Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 438320, 12 pages
http://dx.doi.org/10.1155/2013/438320

Research Article

Real Fast Structure-Preserving Algorithm for Eigenproblem of Complex Hermitian Matrices

Jiangzhou Lai[1] and Linzhang Lu[2,3]

[1] *School of Mathematics and Computer Science, Fuzhou University, Fuzhou 351008, China*
[2] *School of Mathematics and Computer Science, Guizhou Normal University, Guiyang, Guizhou 550001, China*
[3] *School of Mathematical Sciences, Xiamen University, Xiamen 361005, China*

Correspondence should be addressed to Jiangzhou Lai; jzlai868@fzu.edu.cn

Received 1 January 2013; Accepted 18 February 2013

Academic Editor: Piermarco Cannarsa

It is well known that the flops for complex operations are usually 4 times of real cases. In the paper, using real operations instead of complex, a real fast structure-preserving algorithm for eigenproblem of complex Hermitian matrices is given. We make use of the real symmetric and skew-Hamiltonian structure transformed by Wilkinson's way, focus on symplectic orthogonal similarity transformations and their structure-preserving property, and then reduce it into a two-by-two block tridiagonal symmetric matrix. Finally a real algorithm can be quickly obtained for eigenvalue problems of the original Hermitian matrix. Numerical experiments show that the fast algorithm can solve real complex Hermitian matrix efficiently, stably, and with high precision.

1. Introduction

We consider eigenvalue problems of the complex Hermitian matrix of the form

$$Hx = \lambda x, \quad H \in \mathbb{C}^{n\times n}, \ x \in \mathbb{C}^n, \tag{1}$$

where λ is a real scalar and $H \in \mathbb{C}^{n\times n}$ is a complex Hermitian matrix of the form

$$H = A + iB, \tag{2}$$

where A is a real symmetric matrix; that is, $A^T = A \in \mathbb{R}^n$ and B is a real skew-symmetric nonzero matrix, that is, $B = -B^T \neq 0 \in \mathbb{R}^n$, given by

$$A = \frac{H^T + H}{2}, \qquad B = \frac{H - H^T}{2i}, \tag{3}$$

and $x \in \mathbb{C}^n$ can be written as $x = u + iv$, where $u, v \in \mathbb{R}^n$.

We know that the property of the complex Hermitian matrix is also possessed by the symmetry matrix; therefore little work has been done on complex Hermitian matrix. However, from the view of numerical computation, the biggest difference between these two types of matrices is a complex Hermitian matrix that involves complex operations, while the latter does not. It is well known that a complex Hermitian matrix is unitary similar to a real diagonal matrix; that is, it has real eigenvalue but complex eigenvector. Because the flops for the complex operation are usually 4 times of the real case, so how to reduce the computation cost of the eigenvalue problem of complex Hermitian matrices is very meaningful and worthy of research.

In 1965, the famous numerical analysis expert Wilkinson [1] gave a way to transform a complex Hermitian matrix into real case. To be more precise, if $(\lambda, u + iv)$ is the eigenpair of H, then

$$Hx = (A + iB)(u + iv) = \lambda (u + iv). \tag{4}$$

Since the eigenvalues of H are real, the previous equation can be written as

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \end{bmatrix}. \tag{5}$$

So the eigenproblem (1) is transformed into the eigenproblem of the real symmetry matrix

$$S = \begin{bmatrix} A & -B \\ B & A \end{bmatrix} \in \mathbb{R}^{2n\times 2n}. \tag{6}$$

Note that the matrix S in (6) is not only real symmetry but also skew-Hamiltonian. Therefore, in the following discussion, inspired by the algorithms for the eigenproblem of Hamiltonian (skew-Hamiltonian) matrix [2–11], we devise an algorithm by taking full advantage of the special structure of (6).

In this paper we first prove that orthogonal symplectic similarity transformations not only preserve the symmetry and skew-Hamiltonian structure of (6), but also eigenvectors of (6), since the eigenvectors of (6) have special structure; that is, it is also orthogonal symplectic. Therefore, by these transformations, a real stable, accurate, and fast method is devised to reduce (6) into a block symmetry tridiagonal matrix, and then we get the eigenproblem of the original Hermitian matrix. When implementing the method, although its dimension doubles the one of H, we can make full use of the special structure of S to avoid superfluous calculation and storage.

The remainder of the paper is organized as follows. In Section 2, we show the basic orthogonal symplectic matrix and its algorithm. In Section 3, we present the overall real algorithm for eigenproblem of complex Hermitian matrix. Some implementation aspects are discussed in Section 4. Finally, in Section 5, the results of numerical experiments are discussed to end the paper.

2. Basic Orthogonal Symplectic Matrix and Its Algorithm

It is well known that by choosing a proper unitary vector w, Householder transformation can zero some entries of a given vector; likewise, we defined Householder symplectic transformation which has the following form.

Definition 1. A real $2n \times 2n$ matrix S is a symplectic matrix if $SJS^T = J$, where S^T is the transpose of S and J is the orthogonal matrix:

$$J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix} \in \mathbb{R}^{2n\times 2n}. \tag{7}$$

Here I_n is the identity $n \times n$ matrix and 0 is the zero $n \times n$ matrix.

Definition 2. Letting $k \in \{1, 2, \ldots, n\}$, a matrix $H(k, w) \in \mathbb{R}^{2n\times 2n}$ is called Householder symplectic matrix, if it has the form

$$H(k, w) = \begin{bmatrix} \operatorname{diag}(I_{k-1}, P) & 0 \\ 0 & \operatorname{diag}(I_{k-1}, P) \end{bmatrix} \in \mathbb{R}^{2n\times 2n}, \tag{8}$$

where

$$P = I - \frac{2ww^T}{w^T w}, \quad w \in \mathbb{R}^{n-k+1}. \tag{9}$$

Obviously, $H = \operatorname{diag}(I_{k-1}, P)$ is Householder transformation of order n, when $w = 0$, $H(k, w) = I_{2n}$.

Note that Householder symplectic matrix $H(k, w)$ is just a direct sum of two "ordinary" n-by-n Householder matrices, specially when $w = 0$, $H(k, w) = I_{2n}$. So based on the algorithm of "ordinary" Householder transformation, we obtain the following algorithm for Householder symplectic transformation $H(k, w)$ of order $2n$.

Algorithm 3 (Householder symplectic transformation [8]). Given k ($1 \le k < n$) and $y, z \in \mathbb{R}^n$, the following algorithm determines $w = (w_k, \ldots, w_n)^T$, such that if

$$H(k, w)\begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} v \\ x \end{pmatrix}, \tag{10}$$

then $x_i = 0$, $\quad i = k + 1, \ldots, n$:

$$\begin{aligned} \sigma &:= \left(z_k^2 + z_{k+1}^2 + \cdots + z_n^2\right)^{1/2}, \\ w_k &:= z_k + \operatorname{sign}(z_k)\,\sigma, \\ w_i &:= z_i \end{aligned} \tag{11}$$

end.

$H(k, w)$ is completely determined by kth to nth components of vector z; it makes x satisfy $x_i = 0$ for $i = k + 1, \ldots, n$. Note that by interchanging the roles of y and z, the previous algorithm can be used to determine $H(k, w)$ such that $v_i = 0$ for $i = k + 1, \ldots, n$.

Householder symplectic transformation can be used to zero large portions of a vector as Householder transformation; likewise, we can define Givens symplectic transformation, which can be used to zero single entries.

