NAG Library Function Document

nag_zhpevd (f08gqc)

1 Purpose

nag_zhpevd (f08gqc) computes all the eigenvalues and, optionally, all the eigenvectors of a complex Hermitian matrix held in packed storage. If the eigenvectors are requested, then it uses a divide-and-conquer algorithm to compute eigenvalues and eigenvectors. However, if only eigenvalues are required, then it uses the Pal–Walker–Kahan variant of the QL or QR algorithm.

2 Specification

```c
#include <nag.h>
#include <nagf08.h>
void nag_zhpevd (Nag_OrderType order, Nag_JobType job, Nag_UploType uplo,
                  Integer n, Complex ap[], double w[], Complex z[], Integer pdz,
                  NagError *fail)
```

3 Description

nag_zhpevd (f08gqc) computes all the eigenvalues and, optionally, all the eigenvectors of a complex Hermitian matrix $A$ (held in packed storage). In other words, it can compute the spectral factorization of $A$ as

$$A = Z \Lambda Z^H,$$

where $\Lambda$ is a real diagonal matrix whose diagonal elements are the eigenvalues $\lambda_i$, and $Z$ is the (complex) unitary matrix whose columns are the eigenvectors $z_i$. Thus

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

4 References


5 Arguments

1:  **order** – Nag_OrderType  

   **Input**

   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  

   *Input*

   *On entry*: indicates whether eigenvectors are computed.

   **job** = Nag_DoNothing  
   Only eigenvalues are computed.

   **job** = Nag_EigVecs  
   Eigenvalues and eigenvectors are computed.

   **Constraint**: **job** = Nag_DoNothing or Nag_EigVecs.

3:  **uplo** – Nag_UploType  

   *Input*

   *On entry*: indicates whether the upper or lower triangular part of **A** is stored.

   **uplo** = Nag_Upper  
   The upper triangular part of **A** is stored.

   **uplo** = Nag_Lower  
   The lower triangular part of **A** is stored.

   **Constraint**: **uplo** = Nag_Upper or Nag_Lower.

4:  **n** – Integer  

   *Input*

   *On entry*: **n**, the order of the matrix **A**.

   **Constraint**: **n** ≥ 0.

5:  **ap[dim]** – Complex  

   *Input/Output*

   **Note**: the dimension, **dim**, of the array **ap** must be at least \( \max(1, \text{n} \times (\text{n} + 1)/2) \).

   *On entry*: the upper or lower triangle of the \( n \) by \( n \) Hermitian matrix **A**, packed by rows or columns.

   The storage of elements \( A_{ij} \) depends on the **order** and **uplo** arguments as follows:

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

   *On exit*: **ap** is overwritten by the values generated during the reduction to tridiagonal form. The elements of the diagonal and the off-diagonal of the tridiagonal matrix overwrite the corresponding elements of **A**.

6:  **w[dim]** – double  

   *Output*

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

   *On exit*: the eigenvalues of the matrix **A** in ascending order.

7:  **z[dim]** – Complex  

   *Output*

   **Note**: the dimension, **dim**, of the array **z** must be at least

   \( \max(1, \text{pdz} \times \text{n}) \) when **job** = Nag_EigVecs;

   1 when **job** = Nag_DoNothing.

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

   \( z[(j - 1) \times \text{pdz} + i - 1] \) when **order** = Nag_ColMajor;

   \( z[(i - 1) \times \text{pdz} + j - 1] \) when **order** = Nag_RowMajor.
On exit: If job = Nag_EigVecs, z is overwritten by the unitary matrix Z which contains the eigenvectors of A.

If job = Nag_DoNothing, z is not referenced.

8: pdz – Integer

Input

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

Constraints:

- if job = Nag_EigVecs, pdz ≥ max(1, n);
- if job = Nag_DoNothing, pdz ≥ 1.

9: fail – 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 ⟨value⟩ had an illegal value.

