

## NAG Toolbox

### nag\_lapack\_zhegvd (f08sq)

#### 1 Purpose

nag\_lapack\_zhegvd (f08sq) 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 and  $B$  is also positive definite. If eigenvectors are desired, it uses a divide-and-conquer algorithm.

#### 2 Syntax

```
[a, b, w, info] = nag_lapack_zhegvd(itype, jobz, uplo, a, b, 'n', n)
[a, b, w, info] = f08sq(itype, jobz, uplo, a, b, 'n', n)
```

#### 3 Description

nag\_lapack\_zhegvd (f08sq) first performs a Cholesky factorization of the matrix  $B$  as  $B = U^H U$ , when **uplo** = 'U' or  $B = LL^H$ , when **uplo** = 'L'. 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

$$Z^H A Z = \Lambda \quad \text{and} \quad Z^H B Z = I,$$

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

$$Z^{-1} A Z^{-H} = \Lambda \quad \text{and} \quad Z^H B Z = I,$$

and for  $BAz = \lambda z$  we have

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

#### 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 Parameters

### 5.1 Compulsory Input Parameters

1: **itype** – INTEGER

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.

2: **jobz** – CHARACTER(1)

Indicates whether eigenvectors are computed.

**jobz** = 'N'  
 Only eigenvalues are computed.

**jobz** = 'V'  
 Eigenvalues and eigenvectors are computed.

*Constraint:* **jobz** = 'N' or 'V'.

3: **uplo** – CHARACTER(1)

If **uplo** = 'U', the upper triangles of  $A$  and  $B$  are stored.

If **uplo** = 'L', the lower triangles of  $A$  and  $B$  are stored.

*Constraint:* **uplo** = 'U' or 'L'.

4: **a**(*lda*,:) – COMPLEX (KIND=nag\_wp) array

The first dimension of the array **a** must be at least  $\max(1, \mathbf{n})$ .

The second dimension of the array **a** must be at least  $\max(1, \mathbf{n})$ .

The  $n$  by  $n$  Hermitian matrix  $A$ .

If **uplo** = 'U', the upper triangular part of  $a$  must be stored and the elements of the array below the diagonal are not referenced.

If **uplo** = 'L', the lower triangular part of  $a$  must be stored and the elements of the array above the diagonal are not referenced.

5: **b**(*ldb*,:) – COMPLEX (KIND=nag\_wp) array

The first dimension of the array **b** must be at least  $\max(1, \mathbf{n})$ .

The second dimension of the array **b** must be at least  $\max(1, \mathbf{n})$ .

The  $n$  by  $n$  Hermitian matrix  $B$ .

If **uplo** = 'U', the upper triangular part of  $b$  must be stored and the elements of the array below the diagonal are not referenced.

If **uplo** = 'L', the lower triangular part of  $b$  must be stored and the elements of the array above the diagonal are not referenced.

## 5.2 Optional Input Parameters

1: **n** – INTEGER

*Default:* the first dimension of the arrays **a**, **b** and the second dimension of the arrays **a**, **b**. (An error is raised if these dimensions are not equal.)

*n*, the order of the matrices *A* and *B*.

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

## 5.3 Output Parameters

1: **a**(*lda*,:) – COMPLEX (KIND=nag\_wp) array

The first dimension of the array **a** will be  $\max(1, \mathbf{n})$ .

The second dimension of the array **a** will be  $\max(1, \mathbf{n})$ .

If **jobz** = 'V', **a** contains the matrix *Z* of eigenvectors. The eigenvectors are normalized as follows:

if **itype** = 1 or 2,  $Z^H B Z = I$ ;

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

If **jobz** = 'N', the upper triangle (if **uplo** = 'U') or the lower triangle (if **uplo** = 'L') of **a**, including the diagonal, is overwritten.

2: **b**(*ldb*,:) – COMPLEX (KIND=nag\_wp) array

The first dimension of the array **b** will be  $\max(1, \mathbf{n})$ .

The second dimension of the array **b** will be  $\max(1, \mathbf{n})$ .

The triangular factor *U* or *L* from the Cholesky factorization  $B = U^H U$  or  $B = L L^H$ .

3: **w**(**n**) – REAL (KIND=nag\_wp) array

The eigenvalues in ascending order.

4: **info** – INTEGER

**info** = 0 unless the function detects an error (see Section 6).

## 6 Error Indicators and Warnings

**info** =  $-i$

If **info** =  $-i$ , parameter *i* had an illegal value on entry. The parameters are numbered as follows:

1: **itype**, 2: **jobz**, 3: **uplo**, 4: **n**, 5: **a**, 6: **lda**, 7: **b**, 8: **ldb**, 9: **w**, 10: **work**, 11: **lwork**, 12: **rwork**, 13: **lrwork**, 14: **iwork**, 15: **liwork**, 16: **info**.

