Front. Math. China 2021, 16(2): 345–379
https://doi.org/10.1007/s11464-021-0909-6

RESEARCH ARTICLE

Computing top eigenpairs of Hermitizable matrix

Mu-Fa CHEN[1,2,3], **Zhi-Gang JIA**[1,4], **Hong-Kui PANG**[1,4]

1 Research Institute of Mathematical Science, Jiangsu Normal University, Xuzhou 221116, China
2 School of Mathematical Sciences, Beijing Normal University, Beijing 100875, China
3 Laboratory of Mathematics and Complex Systems (Beijing Normal University), Ministry of Education, Beijing 100875, China
4 School of Mathematics and Statistics, Jiangsu Normal University, Xuzhou 221116, China

© Higher Education Press 2021

Abstract The top eigenpairs at the title mean the maximal, the submaximal, or a few of the subsequent eigenpairs of an Hermitizable matrix. Restricting on top ones is to handle with the matrices having large scale, for which only little is known up to now. This is different from some mature algorithms, that are clearly limited only to medium-sized matrix for calculating full spectrum. It is hoped that a combination of this paper with the earlier works, to be seen soon, may provide some effective algorithms for computing the spectrum in practice, especially for matrix mechanics.

Keywords Hermitizable, Householder transformation, birth-death matrix, isospectral matrices, top eigenpairs, algorithm

MSC2020 15A18, 15A57, 60J27, 65F10, 65F15, 65F30

This paper is a continuation of [7] which surveys partially the results (algorithms) presented in [3–5], plus some additional materials. The main context in [7] is on real tridiagonal matrix, except few comments on the complex situation. In the real context, the theoretical study on the leading spectrum of the infinitesimal matrix operator is reviewed in [2]. This paper starts at a computational technique for checking the Hermitizability and then goes to study the Householder transformation, and furthermore the submaximal eigenpair for Hermitizable matrices. The algorithms can also be used to compute a few number of the other subsequent eigenpairs. The price we have to pay is mainly for the Householder transformation (Algorithm 3) which is a famous algorithm having complexity $O(N^3)$. The other algorithms in the paper are mainly $O(N)$ algorithm. In Section 4 of the paper, except some remarks on

Received July 26, 2020; accepted December 4, 2020
Corresponding author: Mu-Fa CHEN, E-mail: mfchen@bnu.edu.cn

our algorithms, a proof of a key result, an isospectral property of the Hermitizable matrix and a Jacobi (birth-death) one, originally given in [5], is presented. The last section of the paper is devoted to the practical implementation of the results obtained in the previous sections on large scale matrices. Some additional analysis and the programs in MatLab of the algorithms, as well as a number of tests in comparison with the known programs are presented.

1 Checking the Hermitizability

Let $A = (a_{ij})_{i,j=0}^{N}$ be a given complex matrix. We are going to check by computer its Hermitizability introduced in [5]: there exists a positive measure μ such that

$$\mu_i a_{ij} = \mu_j \bar{a}_{ji}, \quad \forall\, i, j, \tag{1}$$

where $\bar{a}$ denotes the conjugate of a. Note that we have a very simple necessary condition for the property (1): for each pair (i, j), either $a_{ij} = 0$ and $a_{ji} = 0$ simultaneously, or $a_{ij}a_{ji} > 0$ (cf. [5]). In particular, $(a_{ii})_{i=0}^{N}$ must be real. However, for the criterion of the Hermitizability, one more condition is essential: the so-called circle condition. The analytic method for checking the circle condition was given in [5; Theorem 5]. Here we introduce an algorithm for checking the condition by computer. Define a column vector $\mathbb{1}$ having elements 1 everywhere and denote by $\mathrm{Diag}(u)$ the diagonal matrix with vector u as its diagonal elements. For simplicity, let $B = A - \mathrm{Diag}(\bar{A}\mathbb{1})$. Denote by $B^{\setminus\{\text{last line}\}}$ the matrix obtained from B by removing its last line.

The checking procedure consists of three steps.

Algorithm 1 (1) *Computing the harmonic measure.* Consider the (row-) harmonic equation: $\mu B = 0$ with $\mu_0 \neq 0$. Assume that there exists at least one non-zero solution μ. Equivalently, the equation

$$B^{*\setminus\{\text{last line}\}}\mu^* = 0$$

has at least one solution $(\mu_0, \ldots, \mu_N)$ with fixed boundary condition, say $\mu_0 = 1$ for instance. Actually, in the Hermitizable case, the resulting measure μ must be positive (and is indeed unique under the irreducible condition, cf. [5]), then we can go to the next step. Otherwise, the matrix A is not Hermitizable.

(2) Define the *quasi-Hermitizing matrix* as follows:

$$\hat{A} = \mathrm{Diag}(\mu^{1/2})A\mathrm{Diag}(\mu^{-1/2}), \quad \hat{a}_{ij} = \frac{\sqrt{\mu_i}\, a_{ij}}{\sqrt{\mu_j}}, \quad \forall\, i, j. \tag{2}$$

(3) *Hermitizability criterion.* Now, A is Hermitizable if and only if $\hat{A} = \hat{A}^H$, where the superscript H means the conjugate transpose.

The next example illustrates an application of Algorithm 1.

Example 2 [7; Example 3] *Let*

$$A = \begin{bmatrix} -2 & 2+2\,\mathrm{i} & 1-\mathrm{i} & 0 \\ \dfrac{1-\mathrm{i}}{2} & -3 & 1-\dfrac{\mathrm{i}}{2} & 3+\mathrm{i} \\ 1+\mathrm{i} & 4+2\,\mathrm{i} & -4 & 8+2\,\mathrm{i} \\ 0 & 3-\mathrm{i} & 2-\dfrac{\mathrm{i}}{2} & -5 \end{bmatrix}.$$

Then $\mu_1 = 1$, $\mu_2 = 4$, $\mu_3 = 1$, *and* $\mu_4 = 4$. *Furthermore,*

$$\mathrm{Diag}(\mu)^{1/2} A\, \mathrm{Diag}(\mu)^{-1/2} = \begin{bmatrix} -2 & 1+\mathrm{i} & 1-\mathrm{i} & 0 \\ 1-\mathrm{i} & -3 & 2-\mathrm{i} & 3+\mathrm{i} \\ 1+\mathrm{i} & 2+\mathrm{i} & -4 & 4+\mathrm{i} \\ 0 & 3-\mathrm{i} & 4-\mathrm{i} & -5 \end{bmatrix},$$

which is clearly Hermitian. Its eigenvalues are as follows:

$$-9.1026, \quad -5.75255, \quad 2.62816, \quad -1.77301.$$

Proof Note that

$$B = \begin{bmatrix} -3+\mathrm{i} & 2+2\,\mathrm{i} & 1-\mathrm{i} & 0 \\ \dfrac{1-\mathrm{i}}{2} & -\dfrac{9}{2} & 1-\dfrac{\mathrm{i}}{2} & 3+\mathrm{i} \\ 1+\mathrm{i} & 4+2\,\mathrm{i} & -13+5\,\mathrm{i} & 8+2\,\mathrm{i} \\ 0 & 3-\mathrm{i} & 2-\dfrac{\mathrm{i}}{2} & -5-\dfrac{3\,\mathrm{i}}{2} \end{bmatrix}.$$

And then

$$B^{*\backslash\{\text{last line}\}} = \begin{bmatrix} -3+\mathrm{i} & \dfrac{1-\mathrm{i}}{2} & 1+\mathrm{i} & 0 \\ 2+2\,\mathrm{i} & -\dfrac{9}{2} & 4+2\,\mathrm{i} & 3-\mathrm{i} \\ 1-\mathrm{i} & 1-\dfrac{\mathrm{i}}{2} & -13+5\,\mathrm{i} & 2-\dfrac{\mathrm{i}}{2} \end{bmatrix}.$$

Now, the conclusion follows from Algorithm 1. □

2 Reducing Hermite matrix to tridiagonal one

We have in the last section reduced the Hermitizable matrix to an isospectral Hermitian matrix $\hat{A}$ given in (2). In this section, we further reduce an Hermitian matrix to some isospectral symmetric tridiagonal matrix with nonnegative subdiagonal elements, in terms of Householder transformation. Thus, throughout this section, we fix an Hermitian matrix $A = (a_{ij})_{i,j=1}^{N}$. We are going to use some unitary similar transformation, making the off-tridiagonal elements to be zero. The algorithm is running column by column. Let $A_{k-1} = (a_{ij}^{(k-1)})$

