nag_zheevr (f08frc) computes eigenvalues by the dqds algorithm, while orthogonal eigenvectors are computed from various ‘good’ LDL\(^T\) representations (also known as Relatively Robust Representations). Gram–Schmidt orthogonalization is avoided as far as possible. More specifically, the various steps of the algorithm are as follows. For the \( i \)th unreduced block of \( T \):

(a) compute \( T - \sigma_i I = L_i D_i L_i^T \), such that \( L_i D_i L_i^T \) is a relatively robust representation,

(b) compute the eigenvalues, \( \lambda_i \), of \( L_i D_i L_i^T \) to high relative accuracy by the dqds algorithm,

(c) if there is a cluster of close eigenvalues, 'choose' \( \sigma_i \) close to the cluster, and go to (a),

(d) given the approximate eigenvalue \( \lambda_i \) of \( L_i D_i L_i^T \), compute the corresponding eigenvector by forming a rank-revealing twisted factorization.

The desired accuracy of the output can be specified by the argument \texttt{abstol}. For more details, see Dhillon (1997) and Parlett and Dhillon (2000).

4 References


Mark 25

f08frc.1
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: if range = Nag_AllValues, all eigenvalues will be found.  
   If range = Nag_Interval, all eigenvalues in the half-open interval (vl, vu] will be found.  
   If range = Nag_Indices, the ilth to iuth eigenvalues will be found.  
   For range = Nag_Interval or Nag_Indices and il – il < n – 1, nag_dstebz (f08jjc) and nag_zstein (f08jxc) are called.  
   Constraint: range = Nag_AllValues, Nag_Interval or Nag_Indices.

4: uplo – Nag_UploType  
   On entry: 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.

5: n – Integer  
   On entry: n, the order of the matrix A.  
   Constraint: n ≥ 0.

6: a[dim] – Complex  
   Note: the dimension, dim, of the array a must be at least max(1, pda × n).  
   On entry: the n by n Hermitian matrix A.  
   If order = Nag_ColMajor, Aij is stored in a[(j – 1) × pda + i – 1].  
   If order = Nag_RowMajor, Aij is stored in a[(i – 1) × pda + j – 1].  
   If uplo = Nag_Upper, the upper triangular part of A must be stored and the elements of the array below the diagonal are not referenced.  
   If uplo = Nag_Lower, the lower triangular part of A must be stored and the elements of the array above the diagonal are not referenced.  
   On exit: the lower triangle (if uplo = Nag_Lower) or the upper triangle (if uplo = Nag_Upper) of a, including the diagonal, is overwritten.
7: \texttt{pda} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the stride separating row or column elements (depending on the value of \texttt{order}) in the array \texttt{a}.

\textit{Constraint:} \texttt{pda} \geq \max(1, n).

8: \texttt{vl} – double \hspace{1cm} \textit{Input}

9: \texttt{vu} – double \hspace{1cm} \textit{Input}

\textit{On entry:} if \texttt{range} = Nag\_Interval, the lower and upper bounds of the interval to be searched for eigenvalues.

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

\textit{Constraint:} if \texttt{range} = Nag\_Interval, \texttt{vl} < \texttt{vu}.

10: \texttt{il} – Integer \hspace{1cm} \textit{Input}

11: \texttt{iu} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} if \texttt{range} = Nag\_Indices, the indices (in ascending order) of the smallest and largest eigenvalues to be returned.

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

\textit{Constraints:}
- if \texttt{range} = Nag\_Indices and \texttt{n} = 0, \texttt{il} = 1 and \texttt{iu} = 0;
- if \texttt{range} = Nag\_Indices and \texttt{n} > 0, 1 \leq \texttt{il} \leq \texttt{iu} \leq \texttt{n}.

12: \texttt{abstol} – double \hspace{1cm} \textit{Input}

\textit{On entry:} the absolute error tolerance for the eigenvalues. An approximate eigenvalue is accepted as converged when it is determined to lie in an interval \([a, b]\) of width less than or equal to

\[\text{abstol} + \epsilon \max(|a|, |b|),\]

where \(\epsilon\) is the \textit{machine precision}. If \texttt{abstol} is less than or equal to zero, then \(\epsilon\|T\|_1\) will be used in its place, where \(T\) is the real tridiagonal matrix obtained by reducing \(A\) to tridiagonal form. See Demmel and Kahan (1990).

