Accurate eigenvalue decomposition of arrowhead matrices and applications

Nevena Jakovčević Stor
FESB, University of Split

joint work with Ivan Slapničar and Jesse Barlow

IWASEP9
June 4th, 2012.

We present a new algorithm (*aheig*) for computing eigenvalue decomposition of real symmetric arrowhead matrix.

We present a new algorithm (*aheig*) for computing eigenvalue decomposition of real symmetric arrowhead matrix.

- Under certain conditions, the *aheig* algorithm computes all eigenvalues and all components of corresponding eigenvectors with high rel. acc. in $O(n^2)$ operations.

We present a new algorithm (*aheig*) for computing eigenvalue decomposition of real symmetric arrowhead matrix.

- Under certain conditions, the *aheig* algorithm computes all eigenvalues and all components of corresponding eigenvectors with high rel. acc. in $O(n^2)$ operations.

We present a new algorithm (*aheig*) for computing eigenvalue decomposition of real symmetric arrowhead matrix.

- Under certain conditions, the *aheig* algorithm computes all eigenvalues and all components of corresponding eigenvectors with high rel. acc. in $O(n^2)$ operations.
- The algorithm is based on shift-and-invert technique and limited finite $O(n)$ use of double precision arithmetic when necessary.

We present a new algorithm (*aheig*) for computing eigenvalue decomposition of real symmetric arrowhead matrix.

- Under certain conditions, the *aheig* algorithm computes all eigenvalues and all components of corresponding eigenvectors with high rel. acc. in $O(n^2)$ operations.
- The algorithm is based on shift-and-invert technique and limited finite $O(n)$ use of double precision arithmetic when necessary.
- Orthogonality of eigenvectors computed by *aheig* algorithm is consequence of their accuracy (no need for follow-up orthogonalization).

We present a new algorithm (*aheig*) for computing eigenvalue decomposition of real symmetric arrowhead matrix.

- Under certain conditions, the *aheig* algorithm computes all eigenvalues and all components of corresponding eigenvectors with high rel. acc. in $O(n^2)$ operations.
- The algorithm is based on shift-and-invert technique and limited finite $O(n)$ use of double precision arithmetic when necessary.
- Orthogonality of eigenvectors computed by *aheig* algorithm is consequence of their accuracy (no need for follow-up orthogonalization).
- Each eigenvalue and corresponding eigenvector can be computed independently, which makes the algorithm adaptable for parallel computing.

We present a new algorithm (*aheig*) for computing eigenvalue decomposition of real symmetric arrowhead matrix.

- Under certain conditions, the *aheig* algorithm computes all eigenvalues and all components of corresponding eigenvectors with high rel. acc. in $O(n^2)$ operations.
- The algorithm is based on shift-and-invert technique and limited finite $O(n)$ use of double precision arithmetic when necessary.
- Orthogonality of eigenvectors computed by *aheig* algorithm is consequence of their accuracy (no need for follow-up orthogonalization).
- Each eigenvalue and corresponding eigenvector can be computed independently, which makes the algorithm adaptable for parallel computing.
- We also present the applications of *aheig* algorithm to hermitian arrowhead, symmetric tridiagonal matrices and diagonal plus rank one matrices.

Outline

1. Introduction
2. The idea of the *aheig* algorithm
3. Example
4. Accuracy of the *aheig* algorithm
5. Application to hermitian arrowhead matrices
6. Application to tridiagonal symmetric matrices
7. Application to "diagonal + rank-one" matrices $(D + uu^T)$

Let

$$A = \begin{bmatrix} D & z \\ z^T & \alpha \end{bmatrix}$$

be $n \times n$ real symmetric arrowhead matrix, where

$$D = \operatorname{diag}(d_1, d_2, \ldots, d_{n-1}),\ z = \begin{bmatrix} \zeta_1 & \zeta_2 & \cdots & \zeta_{n-1} \end{bmatrix}^T,\ \alpha \in \mathbb{R}$$

and

Let

$$A = \begin{bmatrix} D & z \\ z^T & \alpha \end{bmatrix}$$

be $n \times n$ real symmetric arrowhead matrix, where

$$D = \operatorname{diag}(d_1, d_2, \ldots, d_{n-1}),\ z = \begin{bmatrix} \zeta_1 & \zeta_2 & \cdots & \zeta_{n-1} \end{bmatrix}^T,\ \alpha \in \mathbb{R}$$

and

$$A = V \Lambda V^T$$

eigenvalue decomposition of A, where

$$\Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

and

$$V = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix}.$$

We assume that A is irreducible:

$$\zeta_i \neq 0, \forall\ i, \ \ d_i \neq d_j, \forall i \neq j.$$

Without loss of generality we can assume that $\zeta_i > 0, \forall i$ and that diagonal elements of D are ordered, that is

$$d_1 > d_2 > \cdots > d_{n-1}.$$

We assume that A is irreducible:

$$\zeta_i \neq 0, \forall\ i, \;\; d_i \neq d_j, \forall i \neq j.$$

Without loss of generality we can assume that $\zeta_i > 0, \forall i$ and that diagonal elements of D are ordered, that is

$$d_1 > d_2 > \cdots > d_{n-1}.$$

The assumptions on D imply the interlacing property

$$\lambda_1 > d_1 > \lambda_2 > d_2 > \cdots > d_{n-2} > \lambda_{n-1} > d_{n-1} > \lambda_n$$

where $\lambda_i, i = 1, \ldots, n$, are eigenvalues of matrix A.

The eigenvalues of A are the zeros of function:

$$\varphi_A(\lambda) = \alpha - \lambda - \sum_{i=1}^{n-1} \frac{\zeta_i^2}{d_i - \lambda} = \alpha - \lambda - z^T (D - \lambda I)^{-1} z.$$

The eigenvalues of A are the zeros of function:

$$\varphi_A(\lambda) = \alpha - \lambda - \sum_{i=1}^{n-1} \frac{\zeta_i^2}{d_i - \lambda} = \alpha - \lambda - z^T (D - \lambda I)^{-1} z.$$

and the eigenvectors are given by:

$$v_i = \frac{x_i}{\|x_i\|_2}, \; x_i = \begin{bmatrix} (D - \lambda_i I)^{-1} z \\ -1 \end{bmatrix}, \; i = 1, \ldots, n.$$

The eigenvalues of A are the zeros of function:

$$\varphi_A(\lambda) = \alpha - \lambda - \sum_{i=1}^{n-1} \frac{\zeta_i^2}{d_i - \lambda} = \alpha - \lambda - z^T (D - \lambda I)^{-1} z.$$

and the eigenvectors are given by:

$$v_i = \frac{x_i}{\|x_i\|_2},\ x_i = \begin{bmatrix} (D - \lambda_i I)^{-1} z \\ -1 \end{bmatrix},\ i = 1, \ldots, n.$$

Problem: λ_i is not exact $\Longrightarrow$ v_i may not be orthogonal.