Definition 4. Given $k \in \{1, 2, \ldots, n\}$, a matrix $J(k, \theta) \in \mathbb{R}^{2n\times 2n}$ is called Givens symplectic matrix, if it has the form

$$J(k, \theta) = \begin{bmatrix} C & S \\ -S & C \end{bmatrix} = \begin{bmatrix} I_{k-1} & & & 0_{k-1} & & \\ & \cos\theta & & & \sin\theta & \\ & & I_{n-k} & & & 0_{n-k} \\ 0_{k-1} & & & I_{k-1} & & \\ & -\sin\theta & & & \cos\theta & \\ & & 0_{n-k} & & & I_{n-k} \end{bmatrix}, \tag{12}$$

where

$$\begin{aligned} C &= \operatorname{diag}(I_{k-1}, \cos\theta, I_{n-k}), \\ S &= \operatorname{diag}(0_{k-1}, \sin\theta, 0_{n-k}). \end{aligned} \tag{13}$$

$J(k, \theta)$ is an "ordinary" $2n$-by-$2n$ Givens rotation that rotates in planes k and $n + k$. The corresponding algorithm is as follows.

Algorithm 5 (Givens symplectic transformation [8]). Given $k\,(1 \le k < n)$ and $y, z \in \mathbb{R}^n$, the following algorithm determines $c = \cos\theta$, $s = \sin\theta$ such that if

$$J(k, w)\begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} v \\ x \end{pmatrix}, \tag{14}$$

then $x_k = 0$:

$$\begin{aligned}
&\sigma := \left(y_k^2 + z_k^2\right)^{1/2}, \\
&\text{if } \sigma = 0 \\
&\qquad \text{then } c := 1 \text{ and } s := 0 \\
&\qquad \text{else } c := y_k/\sigma \text{ and } s := z_k/\sigma \\
&\text{end.}
\end{aligned} \tag{15}$$

3. The Reduction of the Symmetry and Skew-Hamiltonian Matrix S

Now we consider some properties of the symmetry and skew-Hamiltonian matrix S of (6).

Lemma 6. *If* $(\lambda, [\begin{smallmatrix} u \\ v \end{smallmatrix}])$ *is the eigenpair of*

$$S = \begin{bmatrix} A & -B \\ B & A \end{bmatrix}, \tag{16}$$

then $(\lambda, [\begin{smallmatrix} -v \\ u \end{smallmatrix}])$ *is also its eigenpair.*

Proof. If

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \end{bmatrix}, \tag{17}$$

then we have

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix}\begin{bmatrix} -v \\ u \end{bmatrix} = \lambda \begin{bmatrix} -v \\ u \end{bmatrix}. \tag{18}$$

□

According to Lemma 6, there are two different eigenvectors corresponding to the same eigenvalue, so it can be easy to prove that

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix}\begin{bmatrix} U & -V \\ V & U \end{bmatrix} = \Lambda \begin{bmatrix} U & -V \\ V & U \end{bmatrix}, \tag{19}$$

where $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ is the set of eigenvalues of (6) and

$$\begin{bmatrix} U & -V \\ V & U \end{bmatrix} \in \mathbb{R}^{2n\times 2n} \tag{20}$$

is the eigenvectors of (6). Note that the matrix in (20) is symplectic orthogonal. The following lemma tells us that the product of symplectic orthogonal matrices is also symplectic orthogonal.

Lemma 7. *Supposing that 2n-by-2n real matrices of the form*

$$Q = \begin{bmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{bmatrix}, \qquad \widetilde{Q} = \begin{bmatrix} \widetilde{Q}_1 & \widetilde{Q}_2 \\ -\widetilde{Q}_2 & \widetilde{Q}_1 \end{bmatrix} \tag{21}$$

are both orthogonal and symplectic, then their product $Q\widetilde{Q}$ *is both orthogonal and symplectic.*

Lemma 7 shows that the special structure of eigenvectors of S can be preserved by the symplectic orthogonal similarity transformations. The following theorem is the foundation of the fast and stable structure-preserving algorithm given in the following discussion.

Theorem 8. *Suppose* $A^T = A \in R^{n\times n}, B^T = -B \neq 0 \in R^{n\times n}$, *and*

$$Q = \begin{bmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{bmatrix} \tag{22}$$

is an orthogonal symplectic matrix. If

$$\begin{bmatrix} \widetilde{A}_{11} & \widetilde{A}_{12} \\ \widetilde{A}_{21} & \widetilde{A}_{22} \end{bmatrix} \tag{23}$$

satisfy

$$\begin{bmatrix} \widetilde{A}_{11} & \widetilde{A}_{12} \\ \widetilde{A}_{21} & \widetilde{A}_{22} \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{bmatrix}^T \begin{bmatrix} A & -B \\ B & A \end{bmatrix}\begin{bmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{bmatrix}, \tag{24}$$

then

$$\begin{aligned}
\widetilde{A}_{11} &= \widetilde{A}_{22}, & \widetilde{A}_{21} &= -\widetilde{A}_{12}, \\
\widetilde{A}_{11}^T &= \widetilde{A}_{11}, & \widetilde{A}_{12}^T &= -\widetilde{A}_{12}.
\end{aligned} \tag{25}$$

Proof. It is easy to obtain

$$\begin{aligned}
\widetilde{A}_{11} &= Q_1^T A Q_1 + Q_1^T B Q_2 - Q_2^T B Q_1 + Q_2^T A Q_2, \\
\widetilde{A}_{22} &= Q_2^T A Q_2 - Q_2^T B Q_1 + Q_1^T A Q_1 + Q_1^T B Q_2, \\
\widetilde{A}_{12} &= Q_1^T A Q_2 - Q_1^T B Q_1 - Q_2 B Q_2 - Q_2^T A Q_1, \\
\widetilde{A}_{21} &= Q_2^T A Q_1 + Q_2^T B Q_2 + Q_1^T B Q_1 - Q_1^T A Q_2.
\end{aligned} \tag{26}$$

Then

$$\widetilde{A}_{11} = \widetilde{A}_{22}, \qquad \widetilde{A}_{21} = -\widetilde{A}_{12}. \tag{27}$$

Moreover $A^T = A,\ B^T = -B$, hence we have

$$\widetilde{A}_{11}^T = \widetilde{A}_{11}, \qquad \widetilde{A}_{12}^T = -\widetilde{A}_{12}. \tag{28}$$

□

The previous theorem shows that the symplectic similarity transformation preserves the symmetry and skew-Hamiltonian structure of

$$S = \begin{bmatrix} A & -B \\ B & A \end{bmatrix}, \quad \left(A^T = A,\ B^T = -B\right). \tag{29}$$

This structure-preserving property provides a way to transform the matrix into a special skew-Hamiltonian matrix:

$$\begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix} \in \mathbb{R}^{2n\times 2n}, \tag{30}$$

where $T \in \mathbb{R}^{n\times n}$ is a real symmetry tridiagonal matrix. So we have the following results, which provide a theoretical basis for the real fast structure-preserving algorithm for eigenproblems of a Hermitian matrix.