($A_0 := A$) and $b^{(k)}$ be the kth column of A_{k-1} given in Fig. 1. Replacing the first k components of $b^{(k)}$ by zero, we obtain the vector $x^{(k)}$. Next, define $y^{(k)}$ by the following procedure: replacing each component by 0 in $x^{(k)}$, except the element $b^{(k)}_{k+1}$ is replaced by $s_k := \sqrt{x^{(k)H}x^{(k)}}$.

$$b^{(k)} = \begin{bmatrix} b^{(k)}_1 \\ \vdots \\ b^{(k)}_k \\ b^{(k)}_{k+1} \\ b^{(k)}_{k+2} \\ \vdots \\ b^{(k)}_N \end{bmatrix} \longrightarrow x^{(k)} := \begin{bmatrix} 0 \\ \vdots \\ 0 \\ b^{(k)}_{k+1} \\ b^{(k)}_{k+2} \\ \vdots \\ b^{(k)}_N \end{bmatrix} \longrightarrow y^{(k)} := \begin{bmatrix} 0 \\ \vdots \\ 0 \\ s_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Fig. 1 Construction of two vectors: $x^{(k)}$ and $y^{(k)}$

Algorithm 3 At the $k\,(\geqslant 1)$th step, suppose that the matrix obtained after $k-1$ transformations is A_{k-1}. Then, we want to transform $x^{(k)}$ into $y^{(k)} = (s_k\delta_{i,k+1:\,1\leqslant i\leqslant N})$ by a unitary transformation defined by using $x^{(k)}$ and $y^{(k)}$:

$$U_k = I + \frac{uu^H}{\alpha}, \quad u := x^{(k)} - y^{(k)}, \quad \alpha := s_k(b^{(k)}_{k+1} - s_k),$$

or in pointwise form:

$$U_k(i,j) = \delta_{ij} + \frac{1}{s_k(b^{(k)}_{k+1} - s_k)}\,(x^{(k)}_i - s_k\delta_{i,k+1})(\bar{x}^{(k)}_j - s_k\delta_{j,k+1}).$$

Furthermore, we obtain the transformed matrix A_k at step k:

$$A_k = U_k A_{k-1} U_k^H.$$

Note that in the special case that $s_k = 0$ or $s_k = b^{(k)}_{k+1} > 0$, the U_k defined above is meaningless, we can simply ignore this step (or reset $U_k = I$) and jump to the next step at $k+1$. At the last step $k = N-1$ (at most), we obtain the required real symmetric tridiagonal matrix A_{N-1}.

We mention that the unitary matrix $I + (uu^H/\alpha)$ is Hermitian if and only if α is real, or equivalently, so is $b^{(k)}_{k+1}$. If $\alpha \neq 0$, then

$$\begin{aligned} u^H u &= \sum_{j\neq k+1} \bar{x}^{(k)}_j x^{(k)}_j + (\bar{x}^{(k)}_{k+1} - s_k)(x^{(k)}_{k+1} - s_k) \\ &= 2s_k^2 - s_k(x^{(k)}_{k+1} + \bar{x}^{(k)}_{k+1}) \\ &= 2s_k^2 - s_k(b^{(k)}_{k+1} + \bar{b}^{(k)}_{k+1}) \\ &= -(\alpha + \bar{\alpha}). \end{aligned}$$

Hence,

$$U_k^H U_k = I + \frac{uu^H}{\bar{\alpha}\alpha}\,(\alpha + \bar{\alpha} + u^H u) = I.$$

Equivalently, $U_k U_k^H = I$ in view of the operation $H : A \to A^H$. Thus, U_k is surely unitary.

The following algorithm is for computing the maximal eigenvector $g_{\max}(A)$, which can be run in parallel to Algorithm 3 above.

Algorithm 4 Starting at $V_1 = U_1^H$, update V_j step by step in parallel to Algorithm 3:

$$V_k = V_{k-1} U_k^H, \quad k = 2, \ldots, N-1.$$

Denote by $(\lambda_{\max}(T), g_{\max}(T))$ the maximal eigenpair of $T := A_{N-1}$. Then the maximal eigenpair of A can be expressed by

$$(\lambda_{\max}(A),\ g_{\max}(A)) = (\lambda_{\max}(T), V_{N-1} g_{\max}(T)).$$

Similarly, one can compute the other eigenpairs of A using the ones of T with the same transform V_{N-1}.

Alternatively, $g_{\max}(A) =: g^{(0)}$ can be obtained by the following procedure:

$$g^{(k-1)} = U_k^H g^{(k)}, \quad k = N-1, \ldots, 1.$$

The first method in Algorithm 4 does not need to store, step by step, the whole sequence $\{V_j\}_{j=1}^{N-1}$, but it requires about $N(N+1)(N+\frac{1}{2})$ times of multiplications. Here is a careful analysis on the complexity of $g_{\max}(A)$ of the method. First, we compute the complexity of V_k. Note that $U_k = I + (uu^H/\alpha)$, we have $U_k^H = I + (uu^H/\bar{\alpha})$. As usual, we count only the multiplications. Since at the kth step, the first k components of u are zero, and so are $u/\bar{\alpha}$ and u^H, it follows that

$$z := \frac{u}{\bar{\alpha}} \quad \text{requires } N-k \text{ times of multiplications,}$$

$$Z := V_{k-1} z \quad \text{requires } N(N-k) \text{ times of multiplications,}$$

$$Zu^H \quad \text{requires } N(N-k) \text{ times of multiplications.}$$

The last step needs a little explanation. As a product of the column vector Z and the row vector u^H, one often requires N^2 times of multiplications. Here the first k columns of the resulting matrix are zero and so can be ignored, since the first k components of u^H are zero. Hence the total multiplications are reduced to be $N(N-k)$ as given above. Thus, it means that $V_k = V_{k-1} U_k^H$ requires $(2N+1)(N-k)$ times of multiplications. Next, for k varying from 1 to $N-1$, we obtain V_{N-1}, which requires

$$\sum_{k=1}^{N-1} (2N+1)(N-k) = (2N+1)\Big[N(N-1) - \frac{N(N-1)}{2}\Big] = N(N-1)\Big(N + \frac{1}{2}\Big)$$

times of multiplications. Finally, for $g_{\max}(A) = V_{N-1}g_{\max}(T)$, it requires additionally N^2 times of multiplications. Therefore, for $g_{\max}(A)$, it requires totally

$$N(N-1)\Big(N+\frac{1}{2}\Big)+N^2 = N\Big(N^2+\frac{N}{2}-\frac{1}{2}\Big) = N(N+1)\Big(N-\frac{1}{2}\Big)$$

times of multiplications.

Comparing the first method just discussed above, the second one (given at the end of the algorithm) goes on the opposite direction: we have to store the sequence $\{x^{(j)}\}$ (or plus $\{s_j = \|x^{(j)}\|\}$) which generates the sequence $\{U_j\}$, but the iterative computations require only $N(N-1)/2$ multiplications. Thus, the second method is more effective than the first one, at least for large matrices.

We will come back to this topic in Algorithm 7 below.

The Householder transformation goes back to [10]. The representation here is taken from Wang [17] which is based on [16]. Since $s_k = y^{(k)H}u$, the expression of U_k here fits [8; p. 2375, the formula right above part III].

We now illustrate the algorithm by some examples.

Example 5 (Continued) *Let A be the Hermitian matrix given at the end of Example* 2. *Then we have*

$$A_3 = \begin{bmatrix} -2 & 2 & 0 & 0 \\ 2 & -\frac{5}{2} & \frac{\sqrt{67}}{2} & 0 \\ 0 & \frac{\sqrt{67}}{2} & -\frac{265}{134} & \frac{2\sqrt{7717}}{67} \\ 0 & 0 & \frac{2\sqrt{7717}}{67} & -\frac{504}{67} \end{bmatrix}.$$

Notice that the sub-diagonal elements of the symmetric tridiagonal matrix A_3 are positive. We have thus reduced the computation of the maximal eigenpair of Hermitian A to the real tridiagonal one A_3. Furthermore, we have

$$\lambda_{\max}(A) = 2.62816,$$

$$g_{\max}(A) = (.51569 + .137426\,\mathrm{i},\ 1.07178 + .0943814\,\mathrm{i},\ .969716 + .439587\,\mathrm{i},\ 1)^*.$$