If high relative accuracy is important, set \texttt{abstol} to \texttt{nag\_real\_safe\_small\_number()} , although doing so does not currently guarantee that eigenvalues are computed to high relative accuracy. See Barlow and Demmel (1990) for a discussion of which matrices can define their eigenvalues to high relative accuracy.

13: \texttt{m} – Integer * \hspace{1cm} \textit{Output}

\textit{On exit:} the total number of eigenvalues found. 0 \leq \texttt{m} \leq \texttt{n}.

If \texttt{range} = Nag\_AllValues, \texttt{m} = \texttt{n}.

If \texttt{range} = Nag\_Indices, \texttt{m} = \texttt{iu} – \texttt{il} + 1.

14: \texttt{w}[\texttt{dim}] – double \hspace{1cm} \textit{Output}

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

\textit{On exit:} the first \texttt{m} elements contain the selected eigenvalues in ascending order.

15: \texttt{z}[\texttt{dim}] – Complex \hspace{1cm} \textit{Output}

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

1 otherwise.
The \((i, j)\)th element of the matrix \(Z\) is stored in
\[
Z[(j-1) \times \text{pdz} + i - 1] \quad \text{when order = Nag_ColMajor;}
\]
\[
Z[(i-1) \times \text{pdz} + j - 1] \quad \text{when order = Nag_RowMajor.}
\]

On exit: if job = Nag_DoBoth, the first \(m\) columns of \(Z\) contain the orthonormal eigenvectors of
the matrix \(A\) corresponding to the selected eigenvalues, with the \(i\)th column of \(Z\) holding the
eigenvector associated with \(w[i-1]\).

If job = Nag_EigVals, \(Z\) is not referenced.

16: \(\text{pdz} - \text{Integer} \quad \text{Input}\)

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

Constraints:
\[
\text{if job = Nag_DoBoth, pdz} \geq \max(1, n); \quad \text{otherwise pdz} \geq 1.
\]

17: \(\text{isuppz[\text{dim}]} - \text{Integer} \quad \text{Output}\)

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 \(\text{isuppz}[2 \times i - 2] \text{ through } \text{isuppz}[2 \times i - 1]\).
Implemented only for range = Nag_AllValues or Nag_Indices and \(iu - il = n - 1\).

18: \(\text{fail} - \text{NagError} * \quad \text{Input/Output}\)

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
nag_zheevr (f08frc) 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 \(\text{job} = \text{Nag_DoBoth}, \text{pdz} \geq \max(1, n); \quad \text{otherwise pdz} \geq 1\).

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

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 \(\text{range} = \text{Nag_Interval}, \text{vl} < \text{vu}\).
NE_INT
On entry, $n = \langle value \rangle$.
Constraint: $n \geq 0$.
On entry, $pda = \langle value \rangle$.
Constraint: $pda > 0$.
On entry, $pdz = \langle value \rangle$.
Constraint: $pdz > 0$.

NE_INT_2
On entry, $pda = \langle value \rangle$ and $n = \langle value \rangle$.
Constraint: $pda \geq \max(1, n)$.

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.

NE_NO_LICENCE
Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

7 Accuracy
The computed eigenvalues and eigenvectors are exact for a nearby matrix $(A + E)$, where

$$\|E\|_2 = O(\epsilon)\|A\|_2,$$

and $\epsilon$ is the machine precision. See Section 4.7 of Anderson et al. (1999) for further details.

8 Parallelism and Performance
nag_zheevr (f08frc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_zheevr (f08frc) 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 is proportional to $n^3$.
The real analogue of this function is nag_dsyevr (f08fdc).
10 Example

This example finds the eigenvalues with indices in the range $[2, 3]$, and the corresponding eigenvectors, of the Hermitian matrix

$$A = \begin{pmatrix} 1 & 2 - i & 3 - i & 4 - i \\ 2 + i & 2 & 3 - 2i & 4 - 2i \\ 3 + i & 3 + 2i & 3 & 4 - 3i \\ 4 + i & 4 + 2i & 4 + 3i & 4 \end{pmatrix}.$$