Theorem 9. *Letting $S = \left[\begin{smallmatrix} A & -B \\ B & A \end{smallmatrix}\right] \in \mathbb{R}^{2n\times 2n}$, where $A^T = A$, $B^T = -B$, then there exists an orthogonal symplectic matrix $Q = \left[\begin{smallmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{smallmatrix}\right] \in \mathbb{R}^{2n\times 2n}$ such that*

$$Q^T S Q = \begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix} \in \mathbb{R}^{2n\times 2n}, \tag{31}$$

where T is a symmetric tridiagonal matrix. Suppose that T has the eigenvalue decomposition

$$\widetilde{Q}_1{}^T T \widetilde{Q}_1 = \Lambda, \tag{32}$$

that is

$$\begin{bmatrix} \widetilde{Q}_1 & 0 \\ 0 & \widetilde{Q}_1 \end{bmatrix}^T \begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix} \begin{bmatrix} \widetilde{Q}_1 & 0 \\ 0 & \widetilde{Q}_1 \end{bmatrix} = \begin{bmatrix} \Lambda & 0 \\ 0 & \Lambda \end{bmatrix}. \tag{33}$$

Then eigenvalues of S are

$$\lambda(S) = \lambda(T) \sqcup \lambda(T), \tag{34}$$

and eigenvectors are the columns of

$$Q\widetilde{Q} = \begin{bmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{bmatrix} \begin{bmatrix} \widetilde{Q}_1 & 0 \\ 0 & \widetilde{Q}_1 \end{bmatrix} = \begin{bmatrix} Q_1\widetilde{Q}_1 & Q_2\widetilde{Q}_1 \\ -Q_2\widetilde{Q}_1 & Q_1\widetilde{Q}_1 \end{bmatrix}. \tag{35}$$

We now consider how an orthogonal symplectic $Q \in \mathbb{R}^{2n\times 2n}$ can be determined to reduce $S = \left[\begin{smallmatrix} A & -B \\ B & A \end{smallmatrix}\right]$ ($A^T = A$, $B^T = -B$) to

$$Q^T S Q = \begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix}, \tag{36}$$

where T is a symmetry tridiagonal matrix. It is worth illustrating a few steps of the algorithm before presenting it inn full generality. We depict an 8-by-8 symmetry and skew-Hamiltonian matrix ($n = 4$) as follows:

$$S = \begin{bmatrix} A & -B \\ B & A \end{bmatrix} = \left[\begin{array}{cccc|cccc} x & x & x & x & 0 & x & x & x \\ x & x & x & x & x & 0 & x & x \\ x & x & x & x & x & x & 0 & x \\ x & x & x & x & x & x & x & 0 \\ \hline 0 & x & x & x & x & x & x & x \\ x & 0 & x & x & x & x & x & x \\ x & x & 0 & x & x & x & x & x \\ x & x & x & 0 & x & x & x & x \end{array}\right] \quad (n = 4). \tag{37}$$

The zero diagonals in B follow from skew symmetry. It is easy to verify that the (2, 1) and (1, 2) block of this matrix can be zeroed by applying a sequence of Householder and Givens symplectic similarity transformations.

The first step is to zero b_{31} and b_{41} using a Householder symplectic $H_1 = H(2, w)$. This matrix can be determined by executing Algorithm 3 with $k = 2$, $y = Ae_1$, and $z = Be_1$. Note that after performing, only the rows and columns 2, 3, and 4 of A, B, and $-B$ are affected, the result being

$$S = \begin{bmatrix} A & -B \\ B & A \end{bmatrix} := H_1 S H_1^T = \left[\begin{array}{cccc|cccc} x & x & x & x & 0 & x & 0 & 0 \\ x & x & x & x & x & 0 & x & x \\ x & x & x & x & 0 & x & 0 & x \\ x & x & x & x & 0 & x & x & 0 \\ \hline 0 & x & 0 & 0 & x & x & x & x \\ x & 0 & x & x & x & x & x & x \\ 0 & x & 0 & x & x & x & x & x \\ 0 & x & x & 0 & x & x & x & x \end{array}\right] \quad (n = 4). \tag{38}$$

Then (1, 3) and (1, 4) positions of B are zeroed; meanwhile the same positions of $-B$ are zeroed because Householder symplectic similarity transformation preserves the structure of S.

The next step is to zero b_{21} using Givens similarity $J_1 = J(2, \theta)$, which can be achieved by applying Algorithm 5 with $k = 2$, $y = Ae_1$, and $z = Be_1$. And notice that only the second row and column of A, B and $-B$ be affected. This implies that

$$S = \begin{bmatrix} A & -B \\ B & A \end{bmatrix} := J_1 S J_1^T = \left[\begin{array}{cccc|cccc} x & x & x & x & 0 & 0 & 0 & 0 \\ x & x & x & x & 0 & 0 & x & x \\ x & x & x & x & 0 & x & 0 & x \\ x & x & x & x & 0 & x & x & 0 \\ \hline 0 & 0 & 0 & 0 & x & x & x & x \\ 0 & 0 & x & x & x & x & x & x \\ 0 & x & 0 & x & x & x & x & x \\ 0 & x & x & 0 & x & x & x & x \end{array}\right] \quad (n = 4). \tag{39}$$

Next we compute a Householder symplectic $H(2, w)$ to zero a_{31} and a_{41}, which can be achieved by applying Algorithm 3 with $k = 2$, $y = Be_1$, and $z = Ae_1$. Denote $G_1 = H(2, w)$, the result being

$$S = \begin{bmatrix} A & -B \\ B & A \end{bmatrix} := G_1 S G_1^T = \left[\begin{array}{cccc|cccc} x & x & 0 & 0 & 0 & 0 & 0 & 0 \\ x & x & x & x & 0 & 0 & x & x \\ 0 & x & x & x & 0 & x & 0 & x \\ 0 & x & x & x & 0 & x & x & 0 \\ \hline 0 & 0 & 0 & 0 & x & x & 0 & 0 \\ 0 & 0 & x & x & x & x & x & x \\ 0 & x & 0 & x & 0 & x & x & x \\ 0 & x & x & 0 & 0 & x & x & x \end{array}\right] \quad (n = 4). \tag{40}$$

Note that the (1, 3) and (1, 4) positions of A are zeroed. This completes the zeroing in the first column of S. Likewise we can zero the 2th of S with the property that

$$S = \begin{bmatrix} A & -B \\ B & A \end{bmatrix} = \left[\begin{array}{cccc|cccc} x & x & 0 & 0 & 0 & 0 & 0 & 0 \\ x & x & x & 0 & 0 & 0 & 0 & 0 \\ 0 & x & x & x & 0 & 0 & 0 & x \\ 0 & 0 & x & x & 0 & 0 & x & 0 \\ \hline 0 & 0 & 0 & 0 & x & x & 0 & 0 \\ 0 & 0 & 0 & 0 & x & x & x & 0 \\ 0 & 0 & 0 & x & 0 & x & x & x \\ 0 & 0 & x & 0 & 0 & 0 & x & x \end{array}\right] \quad (n = 4). \tag{41}$$

Notice that only the $(3, 4)$ and $(4, 3)$ positions of B are nonzero; they are zeroed only by applying Givens symplectic similarity transformation $G_3 = J(4, \theta)$ to S and we get

$$S = G_3 S G_3^T = \begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix} = \left[\begin{array}{cccc|cccc} x & x & 0 & 0 & 0 & 0 & 0 & 0 \\ x & x & x & 0 & 0 & 0 & 0 & 0 \\ 0 & x & x & x & 0 & 0 & 0 & 0 \\ 0 & 0 & x & x & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & x & x & 0 & 0 \\ 0 & 0 & 0 & 0 & x & x & x & 0 \\ 0 & 0 & 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & 0 & 0 & x & x \end{array}\right] \quad (n = 4). \tag{42}$$

This completes the zeros of S with $n = 4$. The similar method can be applied to zero the general matrix of order $2n$. Overall we have the following procedure "Reduction to block symmetry tridiagonal matrix." Given matrix

$$S = \begin{bmatrix} A & -B \\ B & A \end{bmatrix} \quad \text{with} \quad A^T = A, \; B^T = -B. \tag{43}$$

The following algorithm updates A to the symmetry tridiagonal matrix T; meanwhile B and $-B$ are reduced into an $n \times n$ zero matrix; moreover we have $\lambda(S) = \lambda(T) \sqcup \lambda(T)$.

Algorithm 10 (reduction to block symmetry tridiagonal matrix).