The existing algorithms obtain orthogonal eigenvectors with the following procedure.

- Compute eigenvalues of A.
- Construct new matrix $\widehat{A}$ (inverse problem) with prescribed eigenvalues $\widehat{\lambda}$, and diagonal matrix D (that is compute new $\widehat{z}$ and $\widehat{\alpha}$).
- Compute eigenvectors of $\widehat{A}$ with the previous formula.
- This way computed, eigenvectors are not the exact eigenvectors of starting matrix A, they are exact eigenvectors of matrix

$$\widehat{A} = \begin{bmatrix} D & \widehat{z} \\ \widehat{z}^T & \widehat{\alpha} \end{bmatrix},$$

$$|\widehat{z}_i| = \sqrt{\left(d_i - \widehat{\lambda}_n\right)\left(\widehat{\lambda}_1 - d_i\right) \prod_{j=2}^{i} \frac{\left(\widehat{\lambda}_j - d_i\right)}{\left(d_{j-1} - d_i\right)} \prod_{j=i+1}^{n-1} \frac{\left(\widehat{\lambda}_j - d_i\right)}{\left(d_j - d_i\right)}},$$

$$\widehat{\alpha} = \widehat{\lambda}_n + \sum_{j=1}^{n-1} \left(\widehat{\lambda}_j - d_j\right).$$

Our algorithm has a different concept. Accuracy of the eigenvectors and their orthogonality follow from accuracy of computed eigenvalues. Thus, there is no need for follow-up orthogonalization.

Our algorithm has a different concept. Accuracy of the eigenvectors and their orthogonality follow from accuracy of computed eigenvalues. Thus, there is no need for follow-up orthogonalization.

Let d_i be the diagonal element (pole) in A which is closest to λ.

Our algorithm has a different concept. Accuracy of the eigenvectors and their orthogonality follow from accuracy of computed eigenvalues. Thus, there is no need for follow-up orthogonalization.

Let d_i be the diagonal element (pole) in A which is closest to λ. From the interlacing property it follows that either

$$\lambda = \lambda_i$$

or

$$\lambda = \lambda_{i+1}.$$

Let A_i be the shifted matrix,

$$A_i = A - d_i I = \begin{bmatrix} D_1 & 0 & 0 & z_1 \\ 0 & 0 & 0 & \zeta_i \\ 0 & 0 & D_2 & z_2 \\ z_1^T & \zeta_i & z_2^T & a \end{bmatrix},$$

Let A_i be the shifted matrix,

$$A_i = A - d_i I = \begin{bmatrix} D_1 & 0 & 0 & z_1 \\ 0 & 0 & 0 & \zeta_i \\ 0 & 0 & D_2 & z_2 \\ z_1^T & \zeta_i & z_2^T & a \end{bmatrix},$$

where

$$\begin{aligned} D_1 &= \operatorname{diag}(d_1 - d_i, \ldots, d_{i-1} - d_i) \text{ positive definite}, \\ D_2 &= \operatorname{diag}(d_{i+1} - d_i, \ldots, d_{n-1} - d_i) \text{ negative definite}, \\ z_1 &= \begin{bmatrix} \zeta_1 & \zeta_2 & \cdots & \zeta_{i-1} \end{bmatrix}^T, \\ z_2 &= \begin{bmatrix} \zeta_{i+1} & \zeta_{i+2} & \cdots & \zeta_{n-1} \end{bmatrix}^T, \\ a &= \alpha - d_i. \end{aligned}$$

Let A_i be the shifted matrix,

$$A_i = A - d_i I = \begin{bmatrix} D_1 & 0 & 0 & z_1 \\ 0 & 0 & 0 & \zeta_i \\ 0 & 0 & D_2 & z_2 \\ z_1^T & \zeta_i & z_2^T & a \end{bmatrix},$$

where

$$\begin{aligned} D_1 &= \operatorname{diag}(d_1 - d_i, \ldots, d_{i-1} - d_i) \text{ positive definite}, \\ D_2 &= \operatorname{diag}(d_{i+1} - d_i, \ldots, d_{n-1} - d_i) \text{ negative definite}, \\ z_1 &= \begin{bmatrix} \zeta_1 & \zeta_2 & \cdots & \zeta_{i-1} \end{bmatrix}^T, \\ z_2 &= \begin{bmatrix} \zeta_{i+1} & \zeta_{i+2} & \cdots & \zeta_{n-1} \end{bmatrix}^T, \\ a &= \alpha - d_i. \end{aligned}$$

Obviously

$$\mu = \lambda - d_i$$

is an eigenvalue of A_i iff λ is an eigenvalue of A.

Now

$$A_i^{-1} = \begin{bmatrix} D_1^{-1} & w_1 & 0 & 0 \\ w_1^T & b & w_2^T & 1/\zeta_i \\ 0 & w_2 & D_2^{-1} & 0 \\ 0 & 1/\zeta_i & 0 & 0 \end{bmatrix},$$

Now

$$A_i^{-1} = \begin{bmatrix} D_1^{-1} & w_1 & 0 & 0 \\ w_1^T & b & w_2^T & 1/\zeta_i \\ 0 & w_2 & D_2^{-1} & 0 \\ 0 & 1/\zeta_i & 0 & 0 \end{bmatrix},$$

where

$$\begin{aligned} w_1 &= -D_1^{-1} z_1 \frac{1}{\zeta_i}, \\ w_2 &= -D_2^{-1} z_2 \frac{1}{\zeta_i}, \\ b &= \frac{1}{\zeta_i^2}\left(-a + z_1^T D_1^{-1} z_1 + z_2^T D_2^{-1} z_2\right). \end{aligned}$$

λ is the eigenvalue of matrix A which is closest to the pole d_i.

$$\Downarrow$$

λ is the eigenvalue of matrix A which is closest to the pole d_i.

$$\Downarrow$$

μ is the eigenvalue of matrix A_i which is closest to zero.

$$\Downarrow$$

λ is the eigenvalue of matrix A which is closest to the pole d_i.

$$\Downarrow$$

μ is the eigenvalue of matrix A_i which is closest to zero.

$$\Downarrow$$

$$1/\left|\mu\right| = \left\|A_i^{-1}\right\|_2$$

λ is the eigenvalue of matrix A which is closest to the pole d_i.

$$\Downarrow$$

μ is the eigenvalue of matrix A_i which is closest to zero.

$$\Downarrow$$

$$1/|\mu| = \left\|A_i^{-1}\right\|_2$$

μ is well behaved by the standard perturbation theory.

