NAG C 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

```
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 Λ is a real diagonal matrix whose diagonal elements are the eigenvalues λ_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

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

5 Parameters

1: **order** – Nag_OrderType *Input*

On entry: the **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 **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: **order** = **Nag_RowMajor** or **Nag_ColMajor**.

2: **job** – Nag_JobType *Input*

On entry: indicates whether eigenvectors are computed as follows:

if **job** = **Nag_DoNothing**, only eigenvalues are computed;

if **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 as follows:

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**.

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 $\max(1, \mathbf{n} \times (\mathbf{n}+1)/2)$.

On entry: the Hermitian matrix A, packed by rows or columns. The storage of elements a_{ij} depends on the **order** and **uplo** parameters 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: A 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 $\max(1, \mathbf{n})$.

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
$\max(1, \mathbf{pdz} \times \mathbf{n})$ when **job** $=$ **Nag_EigVecs**;
1 when **job** $=$ **Nag_DoNothing**.

If **order** $=$ **Nag_ColMajor**, the (i,j)th element of the matrix Z is stored in **z**$[(j-1) \times$ **pdz** $+ i - 1]$ and if **order** $=$ **Nag_RowMajor**, the (i,j)th element of the matrix Z is stored in **z**$[(i-1) \times$ **pdz** $+ j - 1]$.

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 matrix row or column elements (depending on the value of **order**) in the array **z**.

Constraints:

if **job** $=$ **Nag_EigVecs**, **pdz** $\geq \max(1, \mathbf{n})$;
if **job** $=$ **Nag_DoNothing**, **pdz** ≥ 1.

9: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

On entry, **pdz** = ⟨*value*⟩.
Constraint: **pdz** > 0.

NE_ENUM_INT_2

On entry, **job** = ⟨*value*⟩, **n** = ⟨*value*⟩, **pdz** = ⟨*value*⟩.
Constraint: if **job** = **Nag_EigVecs**, **pdz** $\geq \max(1, \mathbf{n})$;
if **job** = **Nag_DoNothing**, **pdz** ≥ 1.

NE_CONVERGENCE

The algorithm failed to converge, ⟨*value*⟩ elements of an intermediate tridiagonal form did not converge to zero.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter ⟨*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

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

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

and ϵ is the ***machine precision***.

8 Further Comments

The real analogue of this function is nag_dspevd (f08gcc).

9 Example

To compute all the eigenvalues and eigenvectors of the Hermitian matrix A, where

$$A = \begin{pmatrix} 1.0 + 0.0i & 2.0 - 1.0i & 3.0 - 1.0i & 4.0 - 1.0i \\ 2.0 + 1.0i & 2.0 + 0.0i & 3.0 - 2.0i & 4.0 - 2.0i \\ 3.0 + 1.0i & 3.0 + 2.0i & 3.0 + 0.0i & 4.0 - 3.0i \\ 4.0 + 1.0i & 4.0 + 2.0i & 4.0 + 3.0i & 4.0 + 0.0i \end{pmatrix}.$$

9.1 Program Text

```
/* nag_zhpevd (f08gqc) 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  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    uplo_char[2], job_char[2];
  Complex *ap=0, *z=0;
  double  *w=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

  INIT_FAIL(fail);
  Vprintf("f08gqc Example Program Results\n\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n] ");
  Vscanf("%ld%*[^\n] ", &n);
  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)) )
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  /* Read whether Upper or Lower part of A is stored */
  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;
   }
  /* Read A from data file */
  if (uplo == Nag_Upper)
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = i; j <= n; ++j)
            {
              Vscanf(" ( %lf , %lf )", &A_UPPER(i,j).re,
                     &A_UPPER(i,j).im);
            }
```

```
        }
      Vscanf("%*[^\n] ");
    }
  else
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = 1; j <= i; ++j)
            {
              Vscanf(" ( %lf , %lf )", &A_LOWER(i,j).re,
                     &A_LOWER(i,j).im);
            }
        }
      Vscanf("%*[^\n] ");
    }
  /* Read type of job to be performed */
  Vscanf(" ' %1s '%*[^\n] ", job_char);
  if (*(unsigned char *)job_char == 'V')
    job = Nag_EigVecs;
  else
    job = Nag_DoNothing;
  /* Calculate all the eigenvalues and eigenvectors of A */
  f08gqc(order, job, uplo, n, ap, w, z, pdz, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f08gqc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Print eigenvalues and eigenvectors */
  Vprintf("Eigenvalues\n");
  for (i = 0; i < n; ++i)
    Vprintf("   %5ld     %8.4f\n",i+1,w[i]);
  Vprintf("\n");
  x04dbc(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)
    {
      Vprintf("Error from x04dbc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
 END:
  if (ap) NAG_FREE(ap);
  if (w) NAG_FREE(w);
  if (z) NAG_FREE(z);
  return exit_status;
}
```

9.2 Program Data

```
F08GQC Example Program Data
  4                                                    :Value of N
  'U'                                                  :Value of UPLO
  (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
  'V'                                                  :Value of JOB
```

9.3 Program Results

```
f08gqc Example Program Results

Eigenvalues
       1      -4.2443
       2      -0.6886
       3       1.1412
```

```
       4      13.7916

 Eigenvectors
         1       2       3       4
 1  0.4836  0.6470 -0.4456 -0.3859
   -0.0000 -0.0000 -0.0000  0.0000

 2  0.2912 -0.4984 -0.0230 -0.4441
   -0.3618 -0.1130 -0.5702  0.0156

 3 -0.3163  0.2949  0.5331 -0.5173
   -0.3696  0.3165  0.1317 -0.0844

 4 -0.4447 -0.2241 -0.3510 -0.5277
    0.3406 -0.2878  0.2261 -0.3168
```