Proof At first step, we have

$$x^{(1)} = (0,\ 1-\mathrm{i},\ 1+\mathrm{i},\ 0)^*,$$

$$V_1 = U_1^H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1-\mathrm{i}}{2} & \frac{1+\mathrm{i}}{2} & 0 \\ 0 & \frac{1+\mathrm{i}}{2} & \frac{1-\mathrm{i}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$A_1 = \begin{bmatrix} -2 & 2 & 0 & 0 \\ 2 & -\frac{5}{2} & 2+\frac{\mathrm{i}}{2} & \frac{7+\mathrm{i}}{2} \\ 0 & 2-\frac{\mathrm{i}}{2} & -\frac{9}{2} & \frac{7+3\,\mathrm{i}}{2} \\ 0 & \frac{7-\mathrm{i}}{2} & \frac{7-3\,\mathrm{i}}{2} & -5 \end{bmatrix}.$$

At the second step, we have

$$x^{(2)} = \left(0,\, 0,\, 2-\frac{\mathrm{i}}{2},\, \frac{7-\mathrm{i}}{2}\right)^*,$$

$$V_2 = V_1 U_2^H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & .5-.5\,\mathrm{i} & .305424+.183254\,\mathrm{i} & .106015+.601577\,\mathrm{i} \\ 0 & .5+.5\,\mathrm{i} & .183254-.305424\,\mathrm{i} & .601577-.106015\,\mathrm{i} \\ 0 & 0 & .855186-.122169\,\mathrm{i} & -.38067-.329881\,\mathrm{i} \end{bmatrix},$$

$$A_2 = \begin{bmatrix} -2 & 2 & 0 & 0 \\ 2 & -\frac{5}{2} & \frac{\sqrt{67}}{2} & 0 \\ 0 & \frac{\sqrt{67}}{2} & -\frac{265}{134} & \frac{(81-34\,\mathrm{i})(\sqrt{67}-4+\mathrm{i})^2}{134\,(21-2\sqrt{67}\,)} \\ 0 & 0 & \frac{(81+34\,\mathrm{i})(\sqrt{67}-4-\mathrm{i})^2}{134\,(21-2\sqrt{67}\,)} & -\frac{504}{67} \end{bmatrix}.$$

Finally, at the third step, we have

$$x^{(3)} = \left(0,\, 0,\, 0,\, \frac{(81+34\,\mathrm{i})(\sqrt{67}-4-\mathrm{i})^2}{134(21-2\sqrt{67}\,)}\right)^*,$$

$$V_3 = V_2 U_3^H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & .5-.5\,\mathrm{i} & .305424+.183254\,\mathrm{i} & .148807+.592445\,\mathrm{i} \\ 0 & .5+.5\,\mathrm{i} & .183254-.305424\,\mathrm{i} & .592445-.148807\,\mathrm{i} \\ 0 & 0 & .855186-.122169\,\mathrm{i} & -.403308-.301785\,\mathrm{i} \end{bmatrix},$$

and then A_3 given in the example.

Because

$$\lambda_{\max}(A_3) = 2.62816, \quad g_{\max}(A_3) = (1.60558,\ 3.71545,\ 3.87088,\ 1)^*,$$

we have $\lambda_{\max}(A) = 2.62816$ and

$$\begin{aligned} g_{\max}(A) &= V_3\, g_{\max}(A_3) \\ &= (.51569+.137426\,\mathrm{i},\ 1.07178+.0943814\,\mathrm{i},\ .969716+.439587\,\mathrm{i},\ 1)^*. \end{aligned}$$

The conclusion is checked by

$$\frac{Ag_{\max}(A)}{\lambda_{\max}(A)} = g_{\max}(A). \qquad \square$$

For the computation of the maximal eigenpair of tridiagonal matrix, refer to [7], and see Section 4 of the paper for analytic details. The next example shows a blocking phenomenon which seems not treated before carefully.

Example 6 *Let*

$$A = \begin{bmatrix} \frac{732}{289} & -\frac{81-27\,\mathrm{i}}{289} & -\frac{50+50\,\mathrm{i}}{289} & -\frac{70+60\,\mathrm{i}}{289} \\ -\frac{81+27\,\mathrm{i}}{289} & \frac{813}{289} & -\frac{20+40\,\mathrm{i}}{289} & -\frac{30+50\,\mathrm{i}}{289} \\ -\frac{50-50\,\mathrm{i}}{289} & -\frac{20-40\,\mathrm{i}}{289} & \frac{648}{289} & \frac{91-7\,\mathrm{i}}{289} \\ -\frac{70-60\,\mathrm{i}}{289} & -\frac{30-50\,\mathrm{i}}{289} & \frac{91+7\,\mathrm{i}}{289} & \frac{41}{17} \end{bmatrix}.$$

Then the deduced tridiagonal matrix is divisible:

$$A_3 = \begin{bmatrix} \frac{732}{289} & \frac{3\sqrt{2310}}{289} & & 0 \\ \frac{3\sqrt{2310}}{289} & \frac{713}{289} & & \\ & & \frac{64}{27} & \frac{\sqrt{170}}{27} \\ 0 & & \frac{\sqrt{170}}{27} & \frac{71}{27} \end{bmatrix}.$$

Proof At the first step, we have

$$x^{(1)} = \Big(0, -\frac{81+27\,\mathrm{i}}{289}, -\frac{50-50\mathrm{i}}{289}, -\frac{70-60\,\mathrm{i}}{289}\Big)^*,$$

$V_1 = U_1^H$

$$= -\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & .561769 + .187256\,\mathrm{i} & .254965 + .418919\,\mathrm{i} & .373346 + .519098\,\mathrm{i} \\ 0 & .346771 - .346771\,\mathrm{i} & -.84819 + .018202\,\mathrm{i} & .199173 + .00848164\,\mathrm{i} \\ 0 & .485479 - .416125\,\mathrm{i} & .195533 + .0388436\,\mathrm{i} & -.741923 + .0309434\,\mathrm{i} \end{bmatrix},$$

$$A_1 = \begin{bmatrix} \frac{732}{289} & \frac{3\sqrt{2310}}{289} & & 0 \\ \frac{3\sqrt{2310}}{289} & \frac{713}{289} & & \\ & & \frac{64}{27} & \frac{13-\mathrm{i}}{27} \\ 0 & & \frac{13+\mathrm{i}}{27} & \frac{71}{27} \end{bmatrix}.$$

The second step can be ignored since for which we have $x^{(2)} = 0$. Then we have $U_2 = I$ and so $V_2 = V_1$. We now go to the third step:

$$x^{(3)} = \Big(0,\, 0,\, 0,\, \frac{13+\mathrm{i}}{27}\Big)^*,$$

$$V_3 = V_2 U_3^H$$
$$= -\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & .561769 + .187256\,\mathrm{i} & .254965 + .418919\,\mathrm{i} & .332433 + .546204\,\mathrm{i} \\ 0 & .346771 - .346771\,\mathrm{i} & -.84819 + .018202\,\mathrm{i} & .197936 + .0237325\,\mathrm{i} \\ 0 & .485479 - .416125\,\mathrm{i} & .195533 + .0388436\,\mathrm{i} & -.742111 - .0260506\,\mathrm{i} \end{bmatrix},$$

and A_3 given in the example. Clearly, the matrix A_3 can be reduced to two 2×2 matrices and so it has two repeated pairs of eigenvalues $\{3, 2\}$. The reason is as follows. First, for an irreducible (or unreduced) tridiagonal matrix, its eigenvalues are distinct. This classical result is included in many textbooks, see, for instance, [1; p. 97, Theorem 3.3], [11; p. 36, Theorem 2.2], [14; p. 134, Lemma 7.7.1]), or [18; pp. 300–302]. In this case, the block decomposition for the matrix can be ignored. Hence, A_3 should have multiple eigenvalues and the multiplicity should be less than or equal to 2. Otherwise, there would have more blocks, not only two. If there are three distinct eigenvalues, then there would have two submatrices with size 3×3 and 1×1, respectively. Hence, we are not in this situation. The conclusion can be easily checked by computing the eigenvalues of these two 2×2 submatrices separately.