Example 1/2

Let

$$A = \begin{bmatrix} 2\cdot 10^{-3} & 0 & 0 & 0 & 0 & 10^{7} \\ 0 & 10^{-7} & 0 & 0 & 0 & 10^{7} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -10^{-7} & 0 & 10^{7} \\ 0 & 0 & 0 & 0 & -2\cdot 10^{-3} & 10^{7} \\ 10^{7} & 10^{7} & 1 & 10^{7} & 10^{7} & 10^{20} \end{bmatrix},$$

where

$$\begin{aligned} D &= \begin{bmatrix} 2\cdot 10^{-3} & 10^{-7} & 0 & -10^{-7} & -2\cdot 10^{-3} \end{bmatrix}, \\ z &= \begin{bmatrix} 10^{7} & 10^{7} & 1 & 10^{7} & 10^{7} \end{bmatrix} \text{ and} \\ \alpha &= 10^{20}. \end{aligned}$$

Example 2/2

Eigenvalues computed by Matlab *eig*, *aheig* and Mathematica (100 digits) are:

λ_{eig}	λ_{aheig}	λ_{Math}
$1.000000000000000 \cdot 10^{20}$	$1.000000000000000 \cdot 10^{20}$	$1.000000000000000 \cdot 10^{20}$
$1.999001249000113 \cdot 10^{-3}$	$1.999001249000113 \cdot 10^{-3}$	$1.999001249000113 \cdot 10^{-3}$
$4.987562099695390 \cdot 10^{-9}$	$4.987562099722816 \cdot 10^{-9}$	$4.987562099722817 \cdot 10^{-9}$
$-1.000644853973479 \cdot 10^{-20}$	$-9.999999999980001 \cdot 10^{-21}$	$-9.999999999980001 \cdot 10^{-21}$
$-2.004985562101759 \cdot 10^{-6}$	$-2.004985562101718 \cdot 10^{-6}$	$-2.004985562101717 \cdot 10^{-6}$
$-2.001001251000111 \cdot 10^{-3}$	$-2.001001251000111 \cdot 10^{-3}$	$-2.001001251000111 \cdot 10^{-3}$

λ_3 and λ_4 computed by *aheig* are accurate (to the machine precision).

Example 2/2

Eigenvalues computed by Matlab *eig*, *aheig* and Mathematica (100 digits) are:

λ_{eig}	λ_{aheig}	λ_{Math}
$1.000000000000000 \cdot 10^{20}$	$1.000000000000000 \cdot 10^{20}$	$1.000000000000000 \cdot 10^{20}$
$1.999001249000113 \cdot 10^{-3}$	$1.999001249000113 \cdot 10^{-3}$	$1.999001249000113 \cdot 10^{-3}$
$4.987562099695390 \cdot 10^{-9}$	$4.987562099722816 \cdot 10^{-9}$	$4.987562099722817 \cdot 10^{-9}$
$-1.000644853973479 \cdot 10^{-20}$	$-9.999999999980001 \cdot 10^{-21}$	$-9.999999999980001 \cdot 10^{-21}$
$-2.004985562101759 \cdot 10^{-6}$	$-2.004985562101718 \cdot 10^{-6}$	$-2.004985562101717 \cdot 10^{-6}$
$-2.001001251000111 \cdot 10^{-3}$	$-2.001001251000111 \cdot 10^{-3}$	$-2.001001251000111 \cdot 10^{-3}$

λ_3 and λ_4 computed by *aheig* are accurate (to the machine precision).

Example 2/2

Eigenvalues computed by Matlab *eig*, *aheig* and Mathematica (100 digits) are:

λ_{eig}	λ_{aheig}	λ_{Math}
$1.000000000000000 \cdot 10^{20}$	$1.000000000000000 \cdot 10^{20}$	$1.000000000000000 \cdot 10^{20}$
$1.999001249000113 \cdot 10^{-3}$	$1.999001249000113 \cdot 10^{-3}$	$1.999001249000113 \cdot 10^{-3}$
$4.987562099695390 \cdot 10^{-9}$	$4.987562099722816 \cdot 10^{-9}$	$4.987562099722817 \cdot 10^{-9}$
$-1.000644853973479 \cdot 10^{-20}$	$-9.999999999980001 \cdot 10^{-21}$	$-9.999999999980001 \cdot 10^{-21}$
$-2.004985562101759 \cdot 10^{-6}$	$-2.004985562101718 \cdot 10^{-6}$	$-2.004985562101717 \cdot 10^{-6}$
$-2.001001251000111 \cdot 10^{-3}$	$-2.001001251000111 \cdot 10^{-3}$	$-2.001001251000111 \cdot 10^{-3}$

λ_3 and λ_4 computed by *aheig* are accurate (to the machine precision).