For $k = 1, \ldots, n - 2$ do:

if $k \leq n - 2$ do:

(1) Apply Algorithm 3 with $y = Ae_k$ and $z = Be_k$ to determine $H_k = H(k + 1, w)$. Update is as follows:

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix} := H_k \begin{bmatrix} A & -B \\ B & A \end{bmatrix} H_k^T. \tag{44}$$

(2) Apply Algorithm 5 with $y = Ae_k$ and $z = Be_k$ to determine $J_k = J(k + 1, \theta)$. Updates are as follows:

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix} := J_k \begin{bmatrix} A & -B \\ B & A \end{bmatrix} J_k^T. \tag{45}$$

(3) Apply Algorithm 3 with $y = Be_k$, $z = Ae_k$ to determine $G_k = H(k + 1, w)$. Updates are as follows:

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix} := G_k \begin{bmatrix} A & -B \\ B & A \end{bmatrix} G_k^T. \tag{46}$$

If $k = n - 1$ do the folowing.

(4) Apply Algorithm 5 with $y = Ae_{n-1}$ and $z = Be_{n-1}$ to determine $J_{n-1} = J(n, \theta)$. Update is as follows:

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix} := J_{n-1} \begin{bmatrix} A & -B \\ B & A \end{bmatrix} J_{n-1}^T. \tag{47}$$

After completion, we get the block symmetry tridiagonal matrix $\left[\begin{smallmatrix} T & 0 \\ 0 & T \end{smallmatrix}\right]$, where T is a symmetry tridiagonal matrix. The orthogonal matrix Q^T is given by

$$Q^T = J_{n-1} \left(G_{n-2} J_{n-2} H_{n-2}\right) \cdots \left(G_2 J_2 H_2\right) \left(G_1 J_1 H_1\right). \tag{48}$$

The method of reduction S into block symmetry tridiagonal matrix is given by Algorithm 10; in the former $n - 2$ steps, each step we premultiply by one Givens symplectic similarity and two Householder symplectic similarity and by postmultiply their transpose. Finally in the $n - 1$ step, only the Givens symplectic similarity is done. According to (48), it is clear that Q^T is the product of $3(n - 2) + 1$ orthogonal matrices, which is backward stability.

Algorithm 11 (the overall algorithm).

(1) Transform Hermitian matrix H into the symmetry and skew-Hamiltonian matrix:

$$S = \begin{bmatrix} A & -B \\ B & A \end{bmatrix} \quad \text{where} \quad A^T = A, \; B^T = -B. \tag{49}$$

(2) Apply Algorithm 10 to reduce S into a block symmetry tridiagonal matrix:

$$Q^T H Q = \begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix}. \tag{50}$$

(3) Compute eigenpairs (Λ, X) of T via QR.

(4) Setting

$$\begin{bmatrix} Q_{11} \\ Q_{21} \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{bmatrix} \begin{bmatrix} X \\ 0 \end{bmatrix} = \begin{bmatrix} Q_1 X \\ -Q_2 X \end{bmatrix}, \tag{51}$$

then Λ is eigenvalue of H, and $Q_{11} + iQ_{21}$ is the eigenvector of H.

4. Implementation Aspects

When implementing Algorithm 10, the matrix S must be taken full advantage of in order to avoid superfluous calculation and storage. In this section there are two parts; the first is to consider the computational cost and storage for the reduction to this block symmetry tridiagonal matrix, and the second is the computational cost and storage of the eigenvector Q^T.

Due to the special structure $A = A^T$ and $B^T = -B$, shared by A and B, only n^2 workspace is required for matrix

$$S = \begin{bmatrix} A & -B \\ B & A \end{bmatrix} \quad \text{where} \quad A^T = A, \; B^T = -B. \tag{52}$$

Now we consider the computation cost for reducing S to the block symmetry tridiagonal form, for Householder symplectic similarity, let

$$H_k = H(k+1, w) = \operatorname{diag}(I_k, P, I_k, P), \tag{53}$$

$$S = \begin{array}{c} \left[\begin{array}{cccc} A_{11} & A_{21}^T & B_{11}^T & B_{21}^T \\ A_{21} & A_{22} & -B_{21}^T & B_{22}^T \\ B_{11} & -B_{21}^T & A_{11} & A_{12} \\ B_{21} & B_{22} & A_{12}^T & A_{22} \end{array}\right] \begin{array}{c} k \\ n-k \\ k \\ n-k \end{array} \\ \begin{array}{cccc} k & n-k & k & n-k \end{array} \end{array}, \tag{54}$$

then

$$H_kSH_k = \begin{array}{c} \left[\begin{array}{cccc} A_{11} & A_{12}P & B_{11}^T & B_{21}^TP \\ PA_{12}^T & PA_{22}P & -PB_{21}^T & PB_{22}^T \\ B_{11} & -B_{21}^TP & A_{11} & A_{12} \\ PB_{21} & PB_{22}P & PA_{12}^T & PA_{22}P \end{array}\right] \begin{array}{c} k \\ n-k \\ k \\ n-k \end{array} \\ \begin{array}{cccc} k & n-k & k & n-k \end{array} \end{array}. \tag{55}$$

In kth step, updating S to H_kSH_k is equal to two ordinary Householder similarities for $n \times n$ matrix, one for symmetry matrix A and one for skew-symmetry matrix B, so we only need to compute PA_{12}^T, $PA_{22}P$, PB_{21}, and $PB_{22}P$. Likewise, we can update S to G_kSG_k by the same way.

Algorithm 12 (Householder symplectic similarity).

```
For $k = 1 : n-2$
    $[v, \beta] = \text{house}(B(k+1:n, k))$,
    $w = \beta B(k+1:n, k:n)^T v$,
    $B(k+1:n, k:n) = B(k+1:n, k:n) - vw^T$,
    $w = \beta B(k:n, k+1:n)v^T$,
    $B(k:n, k+1:n) = B(k:n, k+1:n) - wv^T$,        (56)
    $w = \beta A(k+1:n, k:n)^T v$,
    $A(p+1:n, p:n) = A(k+1:n, k:n) - vw^T$,
    $w = \beta A(k:n, k+1:n)v$,
    $A(k:n, k+1:n) = A(k:n, k+1:n) - wv^T$
end.
```

Updating by Givens symplectic similarity is equally simple; setting $c = \cos(\theta)$, $s = \sin(\theta)$ then

$$J_k = J(k+1, \theta) = \begin{bmatrix} I_k & & & 0_k & & \\ & c & & & s & \\ & & I_{n-k-1} & & & 0_{n-k-1} \\ 0_k & & & I_k & & \\ & -s & & & c & \\ & & 0_{n-k-1} & & & I_{n-k-1} \end{bmatrix},$$

$$S = \begin{array}{c} \left[\begin{array}{cccccc} A_{11} & A_{12} & A_{13} & -B_{11} & -B_{12} & -B_{13} \\ A_{21} & A_{22} & A_{23} & -B_{21} & -B_{22} & -B_{23} \\ A_{31} & A_{32} & A_{33} & -B_{31} & -B_{32} & -B_{33} \\ B_{11} & B_{12} & B_{13} & A_{11} & A_{12} & A_{13} \\ B_{21} & B_{22} & B_{23} & A_{21} & A_{22} & A_{23} \\ B_{31} & B_{32} & B_{33} & A_{31} & A_{32} & A_{33} \end{array}\right] \begin{array}{c} k+1 \\ 1 \\ n-k-2 \\ k+1 \\ 1 \\ n-k-2 \end{array} \\ \begin{array}{cccccc} k+1 & 1 & n-k-2 & k & 1 & n-k-2 \end{array} \end{array}. \tag{57}$$

Denoting $S_J = (J_kSJ_k^T)$, then we have

$$S_J(1:2n, 1:n) = \begin{array}{c} \left[\begin{array}{ccc} A_{11} & cA_{12} - sB_{12} & A_{31}^T \\ cA_{21} + sB_{21} & A_{22} & A_{32}^T \\ A_{31} & cA_{32} - sB_{32} & A_{33} \\ B_{11} & cB_{12} + sA_{12} & B_{13} \\ cB_{21} - sA_{21} & B_{22} & cB_{23} - SA_{23} \\ B_{31} & cB_{32} + sA_{32} & B_{33} \end{array}\right] \begin{array}{c} k+1 \\ 1 \\ n-k-2 \\ k+1 \\ 1 \\ n-k-2 \end{array} \\ \begin{array}{ccc} k+1 & 1 & n-k-2 \end{array} \end{array}. \tag{58}$$

Note that updating S into $J_kSJ_k^T$ such that $B(k+1, k) = 0$ affects the rows (resp. the columns) $k+1$, $n+k+1$ of S, that is, the rows(columns) $k+1$, $n+k+1$ of A and B. According to (58), we only need to compute $cA_{21} + sB_{21}$, $cA_{32} - sB_{32}$, $cB_{21} - sA_{21}$ and $cB_{32} + sA_{32}$ because S_J is also symmetry and skew-Hamiltonian matrix. Therefore, only $2(n-k)$ flops are required in kth step, altogether $\sum_{k=1}^{n-1} 2(n-k) = n^2 - n$ flops. Bellow is the algorithm for $J_kSJ_k^T$ in kth step.