NE_CONVERGENCE

If fail.errnum = ⟨value⟩ and job = Nag_DoNothing, the algorithm failed to converge: ⟨value⟩ elements of an intermediate tridiagonal form did not converge to zero; if fail.errnum = ⟨value⟩ and job = Nag_EigVecs, then the algorithm failed to compute an eigenvalue while working on the submatrix lying in rows and column ⟨value⟩/(n + 1) through ⟨value⟩ mod (n + 1).

NE_ENUM_INT_2

On entry, job = ⟨value⟩, pdz = ⟨value⟩ and n = ⟨value⟩.

Constraint: if job = Nag_EigVecs, pdz ≥ max(1, n);
- if job = Nag_DoNothing, pdz ≥ 1.

NE_INT

On entry, n = ⟨value⟩.

Constraint: n ≥ 0.

On entry, pdz = ⟨value⟩.

Constraint: 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.

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 \textit{machine precision}. See Section 4.7 of Anderson \textit{et al.} (1999) for further details.

8 Parallelism and Performance

\texttt{nag\_zhpevd (f08gqc)} is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

\texttt{nag\_zhpevd (f08gqc)} 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 real analogue of this function is \texttt{nag\_dspevd (f08gcc)}.

10 Example

This example computes all the eigenvalues and eigenvectors of the Hermitian matrix \( A \), where
\[
A = \begin{pmatrix}
1.0 + 0.0i & 2.0 - 1.0i & 3.0 - 1.0i & 4.0 - 1.0i \\
2.0 + 1.0i & 2.0 + 0.0i & 3.0 - 2.0i & 4.0 - 2.0i \\
3.0 + 1.0i & 3.0 + 2.0i & 3.0 + 0.0i & 4.0 - 3.0i \\
4.0 + 1.0i & 4.0 + 2.0i & 4.0 + 3.0i & 4.0 + 0.0i
\end{pmatrix}.
\]

10.1 Program Text

/* \texttt{nag\_zhpevd (f08gqc)} Example Program. *
* * Copyright 2014 Numerical Algorithms Group. *
* * Mark 7, 2001. *
*/

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <naga02.h>

int main(void)
{

/* Scalars */
Integer i, j, n, ap_len, pdz, w_len;
Integer exit_status = 0;
NagError fail;
Nag_JobType job;
Nag_UploType uplo;
Nag_OrderType order;
/* Arrays */
char nag_enum_arg[40];
Complex *ap = 0, *z = 0;
double *w = 0;

#ifdef NAG_COLUMN_MAJOR
#define A_UPPER(I, J) ap[J * (J - 1) / 2 + I - 1]
#else
#define A_UPPER(I, J) ap[J * (J - 1) / 2 + I]```
#define A_LOWER(I, J) ap[(2 *n - J) *(J-1) / 2 + I - 1]
#define Z(I, J) z[(J - 1) * pdz + 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]
#define Z(I, J) z[(I - 1) * pdz + J - 1]

order = Nag_RowMajor;
#endif

INIT_FAIL(fail);

printf("nag_zhpevd (f08gqc) Example Program Results\n\n");

/* Skip heading in data file */
#ifdef _WIN32
scanf_s("%*[\n"]);
#else
scanf("%*[\n"]);
#endif
#endif
#ifdef _WIN32
scanf_s("%"NAG_IFMT"%*[\n"] , &n);
#else
scanf("%"NAG_IFMT"%*[\n"] , &n);
#endif

#endif

ap_len = n*(n+1)/2;
w_len = n;
pdz = n;

/* Allocate memory */
if (!(ap = NAG_ALLOC(ap_len, Complex)) ||
 !(z = NAG_ALLOC(n * n, Complex)) ||
 !(w = NAG_ALLOC(w_len, double)))
{
 printf("Allocation failure\n");
 exit_status = -1;
 goto END;
}

/* Read whether Upper or Lower part of A is stored */
#ifdef _WIN32
scanf_s("%39s%*[\n"] , nag_enum_arg, _countof(nag_enum_arg));
#else
scanf("%39s%*[\n"] , nag_enum_arg);
#endif