To compute the maximal eigenvector of A in the above example, let

$$A_3^{(1)} = \frac{1}{289}\begin{bmatrix} 732 & 3\sqrt{2310} \\ 3\sqrt{2310} & 713 \end{bmatrix}, \quad A_3^{(2)} = \frac{1}{27}\begin{bmatrix} 64 & \sqrt{170} \\ \sqrt{170} & 71 \end{bmatrix}.$$

Then, with the same maximal eigenvalue 3, the maximal eigenvectors for them are

$$g_3^{(1)} = \Big(\frac{1}{3}\sqrt{\frac{154}{15}},\, 1\Big)^*, \quad g_3^{(2)} = \Big(\sqrt{\frac{10}{17}},\, 1\Big)^*,$$

respectively. Thus, the matrix A_3 has the maximal eigenvalue 3 with multiplicity 2 and independent eigenvectors as follows:

$$g^{(1)} = \Big(\frac{1}{3}\sqrt{\frac{154}{15}},\, 1,\, 0,\, 0\Big)^*, \quad g^{(2)} = \Big(0,\, 0,\, \sqrt{\frac{10}{17}},\, 1\Big)^*.$$

By Algorithm 4, the similar assertion holds for the original A with independent eigenvectors

$$V_3 g^{(1)} = (1.06805,\, -.561769 - .187256\,\mathrm{i},\, -.346771 + .346771\,\mathrm{i},\\ -.485479 + .416125\,\mathrm{i})^*,$$

$$V_3 g^{(2)} = (0,\ -.527982 - .8675\,\mathrm{i},\ .452596 - .0376928\,\mathrm{i},\ .592145 - .00374103\,\mathrm{i})^*,$$

respectively. Clearly, these two vectors are linear independent. Let us mention that each N-dimensional Hermite matrix has N linear independent eigenvectors, and so does each Hermitizable one. □

It is the position to come back to the computation of the maximal eigenvector. From the last two examples, we have seen that the computation of the sequence $\{V_j\}$ costs heavier work (actually has a higher complexity). We now show that in some cases (the matrix has a smaller size or is rather sparse, for instance), it is possible to use directly the shift inverse iteration.

Algorithm 7 Let $A = (a_{ij})_{i,j=1}^N$ be a given Hermitizable matrix. Set $z = \lambda_{\max}(A)+\varepsilon$ with small $\varepsilon > 0$ (say 10^{-8} for instance) and choose a suitable vector w_0. For a given vector w (may have subscript), here we fix the normalization of w in terms of its first $w(1)$ or last components $w(N)$:

$$v = \frac{w}{w(1)} \quad \text{or} \quad \frac{w}{w(N)}.$$

Now, for given $v := v_{k-1}$, the shift inverse iteration goes as follows. Let $w = w_k$ solve the equation

$$(zI - A)w = v.$$

Then define v_k as the normalization of w_k just defined. Continue the iterations until the solutions become the same up to six digits of precision.

The reason we add a small constant ε in Algorithm 7 is to avoid the singularity of the matrix $zI - A$. The main price we have to pay is for the linear equation involved in the algorithm. Thus, once there is an effective algorithm for solving the equation (when A is symmetric, or sparse, or having smaller size, for instance), the algorithm should work well.

Example 8 (Continued) We now apply Algorithm 7 to Example 6. Set $z = 3 + 10^{-8}$. We have chosen three initials for v_0:

(1) $(1, 1, 0, 0)^*$;

(2) $(0, 0, 1, 1)^*$;

(3) $(1, 1, 1, 1)^*$.

Since the eigenspace of the maximal eigenvalue 3 has dimension 2, there is some freedom for the solution of the algorithm. After one iteration, the outputs of the vector v_1 in the corresponding cases are as follows:

(1) $(1,\ 1.73539 - 1.01172\,\mathrm{i},\ -.44239 + 1.3965\,\mathrm{i},\ -.714757 + 1.77121\,\mathrm{i})^*$;

(2) $(-.588235 - .504202\,\mathrm{i},\ -.252101 - .420168\,\mathrm{i},\ .764706 - .0588235\,\mathrm{i},\ 1)^*$;

(3) $(-.577029 - .140796\,\mathrm{i},\ -.0872077 - .970879\,\mathrm{i},\ .764706 - .0588235\,\mathrm{i},\ 1)^*$.

It is checked by the eigenequation $Av_1/3 = v_1$ so that these three solutions are all the eigenvectors of A. The first two are quite different from those produced by V_3 as seen above. Each pair of them are linear independent.

Originally, the next example was designed as a complex matrix. Since it is too big for the output, we choose a real one. However, the story is very much similar.

Example 9 *Due to the multiplicity of the eigenvalues, the tridiagonalized matrix of the next symmetric one*

$$A = \begin{bmatrix} \frac{8074}{2601} & \frac{271}{5202} & \frac{373}{1734} & \frac{746}{2601} & \frac{1355}{5202} & \frac{169}{867} & \frac{676}{2601} & -\frac{959}{5202} \\ \frac{271}{5202} & \frac{31483}{10404} & \frac{373}{3468} & \frac{373}{2601} & \frac{1355}{10404} & \frac{169}{1734} & \frac{338}{2601} & -\frac{959}{10404} \\ \frac{373}{1734} & \frac{373}{3468} & \frac{2787}{1156} & \frac{475}{867} & \frac{1865}{3468} & \frac{271}{578} & \frac{542}{867} & -\frac{245}{3468} \\ \frac{746}{2601} & \frac{373}{2601} & \frac{475}{867} & \frac{7102}{2601} & \frac{1865}{2601} & \frac{542}{867} & \frac{2168}{2601} & -\frac{245}{2601} \\ \frac{1355}{5202} & \frac{1355}{10404} & \frac{1865}{3468} & \frac{1865}{2601} & \frac{37987}{10404} & \frac{845}{1734} & \frac{1690}{2601} & -\frac{4795}{10404} \\ \frac{169}{867} & \frac{169}{1734} & \frac{271}{578} & \frac{542}{867} & \frac{845}{1734} & \frac{1223}{289} & \frac{268}{867} & -\frac{1673}{1734} \\ \frac{676}{2601} & \frac{338}{2601} & \frac{542}{867} & \frac{2168}{2601} & \frac{1690}{2601} & \frac{268}{867} & \frac{11476}{2601} & -\frac{3346}{2601} \\ -\frac{959}{5202} & -\frac{959}{10404} & -\frac{245}{3468} & -\frac{245}{2601} & -\frac{4795}{10404} & -\frac{1673}{1734} & -\frac{3346}{2601} & \frac{46123}{10404} \end{bmatrix}$$

has a block decomposition with three blocks. Moreover, the distinguished eigenvalues are all included in the largest block of the decomposition.

Proof At Steps 1–3, the outputs are standard. Here is the output in Step 3.

$$x^{(3)} = (0,\ 0,\ 0,\ .122312,\ -.565458,\ -.20606,\ -.274746,\ -.781817)^*,$$

$$A_3 = \begin{bmatrix} 3.10419 & .581915 & 0 & 0 & & & & \\ .581915 & 6.43398 & 1.2267 & 0 & & & 0 & \\ 0 & 1.2267 & 3.6022 & 1.03145 & & & & \\ 0 & 0 & 1.03145 & 2.85964 & & & & \\ & & & & 2.72146 & -.20848 & -.277973 & -.283237 \\ & & & & -.20848 & 3.48396 & -.688054 & -.211994 \\ & 0 & & & -.277973 & -.688054 & 3.08259 & -.282659 \\ & & & & -.283237 & -.211994 & -.282659 & 2.71199 \end{bmatrix}.$$

Note that here and in what follows, once the output is less than 10^{-14}, we simply reset it as zero, because it usually comes from the computational errors. Step 4 can be omitted since $x^{(4)} = 0$. Here is the output of Step 5:

$$x^{(5)} = (0,\ 0,\ 0,\ 0,\ 0,\ -.20848,\ -.277973,\ -.283237)^*,$$

$$A_5 = \begin{bmatrix} 3.10419 & .581915 & & & & & & \\ .581915 & 6.43398 & 1.2267 & & & & 0 & \\ & 1.2267 & 3.6022 & 1.03145 & & & & \\ & & 1.03145 & 2.85964 & 0 & & & \\ & & & & 0 & 2.72146 & .448281 & \\ & & & & & .448281 & 2.27854 & 0 \\ & 0 & & & & & 0 & 3.88097 & .323827 \\ & & & & & & & .323827 & 3.11903 \end{bmatrix}.$$

We do not need to move further since A_5 has already arrived at the required tridiagonal form. Actually, it reduces into three submatrices:

$$A_5^{(1)} = \begin{bmatrix} 3.10419 & .581915 & 0 & 0 \\ .581915 & 6.43398 & 1.2267 & 0 \\ 0 & 1.2267 & 3.6022 & 1.03145 \\ 0 & 0 & 1.03145 & 2.85964 \end{bmatrix},$$

$$A_5^{(2)} = \begin{bmatrix} 2.72146 & .448281 \\ .448281 & 2.27854 \end{bmatrix}, \quad A_5^{(3)} = \begin{bmatrix} 3.88097 & .323827 \\ .323827 & 3.11903 \end{bmatrix}.$$

Clearly, the eigenvalues of $A_5^{(2)}$ and $A_5^{(3)}$ are all included in those of $A_5^{(1)}$. Thus, in the case we are only interested in distinct eigenvalues, ignoring their multiplicity, it suffices to study the eigenvalues of the largest submatrix $A_5^{(1)}$. More explicitly, the eigenvalues of $A_5^{(1)}$, $A_5^{(2)}$, and $A_5^{(3)}$ are $\{7, 4, 3, 2\}$, $\{3, 2\}$, and $\{4, 3\}$, respectively. Corresponding to each distinct eigenvalue, one eigenvector can be deduced from those of the largest $A_5^{(1)}$. Note that the eigenvalue 3 has multiplicity 3, this is the reason why we have 3 submatrices and also the reason we can omit several steps in the computation. □

In the past two examples, we have seen that the Householder transformation transfers an irreducible matrix into a reducible tridiagonal one. This is reasonable as explained before, due to the multiplicity of eigenvalues. Note that the Householder transformation is a unitary one and hence keeps the Hermitizability (symmetrizability), and furthermore the spectrum. By [5; Corollary 6], for the Hermitizability of a tridiagonal matrix, the matrix can be reducible. This property is remarkable since then for which, the Perron–Frobenius theorem may not be true. Therefore, the same theorem may also not true for Hermitian matrix.