Eigenvectors computed by *aheig* are accurate and therefore, orthogonal. For example, let us look at U_4

$U_{4(eig)}$	$U_{4(aheig)}$	$U_{4(Math)}$
$4.999993626151683 \cdot 10^{-11}$	$-4.999999999985000 \cdot 10^{-11}$	$-4.999999999985000 \cdot 10^{-11}$
$9.999999962328609 \cdot 10^{-7}$	$-9.999999999969000 \cdot 10^{-7}$	$-9.999999999969000 \cdot 10^{-7}$
$9.999999999990000 \cdot 10^{-1}$	$-9.999999999989999 \cdot 10^{-1}$	$-9.999999999989999 \cdot 10^{-1}$
$-9.999999964673912 \cdot 10^{-7}$	$9.999999999970999 \cdot 10^{-7}$	$9.999999999970999 \cdot 10^{-7}$
$-5.000006338012225 \cdot 10^{-11}$	$4.999999999985000 \cdot 10^{-11}$	$4.999999999985000 \cdot 10^{-11}$
$-9.999999963825105 \cdot 10^{-21}$	$9.999999999969999 \cdot 10^{-21}$	$9.999999999969999 \cdot 10^{-21}$

Example 2/2

Eigenvalues computed by Matlab *eig*, *aheig* and Mathematica (100 digits) are:

λ_{eig}	λ_{aheig}	λ_{Math}
$1.000000000000000 \cdot 10^{20}$	$1.000000000000000 \cdot 10^{20}$	$1.000000000000000 \cdot 10^{20}$
$1.999001249000113 \cdot 10^{-3}$	$1.999001249000113 \cdot 10^{-3}$	$1.999001249000113 \cdot 10^{-3}$
$4.987562099695390 \cdot 10^{-9}$	$4.987562099722816 \cdot 10^{-9}$	$4.987562099722817 \cdot 10^{-9}$
$-1.000644853973479 \cdot 10^{-20}$	$-9.999999999980001 \cdot 10^{-21}$	$-9.999999999980001 \cdot 10^{-21}$
$-2.004985562101759 \cdot 10^{-6}$	$-2.004985562101718 \cdot 10^{-6}$	$-2.004985562101717 \cdot 10^{-6}$
$-2.001001251000111 \cdot 10^{-3}$	$-2.001001251000111 \cdot 10^{-3}$	$-2.001001251000111 \cdot 10^{-3}$

λ_3 and λ_4 computed by *aheig* are accurate (to the machine precision).

Eigenvectors computed by *aheig* are accurate and therefore, orthogonal. For example, let us look at U_4

$U_{4(eig)}$	$U_{4(aheig)}$	$U_{4(Math)}$
$4.999993626151683 \cdot 10^{-11}$	$-4.999999999985000 \cdot 10^{-11}$	$-4.999999999985000 \cdot 10^{-11}$
$9.999999962328609 \cdot 10^{-7}$	$-9.999999999969000 \cdot 10^{-7}$	$-9.999999999969000 \cdot 10^{-7}$
$9.999999999990000 \cdot 10^{-1}$	$-9.999999999989999 \cdot 10^{-1}$	$-9.999999999989999 \cdot 10^{-1}$
$-9.999999964673912 \cdot 10^{-7}$	$9.999999999970999 \cdot 10^{-7}$	$9.999999999970999 \cdot 10^{-7}$
$-5.000006338012225 \cdot 10^{-11}$	$4.999999999985000 \cdot 10^{-11}$	$4.999999999985000 \cdot 10^{-11}$
$-9.999999963825105 \cdot 10^{-21}$	$9.999999999969999 \cdot 10^{-21}$	$9.999999999969999 \cdot 10^{-21}$

We will use the following notation:

MATRIX	exact eigenvalue	computed eigenvalue
A	λ	$\widetilde{\lambda}$
A_i	μ	$-$
$\widetilde{A}_i = fl(A_i)$	$\widehat{\mu}$	$\widetilde{\mu} = fl(\widehat{\mu})$
A_i^{-1}	ν	$-$
$\widetilde{(A_i^{-1})} = fl(A_i^{-1})$	$\widehat{\nu}$	$\widetilde{\nu} = fl(\widehat{\nu})$

Let $\widetilde{A}_i = fl\left(A_i\right)$

$$\widetilde{A}_i = \begin{bmatrix} D_1\left(I+E_1\right) & 0 & 0 & z_1 \\ 0 & 0 & 0 & \zeta_i \\ 0 & 0 & D_2\left(I+E_2\right) & z_2 \\ z_1^T & \zeta_i & z_2^T & a\left(1+\varepsilon_a\right) \end{bmatrix}$$

where E_1,E_2 are diagonal matrices: $|(E_1)_{ii}| \leq \varepsilon_M, |(E_2)_{ii}| \leq \varepsilon_M$ and $|\varepsilon_a| \leq \varepsilon_M$.
Also,

$$\widetilde{\left(A_i^{-1}\right)} = fl\left(A_i^{-1}\right).$$

Let $\widetilde{A}_i = fl\left(A_i\right)$

$$\widetilde{A}_i = \begin{bmatrix} D_1\left(I+E_1\right) & 0 & 0 & z_1 \\ 0 & 0 & 0 & \zeta_i \\ 0 & 0 & D_2\left(I+E_2\right) & z_2 \\ z_1^T & \zeta_i & z_2^T & a\left(1+\varepsilon_a\right) \end{bmatrix}$$

where E_1,E_2 are diagonal matrices: $|(E_1)_{ii}| \leq \varepsilon_M, |(E_2)_{ii}| \leq \varepsilon_M$ and $|\varepsilon_a| \leq \varepsilon_M$.
Also,

$$\widetilde{\left(A_i^{-1}\right)} = fl\left(A_i^{-1}\right).$$

All elements of A_i^{-1} are computed with high relative accuracy except possibly b. Weather b is computed accurately (or we need "extra" precision) is monitored by condition number.

- Q: What is the accuracy of computed A_i^{-1}?

Let's recall

$$A_i^{-1} = \begin{bmatrix} D_1^{-1} & w_1 & 0 & 0 \\ w_1^T & b & w_2^T & 1/\zeta_i \\ 0 & w_2 & D_2^{-1} & 0 \\ 0 & 1/\zeta_i & 0 & 0 \end{bmatrix},$$

- Q: What is the accuracy of computed A_i^{-1}?

Let's recall

$$A_i^{-1} = \begin{bmatrix} D_1^{-1} & w_1 & 0 & 0 \\ w_1^T & b & w_2^T & 1/\zeta_i \\ 0 & w_2 & D_2^{-1} & 0 \\ 0 & 1/\zeta_i & 0 & 0 \end{bmatrix},$$

where

$$\begin{aligned} w_1 &= -D_1^{-1} z_1 \frac{1}{\zeta_i}, \ fl\left((w_1)_k\right) = -\frac{\zeta_k}{(d_k - d_i)(1+\varepsilon_1)\zeta_i}(1+\varepsilon_2+\varepsilon_3), \\ w_2 &= -D_2^{-1} z_2 \frac{1}{\zeta_i}, \\ b &= \frac{1}{\zeta_i^2}\left(-a + z_1^T D_1^{-1} z_1 + z_2^T D_2^{-1} z_2\right). \end{aligned}$$

- Q: What is the accuracy of computed A_i^{-1}?