It is possible that **info** refers to a parameter that is omitted from the MATLAB interface. This usually indicates that an error in one of the other input parameters has caused an incorrect value to be inferred.

**info** = 1 to **n**

If **info** = *i*, nag\_lapack\_zheevd (f08fq) failed to converge; *i* *i* off-diagonal elements of an intermediate tridiagonal form did not converge to zero.

**info** > **n**

nag\_lapack\_zpotrf (f07fr) returned an error code; i.e., if **info** = **n** + *i*, for  $1 \leq i \leq \mathbf{n}$ , then the leading minor of order *i* of *B* is not positive definite. The factorization of *B* could not be completed and no eigenvalues or eigenvectors were computed.

## 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 Further Comments

The total number of floating-point operations is proportional to  $n^3$ .

The real analogue of this function is nag\_lapack\_dsygvd (f08sc).

## 9 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\_lapack\_zhegv (f08sn) illustrates solving a generalized Hermitian eigenproblem of the form  $Az = \lambda Bz$ .

### 9.1 Program Text

```
function f08sq_example

fprintf('f08sq example results\n\n');

% Upper triangular parts of Hermitian matrix A and positive definite matrix B
uplo = 'Upper';
n = nag_int(4);
a = [-7.36 + 0i, 0.77 - 0.43i, -0.64 - 0.92i, 3.01 - 6.97i;
      0 + 0i, 3.49 + 0i, 2.19 + 4.45i, 1.90 + 3.73i;
      0 + 0i, 0 + 0i, 0.12 + 0i, 2.88 - 3.17i;
      0 + 0i, 0 + 0i, 0 + 0i, -2.54 + 0i];
b = [ 3.23 + 0i, 1.51 - 1.92i, 1.90 + 0.84i, 0.42 + 2.50i;
      0 + 0i, 3.58 + 0i, -0.23 + 1.11i, -1.18 + 1.37i;
      0 + 0i, 0 + 0i, 4.09 + 0i, 2.33 - 0.14i;
      0 + 0i, 0 + 0i, 0 + 0i, 4.29 + 0i];

anorm = norm(a+a'-diag(diag(a)),1);
bnorm = norm(b+b'-diag(diag(b)),1);
```

```

% Generalized eigenvalues and eigenvectors for problem ABz = lambda z.
itype = nag_int(2);
jobz = 'Vectors';
[Z, L, w, info] = f08sq( ...
    itype, jobz, uplo, a, b);

% Normalize: largest elements are real (preserving Z`HBZ = I)
for i = 1:n
    [~,k] = max(abs(real(Z(:,i)))+abs(imag(Z(:,i))));
    Z(:,i) = Z(:,i)*conj(Z(k,i))/abs(Z(k,i));
end

disp('Eigenvalues');
disp(w');
disp('Eigenvectors');
disp(Z);

% Get reciprocal condition number
[rcond, info] = f07tu( ...
    '1', uplo, 'N', L);

rcondb = rcond^2;
fprintf('Estimate of reciprocal condition number for B\n%12.1e\n', rcondb);

% Eigenvector condition numbers
[rcondz, info] = f08fl( ...
    'Eigenvectors', n, n, w);

% Error bounds
t1 = 1/rcond;
t2 = x02aj*t1;
t3 = anorm*bnorm;
errbnd(1:n) = x02aj*t3 + (x02aj/rcondb).*abs(w);
zerrbd(1:n) = t1*t2 + (t2*t3)./rcondz;

fprintf('\n');
disp('Error estimate for the eigenvalues');
fprintf('%12.1e',errbnd);
fprintf('\n\n');
disp('Error estimates for the eigenvectors');
fprintf('%12.1e',zerrbd);
fprintf('\n');

```

## 9.2 Program Results

f08sq example results

Eigenvalues

-61.7321   -6.6195   0.0725   43.1883

Eigenvectors

```

0.3903 + 0.0000i   -0.1560 - 0.0404i   2.2909 + 0.0000i   -0.1943 - 0.0690i
-0.1814 + 0.0114i   -0.1552 - 0.3651i   -0.5042 - 0.7120i   0.3884 + 0.0000i
0.0438 + 0.0338i   0.5364 + 0.0000i   -1.2701 - 0.4547i   0.0657 - 0.2095i
-0.2221 - 0.2272i   -0.1298 - 0.1880i   0.5706 + 1.3132i   0.2924 - 0.0675i

```

Estimate of reciprocal condition number for B

2.5e-03

Error estimate for the eigenvalues

2.7e-12   3.1e-13   2.6e-14   1.9e-12

Error estimates for the eigenvectors

5.2e-14   1.1e-13   1.1e-13   5.4e-14

---