Algorithm 13 (Givens symplectic similarity). Consider

```
$a = A(k+1, k)$,    $b = B(k+1, k)$,
$[c, s] = \text{givens}(a, b)$,
    $A(k+1, k) = cA(k+1, k) + sB(k+1, k)$,
    $A(k, k+1) = A(k+1, k)$,
for $i = k+2 : n$
        $S(i,k+1) = cA(i,k+1) - sB(i,k+1)$,
        $B(i,k+1) = cB(i,k+1) + sA(i,k+1)$,
        $B(k+1,i) = -B(i,k+1)$,
        $A(i,k+1) = S(i,k+1)$,
        $A(k+1,i) = A(i,k+1)$
end
        $B(k+1,k) = 0$,
        $B(k,k+1) = 0$.                                    (59)
```

Overall $(4/3)n^3 \times 4 + \mathcal{O}(n^2) = (16/3)n^3 + \mathcal{O}(n^2)$ flops are required for reducing S to $\left[\begin{smallmatrix} T & 0 \\ 0 & T\end{smallmatrix}\right]$.

This reduction includes 4 times computational cost of Householder tridiagonalized for $n \times n$ symmetry matrix and $n^2 - n$ flops for Givens similarity transformation. Since one complex operation is four times of real operation, $(16/3)n^3 + \mathcal{O}(n^2)$ flops are required for complex Hermitian tridiagonalization.

Now we consider the computation and storage problem for eigenvectors

$$Q^T = J_{n-1}(G_{n-2}J_{n-2}H_{n-2})\cdots(G_2J_2H_2)(G_1J_1H_1). \quad (60)$$

To obtain Q^T, we can premultiply only by $n \times n$ matrices by using the special structure of J_k, G_k, and H_k though they are $2n$-by-$2n$ matrices. When premultiplying by Householder symplectics H_k, let a $2n \times 2n$ matrix Q^T be of the form

$$Q^T = \begin{bmatrix} Q_{N\times N} & -U_{N\times N} \\ U_{N\times N} & Q_{N\times N}\end{bmatrix} = \begin{array}{c} \left[\begin{array}{cccc} Q_{11} & Q_{12} & -U_{11} & -U_{12} \\ Q_{21} & Q_{22} & -U_{21} & -U_{22} \\ U_{11} & U_{12} & Q_{11} & Q_{12} \\ U_{21} & U_{22} & Q_{21} & Q_{22} \end{array}\right] \begin{array}{c} k \\ n-k \\ k \\ n-k \end{array} \\ \begin{array}{cccc} k & n-k & k & n-k \end{array} \end{array}. \quad (61)$$

So two n^2 workspace is required for Q^T, one for $Q_{N\times N}$ and one for $U_{N\times N}$. Premultiplying by $H_k = \operatorname{diag}(I_{k+1}, P_A, I_{k+1}, P_A)$ in kth step, we have

$$H_kQ^T = \begin{array}{c} \left[\begin{array}{cccc} Q_{11} & Q_{12} & -U_{11} & -U_{12} \\ P_AQ_{21} & P_AQ_{22} & -P_AU_{21} & -P_AU_{22} \\ U_{11} & U_{12} & Q_{11} & Q_{12} \\ P_AU_{21} & P_AU_{22} & P_AQ_{21} & P_AQ_{22} \end{array}\right] \begin{array}{c} k \\ n-k \\ k \\ n-k \end{array} \\ \begin{array}{cccc} k & n-k & k & n-k \end{array} \end{array}. \quad (62)$$

Therefore, only Q_{21}, Q_{22} and the real part of Q^T are updated to P_AQ_{21}, P_AQ_{22} and also imaginary part U_{21}, U_{22} updated to P_AU_{21}, P_AU_{22}. The main work is computing the product of Householder matrix $H = I - \beta vv^T \in \mathbb{R}^{(n-k)\times(n-k)}$ and $[Q_{21} \;\; Q_{22}] \in \mathbb{R}^{(n-k)\times(n)}$, and the product of H and $[U_{21} \;\; U_{22}] \in \mathbb{R}^{(n-k)\times(n)}$, where the first columns of Q_{21} and U_{21} are zero. So $4(n-k)(n-k-1)$ flops are required each step, altogether $\sum_{k=1}^{n-2} 8(n-k)(n-k-1) = (8/3)n^3 + \mathcal{O}(n^2)$ flops. Likewise premultiplying by G_k requires other $(8/3)n^3 + \mathcal{O}(n^2)$ flops.

Algorithm 14 (computation of H_kQ^T).
For $k = 1 : n-2$

```
    $[v,\beta] = \text{house}(B(k+1:n,k))$,
    $w = \beta Q1(k+1:n,2:n)^T v$,
    $Q1(k+1:n,2:n) = Q1(k+1:n,2:n) - vw^T$,        (63)
    $w = \beta Q2(k+1:n,2:n)^T v$,
    $Q2(k+1:n,2:n) = Q2(k+1:n,2:n) - vw^T$
end.
```

For Givens symplectic matrix J_k, set

$$Q^T = \begin{array}{c} \left[\begin{array}{cccccc} Q_{11} & Q_{12} & Q_{13} & -U_{11} & -U_{12} & -U_{13} \\ Q_{21} & Q_{22} & Q_{23} & -U_{21} & -U_{22} & -U_{23} \\ Q_{31} & Q_{32} & Q_{33} & -U_{31} & -U_{32} & -U_{33} \\ U_{11} & U_{12} & U_{13} & Q_{11} & Q_{12} & Q_{13} \\ U_{21} & U_{22} & U_{23} & Q_{21} & Q_{22} & Q_{23} \\ U_{31} & U_{32} & U_{33} & Q_{31} & Q_{32} & Q_{33} \end{array}\right] \begin{array}{c} k \\ 1 \\ n-k-1 \\ k \\ 1 \\ n-k-1 \end{array} \\ \begin{array}{cccccc} k & 1 & n-k-1 & k & 1 & n-k-1 \end{array} \end{array}. \quad (64)$$

Premultiplying by J_k, denoting $Q_J^T = J_kQ^T$, we have

$$Q_J^T(1:2n,1:n) = \begin{array}{c} \left[\begin{array}{ccc} Q_{11} & Q_{12} & Q_{13} \\ cQ_{21}+sU_{21} & cQ_{22}+sU_{22} & cQ_{23}+sU_{23} \\ Q_{31} & Q_{32} & Q_{33} \\ 0 & 0 & 0 \\ cU_{21}-sQ_{21} & cU_{22}-sQ_{22} & cU_{23}-sQ_{23} \\ 0 & 0 & 0 \end{array}\right] \begin{array}{c} k \\ 1 \\ n-k-1 \\ k \\ 1 \\ n-k-1 \end{array} \\ \begin{array}{ccc} k & 1 & n-k-1 \end{array} \end{array}. \quad (65)$$

Therefore, each step updating the $k+1$th and $n+k+1$th row of Q^T requires $4(n-1)$ flops, altogether $4(n-1)^2$ flops. Below is the algorithm.