Let's recall

$$A_i^{-1} = \begin{bmatrix} D_1^{-1} & w_1 & 0 & 0 \\ w_1^T & b & w_2^T & 1/\zeta_i \\ 0 & w_2 & D_2^{-1} & 0 \\ 0 & 1/\zeta_i & 0 & 0 \end{bmatrix},$$

where

$$\begin{aligned} w_1 &= -D_1^{-1} z_1 \frac{1}{\zeta_i}, \quad fl\left((w_1)_k\right) = -\frac{\zeta_k}{(d_k - d_i)(1+\varepsilon_1)\zeta_i}(1+\varepsilon_2+\varepsilon_3), \\ w_2 &= -D_2^{-1} z_2 \frac{1}{\zeta_i}, \\ b &= \frac{1}{\zeta_i^2}\left(-a + z_1^T D_1^{-1} z_1 + z_2^T D_2^{-1} z_2\right). \end{aligned}$$

$$A : \widetilde{\left(A_i^{-1}\right)}_{ij} = fl\left(A_i^{-1}\right)_{ij} = \left(A_i^{-1}\right)_{ij}(1+\varepsilon_{ij}), \quad |\varepsilon_{ij}| \leq 3\varepsilon_M$$

for all elements of matrix A_i^{-1} with possible exception of element b.

Q: When b is not computed accurately and how to fix it?

Q: When b is not computed accurately and how to fix it?

Condition K_1

$$K_1\left(\lambda_i\right) = \frac{|a| + \left|z_1^T D_1^{-1} z_1\right| + \left|z_2^T D_2^{-1} z_2\right|}{\left\|A_i^{-1}\right\|_2 \cdot \zeta_i^2}.$$

Q: When b is not computed accurately and how to fix it?

Condition K_1

$$K_1(\lambda_i) = \frac{|a| + \left|z_1^T D_1^{-1} z_1\right| + \left|z_2^T D_2^{-1} z_2\right|}{\left\|A_i^{-1}\right\|_2 \cdot \zeta_i^2}.$$

If

$$K_1 >> 1$$

we have to compute b in double of standard precision arithmetic.

Q: When b is not computed accurately and how to fix it?

Condition K_1

$$K_1\left(\lambda_i\right) = \frac{|a| + \left|z_1^T D_1^{-1} z_1\right| + \left|z_2^T D_2^{-1} z_2\right|}{\left\|A_i^{-1}\right\|_2 \cdot \zeta_i^2}.$$

If

$$K_1 >> 1$$

we have to compute b in double of standard precision arithmetic.

Also

$$K_1 \leq (n-2)\,\frac{1}{\zeta_i} \max_{\substack{k=1,\ldots,n-1 \\ k\neq i}} \left|\zeta_k\right|.$$

Example (double precision) 1/3

Let

$$A = \begin{bmatrix} 10^{10} & 0 & 0 & 0 & 0 & 10^{10} \\ 0 & 4 & 0 & 0 & 0 & 1 \\ 0 & 0 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 10^{10} & 1 & 1 & 1 & 1 & 10^{10} \end{bmatrix}.$$

Condition numbers of matrix A eigenvalues are:

K_1
$3.999999999339841 \cdot 10^{-20}$
$3.007936375312118 \cdot 10^{09}$
$3.229973232799648 \cdot 10^{09}$
$3.760911826539061 \cdot 10^{09}$
$4.321871634242305 \cdot 10^{09}$
$4.321871634242305 \cdot 10^{09}$

Example (double precision) 2/3

$$A_2 = \begin{bmatrix} 10^{10}-4 & 0 & 0 & 0 & 0 & 10^{10} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -2 & 0 & 1 \\ 0 & 0 & 0 & 0 & -3 & 1 \\ 10^{10} & 1 & 1 & 1 & 1 & 10^{10}-4 \end{bmatrix}.$$

Inverse computed by *aheig* is

$$A_2^{-1} = \begin{bmatrix} 1.0000000004 \cdot 10^{-10} & -1.0000000004 & 0 & 0 & 0 & 0 \\ -1.0000000004 & 6.16666666667 & 1 & 5 \cdot 10^{-1} & 3.3333 \cdot 10^{-1} & 1 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 5 \cdot 10^{-1} & 0 & -5 \cdot 10^{-1} & 0 & 0 \\ 0 & -3.3333 \cdot 10^{-1} & 0 & 0 & -3.3333 \cdot 10^{-1} & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

$b = 6.166666666666667$, by *aheig* ,
$b = 6.166665889418350$, by Matlab *inv*,
$b = 6.166666668266667$, by *aheig_quad*.

Example (double precision) 2/3

$$A_2 = \begin{bmatrix} 10^{10}-4 & 0 & 0 & 0 & 0 & 10^{10} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -2 & 0 & 1 \\ 0 & 0 & 0 & 0 & -3 & 1 \\ 10^{10} & 1 & 1 & 1 & 1 & 10^{10}-4 \end{bmatrix}.$$

Inverse computed by *aheig* is

$$A_2^{-1} = \begin{bmatrix} 1.0000000004 \cdot 10^{-10} & -1.0000000004 & 0 & 0 & 0 & 0 \\ -1.0000000004 & 6.16666666667 & 1 & 5 \cdot 10^{-1} & 3.3333 \cdot 10^{-1} & 1 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 5 \cdot 10^{-1} & 0 & -5 \cdot 10^{-1} & 0 & 0 \\ 0 & -3.3333 \cdot 10^{-1} & 0 & 0 & -3.3333 \cdot 10^{-1} & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

$b = 6.166666666666667$, by *aheig* ,
$b = 6.166665889418350$, by Matlab *inv*,
$b = 6.166666668266667$, by *aheig_quad*.

