NAG Library Function Document

nag_zhpgvd (f08tqc)

1 Purpose

nag_zhpgvd (f08tqc) computes all the eigenvalues and, optionally, the eigenvectors of a complex generalized Hermitian-definite eigenproblem, of the form

$$Az = \lambda Bz, \quad ABz = \lambda z \quad \text{or} \quad BAz = \lambda z,$$

where A and B are Hermitian, stored in packed format, and B is also positive definite. If eigenvectors are desired, it uses a divide-and-conquer algorithm.

2 Specification

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

```
void nag_zhpgvd (Nag_OrderType order, Integer itype, Nag_JobType job,
     Nag_UploType uplo, Integer n, Complex ap[], Complex bp[], double w[],
     Complex z[], Integer pdz, NagError *fail)
```

3 Description

nag_zhpgvd (f08tqc) first performs a Cholesky factorization of the matrix B as $B = U^{\mathrm{H}}U$, when **uplo** = Nag_Upper or $B = LL^{\mathrm{H}}$, when **uplo** = Nag_Lower. The generalized problem is then reduced to a standard symmetric eigenvalue problem

$$Cx = \lambda x,$$

which is solved for the eigenvalues and, optionally, the eigenvectors; the eigenvectors are then backtransformed to give the eigenvectors of the original problem.

For the problem $Az = \lambda Bz$, the eigenvectors are normalized so that the matrix of eigenvectors, z, satisfies

$$Z^{\mathrm{H}}AZ = \Lambda \quad \text{and} \quad Z^{\mathrm{H}}BZ = I,$$

where Λ is the diagonal matrix whose diagonal elements are the eigenvalues. For the problem $ABz = \lambda z$ we correspondingly have

$$Z^{-1}AZ^{-\mathrm{H}} = \Lambda \quad \text{and} \quad Z^{\mathrm{H}}BZ = I,$$

and for $BAz = \lambda z$ we have

$$Z^{\mathrm{H}}AZ = \Lambda \quad \text{and} \quad Z^{\mathrm{H}}B^{-1}Z = I.$$

4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia http://www.netlib.org/lapack/lug

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

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: **itype** – Integer *Input*

On entry: specifies the problem type to be solved.

itype = 1
$Az = \lambda Bz$.

itype = 2
$ABz = \lambda z$.

itype = 3
$BAz = \lambda z$.

Constraint: **itype** = 1, 2 or 3.

3: **job** – Nag_JobType *Input*

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.

4: **uplo** – Nag_UploType *Input*

On entry: if **uplo** = Nag_Upper, the upper triangles of A and B are stored.

If **uplo** = Nag_Lower, the lower triangles of A and B are stored.

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

5: **n** – Integer *Input*

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

Constraint: $\mathbf{n} \geq 0$.

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

7: **bp**[*dim*] – Complex *Input/Output*

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

On entry: the upper or lower triangle of the n by n Hermitian matrix B, packed by rows or columns.

The storage of elements B_{ij} depends on the **order** and **uplo** arguments as follows:

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

On exit: the triangular factor U or L from the Cholesky factorization $B = U^{\mathrm{H}}U$ or $B = LL^{\mathrm{H}}$, in the same storage format as B.

8: **w**[**n**] – double *Output*

On exit: the eigenvalues in ascending order.

9: **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_DoBoth;
1 otherwise.

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

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

On exit: if **job** = Nag_DoBoth, **z** contains the matrix Z of eigenvectors. The eigenvectors are normalized as follows:

if **itype** = 1 or 2, $Z^{\mathrm{H}}BZ = I$;

if **itype** = 3, $Z^{\mathrm{H}}B^{-1}Z = I$.

If **job** = Nag_EigVals, **z** is not referenced.

10: **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_DoBoth, **pdz** $\geq \max(1, \mathbf{n})$;
otherwise **pdz** ≥ 1.

11: **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 $\langle value \rangle$ had an illegal value.

NE_CONVERGENCE

The algorithm failed to converge; $\langle value \rangle$ off-diagonal elements of an intermediate tridiagonal form did not converge to zero.

NE_ENUM_INT_2

On entry, **job** $= \langle value \rangle$, **pdz** $= \langle value \rangle$ and **n** $= \langle value \rangle$.
Constraint: if **job** $=$ Nag_DoBoth, **pdz** $\geq \max(1, \mathbf{n})$;
otherwise **pdz** ≥ 1.