Algorithm 15 (computation of J_kQ^T).
For $k = 1 : n-2$

```
        $T1 = Q1$,
        $T2 = Q2$,
```

$$\begin{aligned}
Q1(k+1,2:n) &= cQ1(k+1,2:n),\\
Q2(k+1,2:n) &= sQ2(k+1,2:n),\\
Q2(k+2:n,2:n) &= 0,\\
Q1(k+1:n,2:n) &= Q1(k+1:n,2:n)\\
&\quad + Q2(k+1:n,2:n),\\
T1(k+1,2:n) &= -sT1(k+1,2:n),\\
T1(k+2:n,2:n) &= 0,\\
T2(k+1,2:n) &= cT2(k+1,2:n),\\
Q2(k+1:n,2:n) &= T1(k+1:n,2:n)\\
&\quad + T2(k+1:n,2:n)
\end{aligned} \tag{66}$$

end.

Overall in each step we only need to premultiply two $n \times n$ Householder matrix and one $n \times n$ Givens matrix in the procedure of computing the eigenvector.

5. Numerical Experiments

All codes were run in MATLAB 7.04 in double precision and all experiments were performed on a personal computer with 1.41 Ghz central processing unit (AMD Sempron(tm) Processor 2500+), 520 MB memory, and Windows XP system. Machine epsilon $eps \approx 2.22 \times 10^{-16}$.

Example 16. To illustrate Algorithm 11 more clearly, we give a simple example and show the processes and results of Algorithm 11 of H_1:

$$H_1 = \begin{bmatrix} 1 & -3i & -4i \\ 3i & 1 & 4i \\ 4i & 4i & 1 \end{bmatrix}. \tag{67}$$

We first transform H_1 into symmetry and skew-Hamiltonian matrix S:

$$H \longrightarrow S = \begin{bmatrix} 1 & 0 & 0 & 0 & 3 & 4 \\ 0 & 1 & 0 & -3 & 0 & 4 \\ 0 & 0 & 1 & -4 & -4 & 0 \\ 0 & -3 & -4 & 1 & 0 & 0 \\ 3 & 0 & -4 & 0 & 1 & 0 \\ 4 & 4 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{68}$$

Then

$$S \longrightarrow H^T S H = \begin{bmatrix} 1 & 0 & 0 & 0 & 5 & 0 \\ 0 & 1 & 0 & -5 & 0 & -4 \\ 0 & 0 & 1 & 0 & 4 & 0 \\ 0 & -5 & 0 & 1 & 0 & 0 \\ 5 & 0 & 4 & 0 & 1 & 0 \\ 0 & -4 & 0 & 0 & 0 & 1 \end{bmatrix} \longrightarrow J^T H^T S H J$$

Table 1: Example 16—real fast algorithm for H_1 (Algorithm 11).

Eigenvalue	Eigenvector		
7.4031	0.5522	0.3534 + 0.4243i	−0.2650 + 0.5657i
−5.4031	0.5522	0.3534 − 0.4243i	−0.2650 − 0.5657i
1.0000	−0.6247	0.6247	−0.4685

Table 2: Example 16—complex algorithm (eig) for eigenproblem H_1.

Eigenvalue	Eigenvector		
7.4031	−0.2343 − 0.5000i	0.2343 − 0.5000i	0.6247
−5.4031	0.2343 − 0.5000i	−0.2343 − 0.5000i	−0.6247
1.0000	−0.6247 − 0.0000i	0.6247 + 0.0000i	−0.4685

$$= \begin{bmatrix} 1 & 5 & 0 & 0 & 0 & 0 \\ 5 & 1 & 4 & 0 & 0 & 0 \\ 0 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 5 & 0 \\ 0 & 0 & 0 & 5 & 1 & 4 \\ 0 & 0 & 0 & 0 & 4 & 1 \end{bmatrix} = \begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix}, \tag{69}$$

where

$$\begin{aligned}
H &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.6 & 0.8 & 0 & 0 & 0 \\ 0 & 0.8 & -0.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.6 & 0.8 \\ 0 & 0 & 0 & 0 & 0.8 & -0.6 \end{bmatrix},\\
J &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{70}$$

Denoting that (D, V) is the eigenpair of T

$$\begin{aligned}
V &= \begin{bmatrix} 0.5522 & -0.6247 & 0.5522 \\ -0.7071 & 0.0000 & 0.7071 \\ 0.4417 & 0.7809 & 0.4417 \end{bmatrix},\\
D &= \begin{bmatrix} -5.4031 & 0 & 0 \\ 0 & 1.0000 & 0 \\ 0 & 0 & 7.4031 \end{bmatrix},
\end{aligned} \tag{71}$$

then

$$\begin{aligned}
Q\begin{bmatrix} V \\ 0 \end{bmatrix} &= H_1 J \begin{bmatrix} V \\ 0 \end{bmatrix}\\
&= \begin{bmatrix} 0.5522 & -0.6247 & 0.5522 \\ 0.3534 & 0.6247 & 0.3534 \\ -0.2650 & -0.4685 & -0.2650 \\ 0 & 0 & 0 \\ -0.4243 & 0.0000 & 0.4243 \\ -0.5657 & 0.0000 & 0.5657 \end{bmatrix}
\end{aligned} \tag{72}$$

Table 3: Example 17—the real fast algorithm for eigenproblem of H_2.

Eigenvalue	Eigenvector				
16.6334	0.4207	$0.4417 + 0.1637i$	$0.4177 + 0.1008i$	$0.3343 + 0.0604i$	$0.5220 + 0.1688i$
−6.0472	0.4928	$0.4942 - 0.0179i$	$0.0460 + 0.0178i$	$-0.3757 - 0.1146i$	$-0.5966 + 0.0018i$
3.3730	−0.1369	$-0.1327 + 0.0555i$	$0.4347 + 0.1567i$	$-0.7694 - 0.2026i$	$0.3370 + 0.0198i$
−1.8356	−0.5240	$0.2953 - 0.2007i$	$0.4425 + 0.4636i$	$0.1982 + 0.1590i$	$-0.2902 - 0.1959i$
−0.1236	0.5356	$-0.5467 - 0.2943i$	$0.1736 + 0.3981i$	$0.0803 + 0.1618i$	$-0.0590 - 0.3210i$

Table 4: Example 17—the real fast algorithm for eigenproblem of H_2.

Eigenvalue	Eigenvector				
16.6334	$0.4003 - 0.1295i$	$0.4706 + 0.0198i$	$0.4285 - 0.0326i$	$0.3367 - 0.0454i$	0.5486
−6.0472	$0.4928 + 0.0015i$	$0.4942 - 0.0164i$	$0.0459 + 0.0179i$	$-0.3753 - 0.1158i$	−0.5966
3.3730	$0.1367 - 0.0080i$	$0.1293 - 0.0631i$	$-0.4431 - 0.1310i$	$0.7800 + 0.1572i$	−0.3376
−1.8356	$0.4343 - 0.2932i$	$-0.1325 + 0.3315i$	$-0.6262 - 0.1367i$	$-0.2532 - 0.0208i$	0.3502
−0.1236	$-0.0968 + 0.5268i$	$0.3882 - 0.4845i$	$-0.4229 + 0.0988i$	$-0.1736 + 0.0497i$	0.3263

Table 5: Example 17—the real fast algorithm for eigenproblem of H_2.