Example (double precision) 3/3

Eigenvalues computed by *aheig*, *aheig_quad* and Mathematica (100 digits) are:

λ_{aheig}	λ_{aheig_quad}	λ_{Math}
$2.000000000000000 \cdot 10^{10}$	$2.000000000000000 \cdot 10^{10}$	$2.000000000000000 \cdot 10^{10}$
4.150396802313551	4.150396802279712	4.150396802279713
3.161498641452035	3.161498641430967	3.161498641430967
2.188045596352105	2.188045596339914	2.188045596339914
1.216093560005649	1.216093584948579	1.216093584948579
$-7.160348702977373 \cdot 10^{-1}$	$-7.160346250991725 \cdot 10^{-1}$	$-7.160346250991725 \cdot 10^{-1}$

Example (double precision) 3/3

Eigenvalues computed by *aheig*, *aheig_quad* and Mathematica (100 digits) are:

λ_{aheig}	λ_{aheig_quad}	λ_{Math}
$2.000000000000000 \cdot 10^{10}$	$2.000000000000000 \cdot 10^{10}$	$2.000000000000000 \cdot 10^{10}$
4.150396802313551	4.150396802279712	4.150396802279713
3.161498641452035	3.161498641430967	3.161498641430967
2.188045596352105	2.188045596339914	2.188045596339914
1.216093560005649	1.216093584948579	1.216093584948579
$-7.160348702977373 \cdot 10^{-1}$	$-7.160346250991725 \cdot 10^{-1}$	$-7.160346250991725 \cdot 10^{-1}$

Example (double precision) 3/3

Eigenvalues computed by *aheig*, *aheig_quad* and Mathematica (100 digits) are:

λ_{aheig}	λ_{aheig_quad}	λ_{Math}
$2.000000000000000 \cdot 10^{10}$	$2.000000000000000 \cdot 10^{10}$	$2.000000000000000 \cdot 10^{10}$
4.150396802313551	4.150396802279712	4.150396802279713
3.161498641452035	3.161498641430967	3.161498641430967
2.188045596352105	2.188045596339914	2.188045596339914
1.216093560005649	1.216093584948579	1.216093584948579
$-7.160348702977373 \cdot 10^{-1}$	$-7.160346250991725 \cdot 10^{-1}$	$-7.160346250991725 \cdot 10^{-1}$

Eigenvectors computed by *aheig_quad* are accurate and therefore, orthogonal. For example, let us look at U_6

$U_{6(eig)}$	$U_{6(aheig\_quad)}$	$U_{6(Math)}$
$6.210673512155598 \cdot 10^{-1}$	$6.210673512038695 \cdot 10^{-1}$	$6.210673512038695 \cdot 10^{-1}$
$1.316927038563995 \cdot 10^{-1}$	$1.316927038539883 \cdot 10^{-1}$	$1.316927038539883 \cdot 10^{-1}$
$1.671317449694245 \cdot 10^{-1}$	$1.671317449663874 \cdot 10^{-1}$	$1.671317449663874 \cdot 10^{-1}$
$2.286669490587952 \cdot 10^{-1}$	$2.286669490546950 \cdot 10^{-1}$	$2.286669490546950 \cdot 10^{-1}$
$3.619200580729778 \cdot 10^{-1}$	$3.619200581179692 \cdot 10^{-1}$	$3.619200581179692 \cdot 10^{-1}$
$-6.210673512600302 \cdot 10^{-1}$	$-6.210673512483401 \cdot 10^{-1}$	$-6.210673512483400 \cdot 10^{-1}$

Example (double precision) 3/3

Eigenvalues computed by *aheig*, *aheig_quad* and Mathematica (100 digits) are:

λ_{aheig}	λ_{aheig_quad}	λ_{Math}
$2.000000000000000 \cdot 10^{10}$	$2.000000000000000 \cdot 10^{10}$	$2.000000000000000 \cdot 10^{10}$
4.150396802313551	4.150396802279712	4.150396802279713
3.161498641452035	3.161498641430967	3.161498641430967
2.188045596352105	2.188045596339914	2.188045596339914
1.216093560005649	1.216093584948579	1.216093584948579
$-7.160348702977373 \cdot 10^{-1}$	$-7.160346250991725 \cdot 10^{-1}$	$-7.160346250991725 \cdot 10^{-1}$

Eigenvectors computed by *aheig_quad* are accurate and therefore, orthogonal. For example, let us look at U_6

$U_{6(eig)}$	$U_{6(aheig\_quad)}$	$U_{6(Math)}$
$6.210673512155598 \cdot 10^{-1}$	$6.210673512038695 \cdot 10^{-1}$	$6.210673512038695 \cdot 10^{-1}$
$1.316927038563995 \cdot 10^{-1}$	$1.316927038539883 \cdot 10^{-1}$	$1.316927038539883 \cdot 10^{-1}$
$1.671317449694245 \cdot 10^{-1}$	$1.671317449663874 \cdot 10^{-1}$	$1.671317449663874 \cdot 10^{-1}$
$2.286669490587952 \cdot 10^{-1}$	$2.286669490546950 \cdot 10^{-1}$	$2.286669490546950 \cdot 10^{-1}$
$3.619200580729778 \cdot 10^{-1}$	$3.619200581179692 \cdot 10^{-1}$	$3.619200581179692 \cdot 10^{-1}$
$-6.210673512600302 \cdot 10^{-1}$	$-6.210673512483401 \cdot 10^{-1}$	$-6.210673512483400 \cdot 10^{-1}$

Q: Is μ_i eigenvalue of A_i closest to zero and if not how far is it from the closest one?

Condition K_2

$$K_2\left(\lambda_i\right) = \left\|A_i^{-1}\right\|_2 . \left|\mu_i\right| .$$

A:

$$\text{If } K_2 >> 1$$

$$\Downarrow$$

μ_i is not eigenvalue of matrix A_i which is closest to zero.

Q: Is μ_i eigenvalue of A_i closest to zero and if not how far is it from the closest one?

Condition K_2

$$K_2(\lambda_i) = \left\|A_i^{-1}\right\|_2 \cdot |\mu_i| \, .$$

A:

If $K_2 >> 1$

$$\Downarrow$$

μ_i is not eigenvalue of matrix A_i which is closest to zero.

Two different cases:

- for λ_1 or λ_n, we can compute them from the starting matrix A.
- for "inside" eigenvalues (only open problem), possible solutions is to send this eigenvalue to the "other" pole.