To conclude this section, we mention that in practice, for improving the efficiency of computations, one may adopt some artistic design for Algorithms 3 and 4. See for instance [15; pp. 106–108]. See also [15; pp. 582, 583] in particular, for some analysis and algorithms on Householder transformation. Besides, refer to [13] for concurrent algorithms.

3 Sub-maximal eigenpair

In this section, we introduce two approaches to compute the submaximal (or its next) eigenpair. For the first one, we need the following result.

Lemma 10 *Let A be Hermitian. Denote by (λ_0, g_0) its maximal eigenpair $(\lambda_{\max}, g_{\max})$ with $\lambda_0 > 0$ and $g_0^H g_0 = 1$. Define*

$$A_1 = (I - g_0 g_0^H)A. \tag{3}$$

Then A_1 is also Hermitian.

Proof We need to prove that $A_1 = A_1^H$. Equivalently,

$$g_0 g_0^H A = A^H g_0 g_0^H.$$

First, we have

$$\begin{aligned} g_0 g_0^H A &= g_0 g_0^H A^H \quad (\text{since } A = A^H) \\ &= \bar{\lambda}_0 g_0 g_0^H \quad (\text{since } Ag_0 = \lambda_0 g_0 \Longrightarrow g_0^H A^H = \bar{\lambda}_0 g_0^H) \\ &= \lambda_0 g_0 g_0^H \quad (\text{since the spectrum of } A \text{ is real}). \end{aligned}$$

Next,

$$\begin{aligned} A^H g_0 g_0^H &= A g_0 g_0^H \quad (\text{since } A = A^H) \\ &= \lambda_0 g_0 g_0^H \quad (\text{since } Ag_0 = \lambda_0 g_0). \end{aligned}$$

We have thus proved the required assertion. □

We now consider a simple example. Let

$$Q = \begin{bmatrix} -1 & 1 & & & \\ 2 & -3 & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & 1 \\ & & & 2 & -3 \end{bmatrix} \in \mathbb{R}^8 \times \mathbb{R}^8.$$

Since the symmetrizing measure of Q is $\mu_k = 2^{-k+1}$, $k = 1, \ldots, 8$, we have

$$Q^{\text{sys}} = \text{Diag}(\mu)^{1/2} Q \,\text{Diag}(\mu)^{-1/2} = \begin{bmatrix} -1 & \sqrt{2} & & & \\ \sqrt{2} & -3 & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & \sqrt{2} \\ & & & \sqrt{2} & -3 \end{bmatrix} \in \mathbb{R}^8 \times \mathbb{R}^8.$$

Example 11 Define $A = Q^{\rm sys} + 3\,I$. Then the eigenvalues of A are as follows:

$$\begin{aligned}\lambda_{\max} = {} & 2.99799,\ -2.63352,\ 2.50514,\ -2.07511,\\ & 1.79552,\ -1.22867,\ .847221,\ -.208572,\end{aligned}$$

and the maximal eigenvector g_0 is as follows:

$$\begin{aligned}g_{\max} = {} & (.715152,\ .504673,\ .354704,\ .247264,\\ & .169471,\ .111997,\ .0679521,\ .0320544)^*.\end{aligned}$$

We are now going to study the sub-maximal eigenpair of A. Actually, there are at least two approaches to do the work: the deflation technique and the optimal search approach, to be studied below.

Deflation technique

First, we use the known deflation technique introduced in [12; Theorem 2.2]. That is, studying the matrix A_1 defined by (3). With g_0 just obtained above, by (3), the matrix A_1 takes the following form:

$$\begin{bmatrix}
.466701 & .332185 & -.760492 & -.530139 & -.363349 & -.240124 & -.145691 & -.0687252\\
.332185 & -.763572 & .877545 & -.374111 & -.25641 & -.169452 & -.102812 & -.0484984\\
-.760492 & .877545 & -.377192 & 1.15127 & -.180215 & -.119098 & -.0722602 & -.0340866\\
-.530139 & -.374111 & 1.15127 & -.183296 & 1.28859 & -.083023 & -.0503726 & -.0237618\\
-.363349 & -.25641 & -.180215 & 1.28859 & -.0861034 & 1.35731 & -.0345246 & -.0162859\\
-.240124 & -.169452 & -.119098 & -.083023 & 1.35731 & -.0376049 & 1.3914 & -.0107628\\
-.145691 & -.102812 & -.0722602 & -.0503726 & -.0345246 & 1.3914 & -.0138432 & 1.40768\\
-.0687252 & -.0484984 & -.0340866 & -.0237618 & -.0162859 & -.0107628 & 1.40768 & -.00308039
\end{bmatrix},$$

which is symmetric by Lemma 10. The eigenvalues of A_1 are as follows:

$$\begin{aligned}& -2.63352,\ 2.50514,\ -2.07511,\ 1.79552,\ -1.22867,\\ & 0.847221,\ -0.208572,\ -1.33596\cdot 10^{-15}.\end{aligned}$$

Hence, $\lambda_{\max}(A_1) = 2.50514$. In other words, the submaximal eigenvalue of A now becomes the maximal one of A_1 (cf. [12]). We remark that at this point, the condition $\lambda_{\max}(A_1) > 0$ is necessary. Otherwise, a shift of A is required.

By Lemma 10, we now come back to the starting point treated in Section 2. By using the Householder transformation, it follows that A_1 is isospectral to the following tridiagonal matrix:

$$T = \begin{bmatrix}
.466701 & 1.08875 & & & & & & \\
1.08875 & .852645 & 1.49569 & & & & & \\
 & 1.49569 & -.727517 & 1.21056 & & & & \\
 & & 1.21056 & .38967 & 1.44937 & & & \\
 & & & 1.44937 & -.592352 & 1.12888 & & \\
 & & & & 1.12888 & -.147293 & 1.25861 & \\
 & & & & & 1.25861 & -1.0249 & .333798\\
 & & & & & & .333798 & -.214945
\end{bmatrix}.$$

We have thus reduced our problem to the setup treated in [7].

From the discussion above, it is clear that the Householder transformation can also be used to study the submaximal eigenpair, provided the given matrix is not sparse. However, at the moment, we are not in such a case. Hence, it is too heavy to use the transformation again for computing the submaximal eigenpair, since then the sparse property of A is lost. Fortunately, we have a different approach given in the next part.

Optimal search approach

Actually, in the present situation, there is an easier way to handle with this problem, which uses the main advantage of the (symmetric) tridiagonal matrix. Following [1; p. 146], we adopt the method of bisection. To fix the notation, as before, let $A \sim (a_k, c_k, b_k)_{k=1}^N$. Throughout the remainder of this section, assume that A is irreducible (i.e., $a_{k+1}b_k > 0$).

Method of bisection

To state the method, we need some notation. Define

$$s_1(\alpha) = c_1 - \alpha,$$

$$s_2(\alpha) = \begin{cases} c_2 - \alpha - \dfrac{b_1^2}{s_1(\alpha)}, & \text{if } s_1(\alpha) \neq 0, \\ -\varepsilon \ (0 < \varepsilon \ll 1), & \text{if } s_1(\alpha) = 0, \end{cases}$$

$$s_k(\alpha) = \begin{cases} c_k - \alpha - \dfrac{b_{k-1}^2}{s_{k-1}(\alpha)}, & \text{if } s_{k-1}(\alpha)s_{k-2}(\alpha) \neq 0, \\ c_k - \alpha, & \text{if } s_{k-2}(\alpha) = 0, \\ -\varepsilon \ (0 < \varepsilon \ll 1), & \text{if } s_{k-1}(\alpha) = 0. \end{cases}$$

Next, define

$$\gamma(\alpha) = \#\{s_k(\alpha) \geqslant 0,\ k = 1, \ldots, N\}.$$

Then, by [1; p. 142, (2.16) and Theorem 2.1; p. 146, (2.23) and the remark below it], we have the following result.