Real fast algorithm	Complex algorithm (eig)	Error
$2.022700124819288e + 001$	$2.022700124819288e + 001$	$7.105427357601002e - 015$
$1.902502989496266e + 001$	$1.902502989496268e + 001$	$1.421085471520200e - 014$
$1.800263600389322e + 001$	$1.800263600389323e + 001$	$1.065814103640150e - 014$
$1.700158594984117e + 001$	$1.700158594984119e + 001$	$2.131628207280301e - 014$
$1.600156256127496e + 001$	$1.600156256127496e + 001$	0
$1.500156225818778e + 001$	$1.500156225818779e + 001$	$8.881784197001252e - 015$
$1.400156222655317e + 001$	$1.400156222655317e + 001$	$3.552713678800501e - 015$
$1.300156061869106e + 001$	$1.300156061869103e + 001$	$2.842170943040401e - 014$
$1.200151108208256e + 001$	$1.200151108208256e + 001$	$3.552713678800501e - 015$
$1.100086192964745e + 001$	$1.100086192964745e + 001$	0
$9.999138070352572e + 000$	$9.999138070352565e + 000$	$7.105427357601002e - 015$
$8.998488917917468e + 000$	$8.998488917917442e + 000$	$2.664535259100376e - 014$
$7.998439381308970e + 000$	$7.998439381308969e + 000$	$8.881784197001252e - 016$
$6.998437773446823e + 000$	$6.998437773446838e + 000$	$1.509903313490213e - 014$
$5.998437741812211e + 000$	$5.998437741812216e + 000$	$5.329070518200751e - 015$
$4.998437438725036e + 000$	$4.998437438725052e + 000$	$1.598721155460225e - 014$
$3.998414050158813e + 000$	$3.998414050158812e + 000$	$8.881784197001252e - 016$
$2.997363996106763e + 000$	$2.997363996106775e + 000$	$1.199040866595169e - 014$
$1.974970105037331e + 000$	$1.974970105037334e + 000$	$3.774758283725532e - 015$
$7.729987518071178e - 001$	$7.729987518071133e - 001$	$4.440892098500626e - 015$

is the first half of eigenvectors of S. The computing eigenproblem of H_1 is showed in Table 1.

According to Tables 1 and 2, it can be seen that the eigenvectors are different except the one corresponding to 1. In fact, by Table 1, the eigenvector corresponding to 7.4031 is $v^{(1)} = (0.5522, 0.3534 + 0.4243i, -0.2650 + 0.5657i)^T$, and $v^{(2)} = (-0.2343 - 0.5000i, 0.2343 - 0.5000i, 0.6247)^T$ by Table 2. It is easy to prove that rank $(v^{(1)}, v^{(2)}) = 1$, that is, $v^{(1)} = e^{i\theta} v^{(2)}$. So the complex eigenvector after normalizing is not unique, which is the biggest difference from real eigenvector.

Example 17. Now we give a little bigger matrix H_2; we depict its eigenproblem in Tables 3 and 4, respectively:

$$H_2 = \begin{bmatrix} 1 & 2+i & 3+i & 4+i & 5+i \\ 2-i & 2 & 3-1i & 4 & 6 \\ 3-i & 3+1i & 3 & 1 & 5 \\ 4-i & 4 & 1 & 3 & 1+i \\ 5-i & 6 & 5 & 1-i & 3 \end{bmatrix}. \quad (73)$$

According to tables, we can see that the eigenvectors are complex and it is easy to prove that the two eigenvectors

Table 6: Example 17—the real fast algorithm for eigenproblem of H_2.

Real fast algorithm	Complex algorithm (eig)	Error
$2.002255917401552e+002$	$2.002255917401532e+002$	$2.074784788419493e-012$
$1.990236230837739e+002$	$1.990236230837733e+002$	$6.252776074688882e-013$
$1.980012299298954e+002$	$1.980012299298937e+002$	$1.676880856393836e-012$
$1.970001799412475e+002$	$1.970001799412465e+002$	$9.663381206337363e-013$
$1.960001565550514e+002$	$1.960001565550502e+002$	$1.193711796076968e-012$
$1.950001562523487e+002$	$1.950001562523486e+002$	$1.136868377216160e-013$
$1.940001562497741e+002$	$1.940001562497715e+002$	$2.586375558166765e-012$
$1.930001562497571e+002$	$1.930001562497559e+002$	$1.165290086646564e-012$
$1.920001562497562e+002$	$1.920001562497558e+002$	$4.547473508864641e-013$
$1.910001562497567e+002$	$1.910001562497556e+002$	$1.193711796076968e-012$
$1.900001562497573e+002$	$1.900001562497560e+002$	$1.364242052659392e-012$
$1.890001562497554e+002$	$1.890001562497561e+002$	$7.389644451905042e-013$
$1.880001562497553e+002$	$1.880001562497555e+002$	$1.989519660128281e-013$
$1.870001562497555e+002$	$1.870001562497559e+002$	$3.694822225952521e-013$
$1.860001562497558e+002$	$1.860001562497558e+002$	$8.526512829121202e-014$
$1.850001562497552e+002$	$1.850001562497558e+002$	$5.400124791776761e-013$
$1.840001562497563e+002$	$1.840001562497560e+002$	$2.842170943040401e-013$
$1.830001562497566e+002$	$1.830001562497558e+002$	$8.526512829121202e-013$
$1.820001562497562e+002$	$1.820001562497559e+002$	$3.410605131648481e-013$
$1.810001562497546e+002$	$1.810001562497558e+002$	$1.193711796076968e-012$

corresponding to the same eigenvalue are linear dependent, so the eigenvectors, which are computed by the real fast algorithm (Algorithm 11) span the same space as complex algorithm (eig).

Example 18. In this example we construct complex Hermitian matrices of the form

$$H_3 = \operatorname{diag}(1,2,\dots,2*n) + \begin{bmatrix} 0 & -\frac{1}{2} & & & \\ -\frac{1}{2} & 0 & -\frac{1}{2} & & \\ & -\frac{1}{2} & \ddots & \ddots & \\ & & \ddots & \ddots & -\frac{1}{2} \\ & & & -\frac{1}{2} & 0 \end{bmatrix}_{2n\times 2n} + i * \begin{bmatrix} 0 & B \\ -B & 0 \end{bmatrix}, \tag{74}$$

where $B = (1/8)\operatorname{diag}(\operatorname{eye}(n))$. In tables we only give a few eigenvalues with largest modulus and compare it with complex algorithm (eig). Let $n = 10, 100$; that is, the order of matrix H_3 is 20, 200, respectively. The eigenvalues and errors, which are computed respectively by real fast algorithm and complex algorithm (eig), are showed in Tables 5 and 6. All the eigenvalues of matrix of order 20 are given, while only the 20th largest modulus eigenvalues of matrix order 200 are present. It is easy to prove that eigenvectors corresponding to the same eigenvalue are linear dependent. According to the tables, we have the same result as the complex algorithm (eig), but only the real operation is used.

Table 7: Example 20—relative error.

$2*N$	Max (relerr)	Min (relerr)	Average (relerr)
200	$0.2645e-013$	0	$0.0288e-013$
300	$0.7872e-013$	0	$0.0447e-013$
400	$0.1949e-013$	0	$0.0242e-013$
500	$0.3536e-013$	0	$0.0287e-013$
600	$0.3297e-012$	0	$0.0058e-012$
700	$0.2149e-013$	0	$0.0292e-013$
800	$0.1773e-011$	0	$0.0007e-011$
900	$0.2841e-012$	0	$0.0037e-012$
1000	$0.6865e-012$	0	$0.0046e-012$

Example 19. Complex Hermitian matrices are constructed by Hilbert matrix, and the MATLAB code is as follows:

$$n = 10; \tag{75}$$

$$H = \operatorname{hilb}(n); \tag{76}$$

$$H_4 = H + \left(\operatorname{tril}(H) - \operatorname{diag}(\operatorname{diag}(H))\right)' i - \left(\operatorname{tril}(H) - \operatorname{diag}(\operatorname{diag}(H))\right). \tag{77}$$

TABLE 8: Example 20—residual $r = Hv_i - v_i\lambda$.