/* nag_enum_name_to_value (x04nac).
* Converts NAG enum member name to value */
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);

/* Read A from data file */
if (uplo == Nag_Upper)
{
 for (i = 1; i <= n; ++i)
  {
   for (j = i; j <= n; ++j)
   {
 #ifdef _WIN32
 scanf_s("( %lf , %lf )", &A_UPPER(i, j).re,
 &A_UPPER(i, j).im);
 #else
 scanf("( %lf , %lf )", &A_UPPER(i, j).re,
 &A_UPPER(i, j).im);
 #endif
  }
 #ifdef _WIN32
 scanf_s("%*[\n"]);
 #else
 scanf("%*[\n"]);
 #endif
  }
#else

}}
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= i; ++j)
    {
        #ifdef _WIN32
            scanf_s("( %lf , %lf )", &A_LOWER(i, j).re,
                   &A_LOWER(i, j).im);
        #else
            scanf("( %lf , %lf )", &A_LOWER(i, j).re,
                   &A_LOWER(i, j).im);
        #endif
    }
}
#endif
}  
  
#ifdef _WIN32
  scanf_s("%*[

"]");
#else
  scanf("%*[

"]");
#endif
}  
  /* Read type of job to be performed */
#endif
}  
  
#ifdef _WIN32
  scanf_s("%39s%*[

"]", nag_enum_arg, _countof(nag_enum_arg));
#else
  scanf("%39s%*[

"]", nag_enum_arg);
#endif
  job = (Nag_JobType) nag_enum_name_to_value(nag_enum_arg);
  /* Calculate all the eigenvalues and eigenvectors of A */
  /* nag_zhpevd (f08gqc). */
  /* All eigenvalues and optionally all eigenvectors of */
  /* complex Hermitian matrix, packed storage */
  /* (divide-and-conquer) */
  /* */
  nag_zhpevd(order, job, uplo, n, ap, w, z, pdz, &fail);
  if (fail.code != NE_NOERROR)
  {
      printf("Error from nag_zhpevd (f08gqc).
%s
", fail.message);
      exit_status = 1;
      goto END;
  }
  /* Normalize the eigenvectors */
  for(j=1; j<=n; j++)
  {
      for(i=n; i>=1; i--)
      {
          Z(i, j) = nag_complex_divide(Z(i, j),Z(1, j));
      }
  }
  /* Print eigenvalues and eigenvectors */
  printf("Eigenvalues\n");
  for (i = 0; i < n; ++i)
    printf(" %5"NAG_IFMT" %8.4f\n", i + 1, w[i]);
  printf("\n");
  /* nag_gen_complx_mat_print_comp (x04dbc). */
  /* Print complex general matrix (comprehensive) */
  /* */
  fflush(stdout);
  nag_gen_complx_mat_print_comp(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n,
                   n, z, pdz, Nag_AboveForm, "%7.4f",
                   "Eigenvectors", Nag_IntegerLabels, 0,
                   Nag_IntegerLabels, 0, 80, 0, 0,
                   &fail);
  if (fail.code != NE_NOERROR)
  {
      printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
  }
END:
NAG_FREE(ap);
NAG_FREE(w);
NAG_FREE(z);
return exit_status;
}

10.2 Program Data

nag_zhpevd (f08gqc) Example Program Data
4
Nag_Upper
(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)
Nag_EigVecs

10.3 Program Results

nag_zhpevd (f08gqc) Example Program Results

Eigenvalues
1 -4.2443
2 -0.6886
3 1.1412
4 13.7916

Eigenvectors
1 2 3 4
1 1.0000 1.0000 1.0000 1.0000
0.0000 0.0000 -0.0000 -0.0000
2 0.6022 -0.7703 0.0516 1.1508
-0.7483 -0.1746 1.2795 -0.0404
3 -0.6540 0.4559 -1.1962 1.3404
-0.7642 0.4892 -0.2954 0.2188
4 -0.9197 -0.3464 0.7876 1.3674
0.7044 -0.4448 -0.5075 0.8207