Lemma 12 *The number of eigenvalues of A on $[\alpha, \infty)$ equals $\gamma(\alpha)$.*

The application of the bisection method, given in the remainder of this section, is mainly based on Lemma 12. To state an algorithm, we need a little preparation. Define

$$\eta = \max\Big\{ \min_{1 \leqslant i \leqslant N} (c_i - a_i - b_i),\ \min_{1 \leqslant i \leqslant N} (c_i - a_{i+1} - b_{i-1}) \Big\}, \quad b_1 := 0,\ a_{N+1} := 0.$$

In the present setup, there are N distinct eigenvalues, listed as

$$\lambda_1 > \cdots > \lambda_N.$$

By Gershgorin Circle Theorem for the spectral radius, we have $\lambda_N \geqslant \eta$. To study the top eigenpairs of A, we start at the top eigenvalues. The computation goes one by one of the eigenvalues. The computation of (λ_1, g_1) was done in [7]. Starting from which, we study the subsequent top eigenvalues. For instance, based on λ_1 and η, we can compute λ_2 by using the bisection method. The algorithm given below consists of two parts. The first one is constructing an initial interval (β_2, β_1) for the bisection method. The second one is the standard sequent search of the method.

Algorithm 13 Let η be defined as above. Suppose that λ_{k-1} $(k \geqslant 2)$ is given and we are going to compute λ_k.

Step 1 Define

$$\xi_0 = \lambda_{k-1}, \quad \xi_1 = \frac{1}{N-k+1}\,[(N-k)\xi_0 + \eta],$$
$$\xi_m = \max\{3\xi_{m-1} - 2\xi_{m-2},\, \eta\}, \quad m \geqslant 2.$$

Compute $\gamma(\xi_m)$ successively until for the first $m = m_0$ such that $\gamma\big(\xi_{m_0}\big) \geqslant k$. Then take $[\beta_2, \beta_1] = [\xi_{m_0},\, \xi_{m_0-1}]$ as the initial interval for using the method of bisection.

Step 2 Set $z = (\beta_2 + \beta_1)/2$ and compute $\gamma(z)$. We update the test interval by making the following changes, step by step. In details, if $\gamma(z) \geqslant k$, then replace β_2 by z; and otherwise replace β_1 by z:

$\gamma(z)$	Change
$\geqslant k$	$\beta_2 \to z$
$= k-1$	$\beta_1 \to z$

Repeating the recursive procedure until the outputs are the same up to six digits of precision, and γ takes value k at the final tested point.

Because

$$2^{-24} \approx 5.96046 \times 10^{-8},$$

for the six/seven digits of precision, the method of bisection requires about 24 times of steps (tests), independent of the matrix size N. Of course, at each step, in computing $\gamma(\alpha)$, it requires N times of computations. Hence, the complexity for using the method of bisection is $O(N)$.

The second (submaximal) eigenpair

(a) We now return to the matrix A given by Example 11. Keeping $\lambda_{\max}(A) = 2.99799$ in mind and using the method of bisection, after 16 steps, the outputs are the same up to six digits of precision, we obtain the submaximal eigenvalue ≈ 2.50514.

(b) Next, we compute the submaximal eigenvector using the shift inverse iteration. Recall that in the present tridiagonal situation, we have the more or

less explicit Thomas algorithm for solving the required linear equation (cf. [7]). By (a), we can fix the shift to be $z = 2.50514 - \varepsilon$ ($\varepsilon = 10^{-8}$ for instance). Here and in what follows, the small modification ε is for avoiding the degeneration in using the shift inverse iteration, and also for avoiding the next eigenvalue. Note that the maximal eigenvector of A is

$$g_{\max} = (22.3106,\ 15.7443,\ 11.0657,\ 7.71389,\ 5.28698,\ 3.49398,\ 2.1199,\ 1)^*.$$

Its normalized vector is

$$\begin{aligned} g = (&.715152,\ .504673,\ .354704,\ .247264,\\ &.169471,\ .111997,\ .0679521,\ .0320544)^*. \end{aligned}$$

Our initial v_0 is chosen to be

$$\begin{aligned} v_0 &:= g - \frac{\mathbb{1}}{g^*\mathbb{1}}\ (\text{then } v_0 \in \mathrm{Span}(g)^\perp)\\ &= (.261281,\ .0508014,\ -.0991675,\ -.206607,\\ &\quad -.284401,\ -.341874,\ -.385919,\ -.421817)^*. \end{aligned}$$

After one iteration, the output of the computation is as follows:

$$\begin{aligned} g_2 := (&.341037,\ .121815,\ -.125255,\ -.343691,\\ &-.483561,\ -.512889,\ -.424972,\ -.239907)^*. \end{aligned}$$

This is checked by using the second iteration, its output is exactly the same. A simpler way to check this conclusion is simply using the eigenequation: $Ag_2 = \lambda_2 g_2$, where $\lambda_2 = 2.50514$. Therefore, g_2 can be regarded as the submaximal eigenvector as we required.

The third (next to the submaximal) eigenpair

To conclude this section, we remark that the same method can also be used to compute the subsequent eigenpairs. Here we mention shortly the computation for the next to the submaximal eigenpair for the same example as above. Rewrite $\lambda_1 = \lambda_{\max}(A)$. Then we have known the eigenpairs (λ_1, g_1) and (λ_2, g_2). We are now going to compute the next one (λ_3, g_3). By using the method of bisection above, we obtain $\lambda_3 = 1.79552$. Now, to apply the shift inverse iteration, choose shift $z = 1.79552 - \varepsilon$. For the initial vector v_0, we choose the form

$$v_0 = (1, x, x, x, y, y, y,\ 1.1)^*$$

with x and y determined by the conditions $v_0 \perp g_1$ and $v_0 \perp g_2$:

$$x = -.753323, \quad y = .238242.$$

Here, the components 1 and 1.1 in v_0 are chosen randomly. If these random numbers are replaced by x and y, respectively, then the homogeneous equations

have only trivial solution $x = y = 0$. In one step (iteration), we obtain the required eigenvector

$$g_3 = (-.350163,\ .0506296,\ .414444,\ .475559,\\ .189337,\ -.235171,\ -.487917,\ -.3843)^*.$$

To conclude this section, we mention that Algorithm 13 can be naturally extended to concurrent computing, simply replacing the bisection method by the equisection one, which is a generalization of the method of bisection in the optimization theory.

4 Remarks and proofs

This section is mainly devoted to the analytical aspect of the algorithms introduced in the previous sections. Besides, it also provides a detailed exploration on [5; Theorem 24]. At the end of this section, we present a simplified proof for a key result concerning with the isospectral property of an Hermitizable tridiagonal matrix and a birth-death one (see Theorem 14 below).

Remark on Algorithm 1

Before moving to the main text, let us make a remark on Algorithm 1 (1). Let $A = (a_{ij})$ be a given complex matrix. By (1), we have

$$\sum_i \mu_i a_{ij} = \mu_j \sum_i \bar{a}_{ji}.$$

Equivalently, $(\mu A)(j) = (\mu \mathrm{Diag}(\bar{A}\mathbb{1}))(j)$. From this, we obtain $\mu B = 0$ as stated in the algorithm.