$2 * N$	$\max_{i=1,\dots,N}(Hv_i - v_i\lambda)$	$\min_{i=1,\dots,N}(Hv_i - v_i\lambda)$	*Average* $(Hv_i - v_i\lambda)$
200	$0.1140e - 012$	$0.0113e - 012$	$0.0282e - 012$
300	$0.2110e - 012$	$0.0168e - 012$	$0.0420e - 012$
400	$0.2275e - 012$	$0.0225e - 012$	$0.0556e - 012$
500	$0.3608e - 012$	$0.0271e - 012$	$0.0682e - 012$
600	$0.6141e - 012$	$0.0324e - 012$	$0.0819e - 012$
700	$0.9271e - 012$	$0.0396e - 012$	$0.0954e - 012$
800	$0.8451e - 012$	$0.0452e - 012$	$0.1080e - 012$
900	$0.6920e - 012$	$0.0496e - 012$	$0.1175e - 012$
1000	$0.1218e - 011$	$0.0056e - 011$	$0.0132e - 011$

When $n = 10$, the 6 largest modulus eigenvalues are present and 3 eigenvectors corresponding to the 3 largest modulus eigenvalues are given here:

$$\lambda(H_4)_{\text{eig}} = \begin{Bmatrix} 2.13798182520799 \\ 0.51461933549659 \\ -0.49873888113430 \\ -0.18865833608333 \\ 0.17747332069964 \\ -0.08500068573387 \end{Bmatrix}, \quad \lambda(H_4)_{\text{Algorithm 11}} = \begin{Bmatrix} 2.13798182520799 \\ 0.51461933549659 \\ -0.49873888113430 \\ -0.18865833608333 \\ 0.17747332069964 \\ -0.08500068573387 \end{Bmatrix}, \tag{78}$$

$$V_{(\text{eig})} = \begin{bmatrix} -0.1267 - 0.0857i & -0.0156 - 0.1526i & 0.1174 + 0.0100i \\ 0.0529 + 0.2493i & -0.0069 + 0.2192i & -0.1508 - 0.1187i \\ 0.0647 - 0.2646i & -0.2177 - 0.2291i & 0.1777 + 0.1521i \\ -0.3229 + 0.1315i & 0.3057 - 0.0144i & -0.0986 - 0.2811i \\ 0.2230 + 0.2709i & -0.1554 + 0.3121i & -0.0529 + 0.2981i \\ 0.2089 - 0.2644i & -0.2960 - 0.2141i & 0.2578 - 0.2379i \\ -0.2801 - 0.2083i & 0.2142 - 0.2740i & -0.3615 - 0.0351i \\ -0.2635 + 0.2411i & 0.2822 + 0.1932i & 0.2113 + 0.3073i \\ 0.1389 + 0.3183i & -0.1273 + 0.3242i & 0.1640 - 0.3632i \\ 0.3304 & -0.3474 & -0.4016 \end{bmatrix}, \tag{79}$$

$$Q = \begin{bmatrix} -0.6616 & -0.4938 & -0.3499 \\ -0.3850 - 0.2226i & 0.1121 - 0.3048i & -0.0667 + 0.5036i \\ -0.2449 - 0.2266i & 0.2977 + 0.0431i & 0.2830 + 0.3297i \\ -0.1682 - 0.2102i & 0.1960 + 0.2453i & 0.3451 + 0.0933i \\ -0.1203 - 0.1921i & 0.0573 + 0.3070i & 0.2918 - 0.0624i \\ -0.0880 - 0.1754i & -0.0550 + 0.2958i & 0.2086 - 0.1463i \\ -0.0649 - 0.1608i & -0.1329 + 0.2524i & 0.1273 - 0.1825i \\ -0.0477 - 0.1480i & -0.1816 + 0.1980i & 0.0581 - 0.1897i \\ -0.0346 - 0.1368i & -0.2082 + 0.1428i & 0.0028 - 0.1802i \\ -0.0243 - 0.1270i & -0.2190 + 0.0916i & -0.0394 - 0.1614i \end{bmatrix}. \tag{80}$$

we prove that $\text{rank}([V(:,k)Q(:,k)]) = 1$ for $k = 1, 2, 3$. It is very interesting that the eigenvectors computed from our algorithm are unique but not the complex algorithm (eig).

Example 20. At last we construct some rand Hermitian matrices of order N; its MATLAB code is as follows.

$$A = \text{rand}(n), \qquad SA = A' + A, \tag{81}$$

$$B = \text{rand}(n), \qquad SB = B - B', \tag{82}$$

$$H = SA + i^* SB. \tag{83}$$

Letting $2 * N = 200 : 100 : 1000$, denote

$$\max(\text{relerr}) = \max_j \frac{\left|\lambda_j^{\text{eig}} - \lambda_j^{\text{Sympl}}\right|}{\left|\lambda_j^{\text{Sympl}}\right|}, \tag{84}$$

$$\min(\text{relerr}) = \min_j \frac{\left|\lambda_j^{\text{eig}} - \lambda_j^{\text{Sympl}}\right|}{\left|\lambda_j^{\text{Sympl}}\right|}. \tag{85}$$

All the relative errors we compute are present by a factor of order $\mathcal{O}(10^d)$. The average relative errors for all examples of each dimension we computed for each example in Table 7 are denoted by average (relerr). The max, min, and average errors of $r = Hv_i - v_i\lambda$ are also considered in Table 8. According to two tables, all the "relerr" and "residual" are order $\mathcal{O}(10^{-13})$ and very small. Hence it can be concluded that eigenproblems computed by the real fast algorithm have high precision, and the method always converges.

Acknowledgments

This work is supported by National Natural Science Foundation of China nos. 11101087 and 10261012 and supported by Research Start Foundation of Fuzhou University nos. 022427 and 600629.

References

[1] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Clarendon, New York, NY, USA, 1965.

[2] P. Benner, H. Faßbender, and M. Stoll, "A Hamiltonian Krylov-Schur-type method based on the symplectic Lanczos process," *Linear Algebra and its Applications*, vol. 435, no. 3, pp. 578–600, 2011.

[3] A. Bunse-Gerstner, R. Byers, and V. Mehrmann, "A chart of numerical methods for structured eigenvalue problems," *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 2, pp. 419–453, 1992.

[4] R. Byers, "A Hamiltonian *QR* algorithm," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 1, pp. 212–229, 1986.

[5] J.-S. Guo, W.-W. Lin, and C.-S. Wang, "Numerical solutions for large sparse quadratic eigenvalue problems," *Linear Algebra and Its Applications*, vol. 225, pp. 57–89, 1995.

[6] T.-M. Hwang, W.-W. Lin, and V. Mehrmann, "Numerical solution of quadratic eigenvalue problems with structure-preserving methods," *SIAM Journal on Scientific Computing*, vol. 24, no. 4, pp. 1283–1302, 2003.

[7] W.-W. Lin, V. Mehrmann, and H. Xu, "Canonical forms for Hamiltonian and symplectic matrices and pencils," *Linear Algebra and Its Applications*, vol. 302-303, pp. 469–533, 1999.

[8] C. F. van Loan, "A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix," *Linear Algebra and Its Applications*, vol. 61, pp. 233–251, 1984.

[9] D. S. Watkins, "On Hamiltonian and symplectic Lanczos processes," *Linear Algebra and Its Applications*, vol. 385, pp. 23–45, 2004.

[10] H. Xu, *Solving algebraic Riccati Equations via Skew-Hamiltonian matrices [Ph.D. thesis]*, Institute of Mathematics. Fudan University, Shanghai, China, 1993.

[11] H. G. Xu and L. Z. Lu, "Properties of a quadratic matrix equation and the solution of the continuous-time algebraic Riccati equation," *Linear Algebra and Its Applications*, vol. 222, pp. 127–145, 1995.