10.1 Program Text

/* nag_zheevr (f08frc) Example Program.  *
* Copyright 2014 Numerical Algorithms Group.  *
* Mark 23, 2011.  */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    double abstol, vl = 0.0, vu = 0.0;
    Integer i, il, iu, j, m, n, pda, pdz;
    Integer exit_status = 0;
    /* Arrays */
    Complex *a = 0, *z = 0;
    double *w = 0;
    Integer *isuppz = 0;
    /* Nag Types */
    Nag_OrderType order;
    NagError fail;
    #ifdef NAG_COLUMN_MAJOR
    #define A(I, J) a[(J - 1) * pda + I - 1]
    #define Z(I, J) z[(J - 1) * pdz + I - 1]
    order = Nag_ColMajor;
    #else
    #define A(I, J) a[(I - 1) * pda + J - 1]
    #define Z(I, J) z[(I - 1) * pdz + J - 1]
    order = Nag_RowMajor;
    #endif
    INIT_FAIL(fail);
    printf("nag_zheevr (f08frc) Example Program Results\n\n");
    /* Skip heading in data file and read n and the lower and upper *
    * indices of the smallest and largest eigenvalues to be found.  */
    #ifdef _WIN32
    scanf_s("%*[\n");
    #else
    scanf("%*[\n");
    #endif
    #ifdef _WIN32
    scanf_s("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[\n");
    &n, &il, &iu);  
    #else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[\n");
    &n, &il, &iu);  
    #endif
    m = iu - il + 1;
```c
pda = n;
pdz = n;

/* Allocate memory */
if (!(a = NAG_ALLOC(n*n, Complex)) ||
   !(z = NAG_ALLOC(n*pdz, Complex)) ||
   !(w = NAG_ALLOC(n, double)) ||
   !(isuppz = NAG_ALLOC(2*pdz, Integer)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read the upper triangular part of the matrix A from data file */
for (i = 1; i <= n; ++i)
    for (j = i; j <= n; ++j)
        #ifdef _WIN32
            scanf_s(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
        #else
            scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
        #endif
    #ifdef _WIN32
        scanf_s("%*[^n]"n);
    #else
        scanf("%*[^n]"n);
    #endif

/* Set the absolute error tolerance for eigenvalues.
* With abstol set to zero, the default value is used instead.
*/
    abstol = 0.0;

/* nag_zheevr (f08frc).
* Solve the symmetric eigenvalue problem.
*/
    nag_zheevr(order, Nag_DoBoth, Nag_Indices, Nag_Upper, n, a, pda, vl, vu, il,
               iu, abstol, &m, w, z, pdz, isuppz, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_zheevr (f08frc).
", 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 solution */
    printf("Selected eigenvalues\n");
    for (j = 0; j < m; ++j)
        printf("%8.4f\n", w[j], (j+1)%8 == 0?"\n":" ");
    printf("\n");

/* nag_gen_complx_mat_print (x04dac).
* Print selected eigenvectors.
*/
    fflush(stdout);
    nag_gen_complx_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n,
                             m, z, pdz, "Selected eigenvectors", 0, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_gen_complx_mat_print (x04dac).
", fail.message);
        exit_status = 1;
        goto END;
    }
```

Mark 25

f08 – Least-squares and Eigenvalue Problems (LAPACK)
END:
NAG_FREE(a);
NAG_FREE(z);
NAG_FREE(w);
NAG_FREE(isuppz);
    return exit_status;
}
#endif A
#endif 2

10.2 Program Data

nag_zheevr (f08frc) Example Program Data

    4   2   3                      :Values of n, il and iu

    (1.0, 0.0)  (2.0,-1.0)  (3.0,-1.0)  (4.0,-1.0)
    (2.0, 0.0)  (3.0,-2.0)  (4.0,-2.0)    
    (3.0, 0.0)  (4.0,-3.0)    
    (4.0, 0.0)                      :End of matrix A

10.3 Program Results

nag_zheevr (f08frc) Example Program Results

Selected eigenvalues
-0.6886    1.1412
Selected eigenvectors

1   2
1   1.0000   1.0000
   0.0000  -0.0000

2  -0.7703   0.0516
   -0.1746  1.2795

3   0.4559  -1.1962
   0.4892  -0.2954

4  -0.3464   0.7876
   -0.4448  -0.5075