Proof of Algorithm 4

Let $g^{(k)} \neq 0$ be the eigenvector of A_k $(0 \leqslant k < N)$ with $A_0 = A$, corresponding to a fixed eigenvalue λ:

$$A_k g^{(k)} = \lambda g^{(k)} \Longleftrightarrow (U_k A_{k-1} U_k^H) g^{(k)} = \lambda g^{(k)} \Longleftrightarrow A_{k-1}(U_k^H g^{(k)}) = \lambda (U_k^H g^{(k)}).$$

This implies that $g^{(k-1)} = U_k^H g^{(k)}$. Then the last assertion in the algorithm follows. Furthermore,

$$g^{(0)} = U_1^H g^{(1)} = U_1^H U_2^H g^{(2)} \cdots = U_1^H U_2^H \cdots U_{N-1}^H g^{(N-1)}.$$

In other words, the eigenvector $g = g^{(0)}$ of A corresponding to the eigenvalue λ can be expressed by

$$g = \Big(\prod_{k=1}^{N-1} U_k^H \Big) g^{(N-1)}.$$

Thus, the recursive formula of $\{V_k\}$ given in Algorithm 4 is an alternative algorithm of the product above:

$$V_1 = U_1^H, \quad V_2 = U_1^H U_2^H = V_1 U_2^H, \quad V_{N-1} = \prod_{k=1}^{N-1} U_k^H = V_{N-2} U_{N-1}^H. \qquad \square$$

Remark on Algorithm 13

First, we explain the main idea in the special case that $k = 2$. By assumption, $\xi_0 = \lambda_1$ and $\lambda_N \geqslant \eta$ give us the upper and lower bounds of the eigenvalues $\{\lambda_j\}_{j=1}^N$, respectively. The left-endpoint ξ_1 of the test interval should be an approximation of λ_2, even a rough approximation is still okay since the convergence of the bisection method is quite fast. Actually, a little smaller one is better since then we can ignore a subinterval on the left. If so, we do not need the double extension of the test interval on the left-hand side. For this, we may assume that $\lambda_N = \eta$. Then there are exact $N - 2$ eigenvalues located inside of the interval $[\eta, \xi_0]$. Suppose that these eigenvalues are located on $N-2$ equal points. Because the length of each equal division is $(\xi_0 - \eta)/(N - 1)$, the submaximal eigenvalue λ_2 should be located around

$$\xi_0 - \frac{1}{N-1}(\xi_0 - \eta) = \frac{1}{N-1}((N-2)\xi_0 + \eta).$$

That is the ξ_1 given in the algorithm corresponding to $k = 2$. Very often, ξ_1 is enough as the left-end point of the initial interval. If not, we need the subsequence $\{\xi_m\}_{m\geqslant 2}$. To which, the construction is as follows. Suppose that we are at ξ_{m-1}. Then choose ξ_m so that

$$\xi_{m-1} - \xi_m = 2(\xi_{m-2} - \xi_{m-1}).$$

This gives us ξ_m as in part (1) of the algorithm, in terms of the natural control by η. The factor 2 used above is optimal, simply based on the method of bisection.

For general $k \geqslant 2$, suppose that λ_{k-1} is known and we are going to compute λ_k. In this case, the first $k-2$ eigenvalues $\lambda_1, \ldots, \lambda_{k-2}$ play no role. The number of the eigenvalues decreases. Therefore, we need to make the following change:

$$N \to N - k + 2, \quad \lambda_1 \to \lambda_{k-1},$$

in the formula just obtained for $k = 2$. We have thus arrived at the expression of ξ_1 presented in part (1) of the algorithm. The change from $k = 2$ to general $k \geqslant 2$ stated in part (2) of the algorithm is now obvious.

Proof of reduction to tridiagonal matrix

We now come back to the main text. In the matrix form, the Hermitizability of A can be expressed as

$$\operatorname{Diag}(\mu)\, A = A^H \operatorname{Diag}(\mu) \Longleftrightarrow \operatorname{Diag}(\mu)\, A \operatorname{Diag}(\mu)^{-1} = A^H \tag{4}$$

(actually, it is better to rewrite $\mathrm{Diag}\,(\mu)^\alpha$ as $\mathrm{Diag}\,(\mu^\alpha)$ in computation). As used in parts (2) and (3) in Algorithm 3, it is easy to see that

$$A \text{ is Hermitizable} \Longleftrightarrow H := \mathrm{Diag}(\mu^{1/2})A\mathrm{Diag}(\mu^{-1/2}) \text{ is Hermitian.} \tag{5}$$

The assertion is clearly important since then every property and algorithm for the Hermitian matrix can be transferred to the Hermitizable one.

Next, for the Hermitian H, as shown in Section 2, there exists a unitary matrix U, as a product of some Householder transformations $\{U_k\}$ (unitary), such that

$$T := UHU^H \text{ becomes a tridiagonal, real, and symmetric matrix.} \tag{6}$$

As also shown in Section 2, the tridiagonal matrix T can be blocked. In other words, T may be reducible, it can be divided into several irreducible blocks, say $T = \mathrm{Diag}(\{T_j\})$, where T_j is irreducible tridiagonal, real, symmetric matrix. In general, the sum of some row of T can be positive and so is not a Q-matrix. Let $m = \sup_k(T\mathbb{1})(k) < \infty$ (which may be a condition when the matrix is infinite). Then the sum of each row of $T - mI$ is not positive, so is each row of $T_j - mI$. Fix j for a moment. By [5; Theorems 15 and 16], there exists a positive nearly $(T_j - mI)$-harmonic function h_j such that

$$Q_j := \mathrm{Diag}(h_j^{-1})(T_j - mI)\mathrm{Diag}(h_j)$$

is a birth-death Q-matrix. In particular, $T_j - mI$ and Q_j are isospectral. To return to the original setup, combining the family $\{h_j\}$ into a single vector h with the same ordering as the blocking of $\{T_j\}$. Then we can combine the family $\{Q_j\}$ into a single one with the same ordering just mentioned, denoted by $Q = \mathrm{Diag}(\{Q_j\})$. Furthermore, from the last formula, we obtain

$$Q = \mathrm{Diag}(h^{-1})(T - mI)\mathrm{Diag}(h).$$

Equivalently,

$$\mathrm{Diag}(h^{-1})T\mathrm{Diag}(h) = Q + mI =: Q^{(m)}. \tag{7}$$

Combining (5)–(7) together, we obtain the following similar transformation of A:

$$\mathrm{Diag}(h^{-1})U\mathrm{Diag}(\mu^{1/2})A\mathrm{Diag}(\mu^{-1/2})U^H\mathrm{Diag}(h) = Q^{(m)}.$$

Set

$$M = \mathrm{Diag}(\mu^{-1/2})U^H\mathrm{Diag}(h) \Longleftrightarrow M^{-1} = \mathrm{Diag}(h^{-1})U\mathrm{Diag}(\mu^{1/2}). \tag{8}$$

It follows that

$$M^{-1}AM = Q^{(m)} \Longleftrightarrow A = MQ^{(m)}M^{-1}.$$

Therefore,

$$Ag = \lambda g \Longleftrightarrow MQ^{(m)}M^{-1}g = \lambda g \Longleftrightarrow Q^{(m)}(M^{-1}g) = \lambda(M^{-1}g).$$

By [5; Theorem 10], with the inner product

$$\langle f, g\rangle := (Mf, Mg)_\mu$$

and $\tilde{f} := M^{-1}f\colon L^2(E,\mu) \to L^2(E, \langle\cdot,\cdot\rangle)$, we have

$$(f,g)_\mu = \langle \tilde{f}, \tilde{g}\rangle \quad \text{and} \quad (Af, g)_\mu = \langle Q^{(m)}\tilde{f}, \tilde{g}\rangle. \tag{9}$$

For this specific M defined by (8), we have here more explicit formulation. Note that

$$\begin{aligned}(Mf, Mg)_\mu &= (\mathrm{Diag}(\mu^{-1/2})\, U^H \mathrm{Diag}(h) f, \mathrm{Diag}(\mu^{-1/2}) U^H \mathrm{Diag}(h) g)_\mu \\ &= (U^H \mathrm{Diag}(h) f, U^H \mathrm{Diag}(h) g)_{\mathrm{d}x} \\ &= (\mathrm{Diag}(\bar{h})\mathrm{Diag}(h) f, g)_{\mathrm{d}x} \quad (\text{since } UU^H = I) \\ &= (f,g)_{\tilde{\mu}},\end{aligned}$$

where $\tilde{\mu} = |h|^2 \mathrm{d}x$ and $\mathrm{d}x$ means the uniform measure in the discrete case: $\mu_k \equiv 1$. More precisely, here in the first equality above, we have used the fact that

$$\begin{aligned}(\mathrm{Diag}(\mu^{-1/2})f, \mathrm{Diag}(\mu^{-1/2})g)_\mu &= (\mathrm{Diag}(\mu^{1/2})f, \mathrm{Diag}(\mu^{-1/2})g)_{\mathrm{d}x} \\ &= g^H \mathrm{Diag}(\mu^{-1/2})\mathrm{Diag}(\mu^{1/2}) f \\ &= (f,g)_{\mathrm{d}x},\end{aligned}$$

and in the second equality, we have used the fact that

$$(U^H f, U^H g)_{\mathrm{d}x} = g^H U U^H f = (f,g)_{\mathrm{d}x}.$$

Hence, we have

$$\langle \tilde{f}, \tilde{g}\rangle = (M\tilde{f}, M\tilde{g})_\mu = (\tilde{f}, \tilde{g})_{\tilde{\mu}}.$$

Therefore, we indeed have $L^2(E, \langle\cdot,\cdot\rangle) = L^2(E, \tilde{\mu})$. Thus, for the mapping $\tilde{f} := M^{-1}f\colon L^2(E,\mu) \to L^2(E,\tilde{\mu})$, by (9), we have the isometry

$$(f,g)_\mu = (\tilde{f}, \tilde{g})_{\tilde{\mu}},$$

and furthermore, the isospectral property

$$(Af, g)_\mu = \langle Q^{(m)}\tilde{f}, \tilde{g}\rangle = (Q^{(m)}\tilde{f}, \tilde{g})_{\tilde{\mu}}.$$

Here is our final conclusion.