NE_INT

On entry, **itype** $= \langle value \rangle$.
Constraint: **itype** $= 1$, 2 or 3.

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 0.

On entry, **pdz** $= \langle value \rangle$.
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_MAT_NOT_POS_DEF

If **fail.errnum** $= \mathbf{n} + \langle value \rangle$, for $1 \leq \langle value \rangle \leq \mathbf{n}$, then the leading minor of order $\langle value \rangle$ of B is not positive definite. The factorization of B could not be completed and no eigenvalues or eigenvectors were computed.

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

If B is ill-conditioned with respect to inversion, then the error bounds for the computed eigenvalues and vectors may be large, although when the diagonal elements of B differ widely in magnitude the eigenvalues and eigenvectors may be less sensitive than the condition of B would suggest. See Section 4.10 of Anderson *et al.* (1999) for details of the error bounds.

The example program below illustrates the computation of approximate error bounds.

8 Parallelism and Performance

nag_zhpgvd (f08tqc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_zhpgvd (f08tqc) 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_dspgvd (f08tcc).

10 Example

This example finds all the eigenvalues and eigenvectors of the generalized Hermitian eigenproblem $ABz = \lambda z$, where

$$A = \begin{pmatrix} -7.36 & 0.77 - 0.43i & -0.64 - 0.92i & 3.01 - 6.97i \\ 0.77 + 0.43i & 3.49 & 2.19 + 4.45i & 1.90 + 3.73i \\ -0.64 + 0.92i & 2.19 - 4.45i & 0.12 & 2.88 - 3.17i \\ 3.01 + 6.97i & 1.90 - 3.73i & 2.88 + 3.17i & -2.54 \end{pmatrix}$$

and

$$B = \begin{pmatrix} 3.23 & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 \end{pmatrix},$$

together with an estimate of the condition number of B, and approximate error bounds for the computed eigenvalues and eigenvectors.

The example program for nag_zhpgv (f08tnc) illustrates solving a generalized Hermitian eigenproblem of the form $Az = \lambda Bz$.