Example "Arrowhead matrix applications in quantum optics" 1/3

The research is about quantum dots excited states decay in real photonic crystals. Matrix has the following structure

$$A = \begin{bmatrix} \Delta & g_1 & g_2 & \cdots & g_n \\ g_1 & \omega_1 & & & \\ g_2 & & \omega_2 & & \\ \vdots & & & \ddots & \\ g_n & & & & \omega_n \end{bmatrix},$$

where

- $\Delta-$ quantum dot transition frequency,
- ω_i- is frequency of optical mode,
- g_i- interaction constant of quantum dot with optical modes.

At this point our task is to compute the eigenvalues of matrix A.

Example "Arrowhead matrix applications in quantum optics" 2/3

The size of the matrix is changeable but in realistic case it is approximately $n \approx 10^3$ to 10^4. For example for $n = 2501$.

- g is vector with components from interval

$$[3.2087698694339995 \cdot 10^{06}, 4.9584253488898976 \cdot 10^{06}].$$

- ω is vector with components from interval

$$[5.8769928900036225 \cdot 10^{14}, 1.3709849013800450 \cdot 10^{15}].$$

- $\Delta = 9.7949881500060375 \cdot 10^{14}$.

Example "Arrowhead matrix applications in quantum optics" 3/3

The components of vector g are of the same order of magnitude $\Longrightarrow$ we can guarantee all eigenvalues will be computed with high relative accuracy. $(K_1 \in \left[1.613222657427353 \cdot 10^{-4}, 1.000000000735696\right])$.

Let

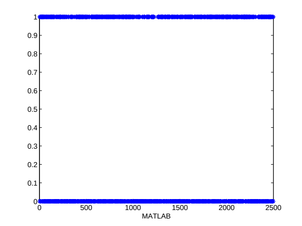
$$y(\lambda) = \begin{cases} 0, & d_i > \lambda_{i+1} > d_{i+1} \\ 1, & \lambda_{i+1} > d_i \text{ or } \lambda_{i+1} < d_{i+1} \end{cases}$$

Example "Arrowhead matrix applications in quantum optics" 3/3

The components of vector g are of the same order of magnitude $\Longrightarrow$ we can guarantee all eigenvalues will be computed with high relative accuracy. $(K_1 \in \left[1.613222657427353 \cdot 10^{-4}, 1.000000000735696\right])$.

Let

$$y(\lambda) = \begin{cases} 0, & d_i > \lambda_{i+1} > d_{i+1} \\ 1, & \lambda_{i+1} > d_i \text{ or } \lambda_{i+1} < d_{i+1} \end{cases}$$

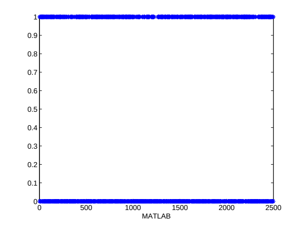


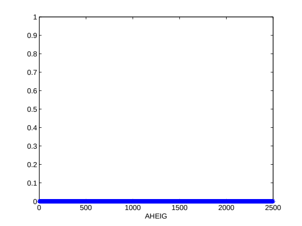
Example "Arrowhead matrix applications in quantum optics" 3/3

The components of vector g are of the same order of magnitude $\Longrightarrow$ we can guarantee all eigenvalues will be computed with high relative accuracy. $(K_1 \in \left[1.613222657427353 \cdot 10^{-4}, 1.000000000735696\right])$.

Let

$$y(\lambda) = \begin{cases} 0, & d_i > \lambda_{i+1} > d_{i+1} \\ 1, & \lambda_{i+1} > d_i \text{ or } \lambda_{i+1} < d_{i+1} \end{cases}$$





Algorithm *herm2ahig*

Let

$$H = \begin{bmatrix} D & r \\ r^T & \alpha \end{bmatrix}, \quad r = \begin{bmatrix} \rho_1 & \rho_2 & \cdots & \rho_{n-1} \end{bmatrix}^T, \rho_i \in \mathbb{C}$$

hermitian arrowhead matrix transform

$$A = \Delta^* H \Delta = \begin{bmatrix} D & z \\ z^T & \alpha \end{bmatrix}.$$

where

$$\Delta = \mathrm{diag}(\frac{\rho_1}{|\rho_1|}, \ldots, \frac{\rho_{n-1}}{|\rho_{n-1}|}, 1).$$

Algorithm *herm2ahig*

Let

$$H=\begin{bmatrix} D & r \\ r^T & \alpha \end{bmatrix}, \quad r=\begin{bmatrix} \rho_1 & \rho_2 & \cdots & \rho_{n-1} \end{bmatrix}^T, \rho_i \in \mathbb{C}$$

hermitian arrowhead matrix transform

$$A=\Delta^* H \Delta=\begin{bmatrix} D & z \\ z^T & \alpha \end{bmatrix}.$$

where

$$\Delta=\operatorname{diag}(\frac{\rho_1}{|\rho_1|},\ldots,\frac{\rho_{n-1}}{|\rho_{n-1}|},1).$$

$$A=V\Lambda V^T \Longrightarrow H=U\Lambda U^*, U=\Delta V.$$

Algorithm *herm2ahig*

Let

$$H = \begin{bmatrix} D & r \\ r^T & \alpha \end{bmatrix}, \quad r = \begin{bmatrix} \rho_1 & \rho_2 & \cdots & \rho_{n-1} \end{bmatrix}^T, \rho_i \in \mathbb{C}$$

hermitian arrowhead matrix transform

$$A = \Delta^* H \Delta = \begin{bmatrix} D & z \\ z^T & \alpha \end{bmatrix}.$$

where

$$\Delta = \operatorname{diag}\left(\frac{\rho_1}{|\rho_1|}, \ldots, \frac{\rho_{n-1}}{|\rho_{n-1}|}, 1\right).$$

$$A = V\Lambda V^T \Longrightarrow H = U\Lambda U^*, U = \Delta V.$$

Accuracy of EVD of $A \Longrightarrow$ Accuracy of EVD of H.

(If *aheig_quad* is needed, we also need to compute z in double of standard precision.)