10.1 Program Text

```
/* nag_zhpgvd (f08tqc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */

#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagf08.h>
#include <nagf16.h>
#include <nagx02.h>

int main(void)
{
  /* Scalars */
  double        anorm, bnorm, eps, rcond, rcondb, t1;
  Integer       i, j, n;
  Integer       exit_status = 0;
```

```
  /* Arrays */
  Complex       *ap = 0, *bp = 0;
  Complex       dummy[1];
  double        *eerbnd = 0, *w = 0;
  char          nag_enum_arg[40];

  /* Nag Types */
  NagError      fail;
  Nag_OrderType order;
  Nag_UploType  uplo;

#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]
#define B_UPPER(I, J) bp[J*(J-1)/2 + I - 1]
#define B_LOWER(I, J) bp[(2*n-J)*(J-1)/2 + I - 1]
  order = Nag_ColMajor;
#else
#define A_UPPER(I, J) ap[(2*n-I)*(I-1)/2 + J - 1]
#define A_LOWER(I, J) ap[I*(I-1)/2 + J - 1]
#define B_UPPER(I, J) bp[(2*n-I)*(I-1)/2 + J - 1]
#define B_LOWER(I, J) bp[I*(I-1)/2 + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);


  printf("nag_zhpgvd (f08tqc) Example Program Results\n\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n]", &n);
#else
  scanf("%"NAG_IFMT"%*[^\n]", &n);
#endif
  if (n < 0)
    {
      printf("Invalid n\n");
      exit_status = 1;
      goto END;;
    }
#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);

  /* Allocate memory */
  if (!(ap     = NAG_ALLOC(n*(n+1)/2, Complex)) ||
      !(bp     = NAG_ALLOC(n*(n+1)/2, Complex)) ||
      !(eerbnd = NAG_ALLOC(n, double)) ||
      !(w      = NAG_ALLOC(n, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read the triangular parts of the matrices A and B 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
      for (i = 1; i <= n; ++i)
        for (j = i; j <= n; ++j)
#ifdef _WIN32
          scanf_s(" ( %lf , %lf )", &B_UPPER(i, j).re, &B_UPPER(i, j).im);
#else
          scanf(" ( %lf , %lf )", &B_UPPER(i, j).re, &B_UPPER(i, j).im);
#endif
    }
  else if (uplo == Nag_Lower)
    {
      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
#ifdef _WIN32
      scanf_s("%*[^\n]");
#else
      scanf("%*[^\n]");
#endif
      for (i = 1; i <= n; ++i)
        for (j = 1; j <= i; ++j)
#ifdef _WIN32
          scanf_s(" ( %lf , %lf )", &B_LOWER(i, j).re, &B_LOWER(i, j).im);
#else
          scanf(" ( %lf , %lf )", &B_LOWER(i, j).re, &B_LOWER(i, j).im);
#endif
    }
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  /* Compute the one-norms of the symmetric matrices A and B
   * using nag_zhp_norm (f16udc).
   */
  nag_zhp_norm(order, Nag_OneNorm, uplo, n, ap, &anorm, &fail);
  nag_zhp_norm(order, Nag_OneNorm, uplo, n, bp, &bnorm, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zhp_norm (f16udc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Solve the generalized symmetric eigenvalue problem
   * A*B*x = lambda*x (itype = 2) using nag_zhpgvd (f08tqc).
   */
  nag_zhpgvd(order, 2, Nag_EigVals, uplo, n, ap, bp, w, dummy, 1, &fail);

  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zhpgvd (f08tqc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
```

```
  /* Print eigensolution */
  printf("Eigenvalues\n   ");
  for (j = 0; j < n; ++j) printf(" %11.4f%s", w[j], j%6 == 5?"\n":"");
  printf("\n\n");

  /* Estimate the reciprocal condition number of the Cholesky factor of B.
   * nag_ztpcon (f07uuc).
   * Note that: cond(B) = 1/(rcond*rcond).
   */
  nag_ztpcon(order, Nag_OneNorm, uplo, Nag_NonUnitDiag, n, bp, &rcond, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_ztpcon (f07uuc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Print the reciprocal condition number of B */
  rcondb = rcond * rcond;
  printf("Estimate of reciprocal condition number for B\n    %12.1e\n", rcondb);

  /* Get the machine precision, using nag_machine_precision (x02ajc) */
  eps = nag_machine_precision;
  if (rcond < eps)
    {
      printf("\nB is very ill-conditioned, error estimates have not been"
             " computed.\n");
      goto END;
    }

  t1 = anorm * bnorm;
  for (i = 0; i < n; ++i) eerbnd[i] = eps * (t1 + fabs(w[i])/ rcondb);

  /* Print the approximate error bounds for the eigenvalues */
  printf("\nError estimates for the eigenvalues\n    ");
  for (i = 0; i < n; ++i)  printf(" %11.1e%s", eerbnd[i], i%6 == 5?"\n":"");
  printf("\n");

 END:
  NAG_FREE(ap);
  NAG_FREE(bp);
  NAG_FREE(eerbnd);
  NAG_FREE(w);

  return exit_status;
}
```

10.2 Program Data

```
nag_zhpgvd (f08tqc) Example Program Data

   4                                                         : n

   Nag_Upper                                                 : uplo

 (-7.36, 0.00) ( 0.77, -0.43) (-0.64, -0.92) ( 3.01, -6.97)
               ( 3.49,  0.00) ( 2.19,  4.45) ( 1.90,  3.73)
                              ( 0.12,  0.00) ( 2.88, -3.17)
                                             (-2.54,  0.00) : matrix A

 ( 3.23, 0.00) ( 1.51, -1.92) ( 1.90,  0.84) ( 0.42,  2.50)
               ( 3.58,  0.00) (-0.23,  1.11) (-1.18,  1.37)
                              ( 4.09,  0.00) ( 2.33, -0.14)
                                             ( 4.29,  0.00) : matrix B
```

10.3 Program Results

```
nag_zhpgvd (f08tqc) Example Program Results

Eigenvalues
       -61.7321     -6.6195      0.0725     43.1883

Estimate of reciprocal condition number for B
         2.5e-03

Error estimates for the eigenvalues
         2.7e-12     3.1e-13     2.6e-14     1.9e-12
```