Algorithm *dc_t2a*

T is symmetric tridiagonal matrix

$$T = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{bmatrix} \equiv \begin{bmatrix} T_1 & \beta_{k+1}e_k & 0 \\ \beta_{k+1}e_k^T & \alpha_{k+1} & \beta_{k+2}e_1^T \\ 0 & \beta_{k+2}e_1 & T_2 \end{bmatrix}.$$

where $1 < k < n$, T_1 and T_2 are $k \times k$ and $(n-k-1) \times (n-k-1)$ submatrices of T, respectively, and e_j is the $j-th$ unit vector of appropriate dimension. Usually k is taken to be $[n/2]$.

Algorithm *dc_t2a*

Let $Q_i D_i Q_i^T = T_i$ be an eigenvalue decomposition of T_i.

$$
\begin{aligned}
T &= \begin{bmatrix} T_1 & \beta_{k+1} e_k & 0 \\ \beta_{k+1} e_k^T & \alpha_{k+1} & \beta_{k+2} e_1^T \\ 0 & \beta_{k+2} e_1 & T_2 \end{bmatrix} \\
&= \begin{bmatrix} Q_1 D_1 Q_1^T & \beta_{k+1} e_k & 0 \\ \beta_{k+1} e_k^T & \alpha_{k+1} & \beta_{k+2} e_1^T \\ 0 & \beta_{k+2} e_1 & Q_2 D_2 Q_2^T \end{bmatrix} \\
&= \begin{bmatrix} 0 & Q_1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & Q_2 \end{bmatrix} \begin{bmatrix} \alpha_{k+1} & \beta_{k+1} l_1^T & \beta_{k+2} f_2^T \\ \beta_{k+1} l_1 & D_1 & 0 \\ \beta_{k+2} f_2 & 0 & D_2 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ Q_1^T & 0 & 0 \\ 0 & 0 & Q_2^T \end{bmatrix} \\
&= QAQ^T,
\end{aligned}
$$

where l_1^T is last row of Q_1 and f_2^T is first row of Q_2. Thus T is reduced to symmetric arrowhead matrix A by orthogonal transformation Q.

Algorithm *dc_t2a*

is used for computing an eigenvalue decomposition of symmetric tridiagonal matrices in a way that:

- We transform a symmetric tridiagonal matrix to symmetric arrowhead matrix by orthogonal transformation.
- We compute eigenvalue decomposition of symmetric arrowhead matrix using *aheig* algorithm.
- We can guarantee high relative accuracy of eigenvalues and orthogonality of eigenvectors of tridiagonal symmetric matrix only when we can guarantee high relative accuracy of eigenvalue decompositions of corresponding symmetric arrowhead matrices emerging during algorithm *dc_t2a*.

Algorithm *dc_t2a*

Example "Wilkinson matrix 21" 1/1

λ_{Math}	$\lambda_{dc\_t2a(T)}$
10.74619418290339	10.74619418290340
10.74619418290332	10.74619418290332
9.210678647361332	9.210678647361331
9.210678647304919	9.210678647304917
8.038941122829023	8.038941122829025
8.038941115814273	8.038941115814275
7.003952209528675	7.003952209528674
7.003951798616375	7.003951798616374
6.000234031584167	6.000234031584169
6.000217522257098	6.000217522257100
5.000244425001913	5.000244425001914
4.999782477742902	4.999782477742903
4.004354023440857	4.004354023440857
3.996048201383625	3.996048201383625
3.043099292578824	3.043099292578824
2.961058884185727	2.961058884185727
2.130209219362506	2.130209219362505
1.789321352695081	1.789321352695081
0.947534367529293	0.947534367529293
0.253805817096678	0.253805817096678
−1.125441522119984	−1.125441522119985

Algorithm *dc_t2a*

Example "Wilkinson matrix 21" 1/1

λ_{Math}	$\lambda_{dc\_t2a(T)}$
10.74619418290339	10.74619418290340
10.74619418290332	10.74619418290332
9.210678647361332	9.210678647361331
9.210678647304919	9.210678647304917
8.038941122829023	8.038941122829025
8.038941115814273	8.038941115814275
7.003952209528675	7.003952209528674
7.003951798616375	7.003951798616374
6.000234031584167	6.000234031584169
6.000217522257098	6.000217522257100
5.000244425001913	5.000244425001914
4.999782477742902	4.999782477742903
4.004354023440857	4.004354023440857
3.996048201383625	3.996048201383625
3.043099292578824	3.043099292578824
2.961058884185727	2.961058884185727
2.130209219362506	2.130209219362505
1.789321352695081	1.789321352695081
0.947534367529293	0.947534367529293
0.253805817096678	0.253805817096678
−1.125441522119984	−1.125441522119985

Algorithm *dpr1_2a*

Let

$$M = D + uu^T$$
$$= \begin{bmatrix} d_1 + u_1^2 & u_1 u_2 & \cdots & u_1 u_n \\ u_2 u_1 & d_2 + u_2^2 & & u_2 u_n \\ \vdots & & \ddots & \vdots \\ u_n u_1 & u_n u_2 & \cdots & d_n + u_n^2 \end{bmatrix}$$

where

$$d_i \in \mathbb{R}, \quad i = 1, \ldots, n,$$
$$u_i \in \mathbb{R},, \quad i = 1, \ldots, n.$$

Now, for

$$x_1 = 0,\ x_j = u_j / u_1,\ j = 2, \ldots, n$$

$$G = \left(I + e_1 x^T\right) M \left(I - e_1 x^T\right)$$

is arrowhead matrix.

Algorithm *dpr1_2a*

Under assumptions

$$d_1 < d_j, \; j = 2, \ldots, n$$

we form

$$\Delta = \operatorname{diag}\left(1, \frac{\sqrt{d_2 - d_1}}{u_1}, \cdots, \frac{\sqrt{d_n - d_1}}{u_1}\right).$$

$$A = \Delta G \Delta^{-1}$$

is symmetric arrowhead matrix of form

$$A = \begin{bmatrix} d_1 + u_1^2 + \cdots + u_n^2 & u_2\sqrt{d_2 - d_1} & \cdots & u_n\sqrt{d_n - d_1} \\ u_2\sqrt{d_2 - d_1} & d_2 & & 0 \\ \vdots & & \ddots & \vdots \\ u_n\sqrt{d_n - d_1} & 0 & \cdots & d_n \end{bmatrix}.$$

Algorithm *aheig* is now used on A. If *aheig_quad* is needed, we also need to compute α and z in double of standard precision.