Theorem 14 *For given Hermitizable A, define M by* (8). *Then the mapping $\tilde{f} := M^{-1}f$ from the complex $L^2(E,\mu)$ to the real $L^2(E,\tilde{\mu})$ is an isometry*: $(f,g)_\mu = (\tilde{f},\tilde{g})_{\tilde{\mu}}$. *Furthermore, it owns the isospectral property*: $(Af,g)_\mu = (Q^{(m)}\tilde{f},\tilde{g})_{\tilde{\mu}}$.

Remark on the computational complexity

Now, we mention the computational complexity of the algorithms used in the paper. The quasi-Hermitizing procedure in (2) requires $2N^2$ multiplications. The computation for the maximal eigenpair of the tridiagonal matrix using the method given in [7], as well as the one of bisection plus Thomas' algorithm for computing the other eigenpairs requires only $O(N)$ multiplications. The main work we need is Householder transformation which requires $2N^3/3$ multiplications. Refer to [18; p. 244].

5 Practical implementation on large scale matrices

In this section, we present some practical implementation of our algorithms on large scale matrices. For counting the number of computational operations, the additions and the multiplications are all collected together, and denote it by 'flops'. All experiments were performed by MatLab on a personal computer with the configuration: Intel(R) Xeon(R) CPU E5-2630 v3 @2.40 GHz and 32 GB of RAM.

Householder-based tridiagonalization

Assuming that the Hermitized $\hat{A}$ of the Hermitizable matrix A is at hand, we are going to propose the details of the Householder tridiagonalization of the Hermitian matrix $\hat{A}$.

We use the following notation. Let N, k ($\leqslant N$), and j ($\leqslant N$) be given three positive integers. Denote by I_N the identity matrix of order N, and by e_k the kth column of the identity matrix. Next, for given $0 < k < N$, an N-dimensional vector x, and an $N \times N$ matrix A, set

- $x(k+1:N)$: the vector of order $N-k$ obtained from x by deleting its first k entries;
- $a(k+1,j)$: the entry of A at the $(k+1)$th row and the jth column;
- $a(k+1:N,j)$: the vector of order $N-k$ obtained from A by deleting its first k entries of the jth column;
- $A(k+1:N,k+1:N)$: the $(N-k)\times(N-k)$ submatrix of A by deleting its first k rows and columns.

Suppose that Householder matrices $U_1,\ldots,U_{k-1}$ have been determined such that if

$$\hat{A}_{k-1} = (U_{k-1}\cdots U_1)\hat{A}(U_{k-1}\cdots U_1)^H,$$

then

$$\hat{A}_{k-1} = \begin{bmatrix} B_{11} & B_{12} & 0 \\ B_{21} & B_{22} & B_{23} \\ 0 & B_{32} & B_{33} \end{bmatrix},$$

where $B_{11} \in \mathbb{C}^{(k-1)\times(k-1)}$, $B_{12} \in \mathbb{C}^{(k-1)\times 1}$, $B_{21} = B_{12}^H$, $B_{22} \in \mathbb{C}$, $B_{23} \in$

$\mathbb{C}^{1\times(N-k)}$, $B_{32} = B_{23}^H$, and $B_{33} = B_{33}^H \in \mathbb{C}^{(N-k)\times(N-k)}$. The kth Householder transformation is computed by

$$\widehat{U}_k = I_{N-k} + \frac{vv^H}{\alpha}, \quad \alpha = s_k(b_{k+1}^{(k)} - s_k), \tag{10a}$$

$$0 \neq v = x^{(k)}(k+1:N) - y^{(k)}(k+1:N). \tag{10b}$$

Define

$$U_k = \begin{bmatrix} I_k & 0 \\ 0 & \widehat{U}_k \end{bmatrix}.$$

Then

$$\hat{A}_k = U_k \hat{A}_{k-1} U_k^H = \begin{bmatrix} B_{11} & B_{12} & 0 \\ B_{21} & B_{22} & B_{23}\widehat{U}_k^H \\ 0 & \widehat{U}_k B_{32} & \widehat{U}_k B_{33} \widehat{U}_k^H \end{bmatrix}.$$

Notice that it is not necessary to compute $B_{23}\widehat{U}_k^H$ and $\widehat{U}_k B_{32}$, since

$$\widehat{U}_k B_{32} = \|B_{32}\|_2 e_1 = s_k e_1 \in \mathbb{R}^{N-k}, \quad B_{23}\widehat{U}_k^H = (\widehat{U}_k B_{32})^H.$$

Thus, the leading $k \times k$ principal submatrix of $\hat{A}_k$ is a tridiagonal matrix. In the calculation of $\hat{A}_k$, it is important to exploit the Hermitian structure during the formation of the matrix. Note that

$$\begin{aligned} \widehat{U}_k B_{33} \widehat{U}_k^H &= \left(I_{N-k} + \frac{vv^H}{\alpha}\right) B_{33} \left(I_{N-k} + \frac{vv^H}{\alpha}\right)^H \\ &= B_{33} + \frac{vv^H}{\alpha} B_{33} + \frac{B_{33} vv^H}{\bar{\alpha}} + \frac{vv^H}{\alpha} B_{33} \frac{vv^H}{\bar{\alpha}}. \end{aligned}$$

Replacing the last term on the right-hand side by the sum of half and half of it, we obtain

$$\begin{aligned} \widehat{U}_k B_{33} \widehat{U}_k^H &= B_{33} + v\left(\frac{v^H B_{33}}{\alpha} + \frac{v^H B_{33} v}{2\alpha\bar{\alpha}} v^H\right) + \left(\frac{B_{33} v}{\bar{\alpha}} + v\frac{v^H B_{33} v}{2\alpha\bar{\alpha}}\right) v^H \\ &= B_{33} + vw^H + wv^H, \end{aligned} \tag{11}$$

where

$$w = \frac{B_{33} v}{\bar{\alpha}} + v\frac{v^H B_{33} v}{2\alpha\bar{\alpha}}.$$

Since only the upper triangular portion of this matrix needs to be calculated, the transition from $\hat{A}_{k-1}$ to $\hat{A}_k$ can be accomplished in only about $4(N-k)^2$ flops.

Algorithm 15 (Householder-based tridiagonalization) Given an Hermitian matrix $\hat{A} \in \mathbb{C}^{N\times N}$, the following algorithm overwrites $\hat{A}$ with $T = U\hat{A}U^H$,

where T is tridiagonal and $U = U_1 \cdots U_{N-1}$ is the product of Householder transformations.

Step 1 **for** $k = 1 : N-1$

Step 2 $x = \hat{a}(k+1:N,k)$, $s = \sqrt{x^H x}$

Step 3 $v = x$, $v(1) = v(1) - s$

Step 4 $\alpha = s(x(1) - s)$

Step 5 $p = \hat{A}(k+1:N, k+1:N)v/\bar{\alpha}$

Step 6 $w = p + (p^H v/(2\bar{\alpha}))v$

Step 7 $\hat{a}(k+1,k) = s$; $\hat{a}(k,k+1) = \hat{a}(k+1,k)$

Step 8 $\hat{A}(k+1:N,k+1:N) = \hat{A}(k+1:N,k+1:N) + vw^H + wv^H$

Step 9 **end**

The transition from $\hat{A}_{k-1}$ to $\hat{A}_k$ totally costs about $\sum_{k=1}^{N-2} 4(N-k)^2$ flops. This is $O(N^3)$ flops. This is implemented in Algorithm 15 by

$$\hat{A}(k+1:N,k+1:N) = \hat{A}(k+1:N,k+1:N) + vw^H + wv^H.$$

This main step makes the total cost of Algorithm 15 to be $O(N^3)$. This algorithm requires $4N^3/3$ flops when Hermitian is exploited in calculating the rank-2 updating as in equation (11). The matrix U is stored in factored form in the subdiagonal portion of $\hat{A}$. If U is explicitly required, then it can be formed with an additional $4N^3/3$ flops.

Remark that the Householder transformation here defined by (10) is different to the traditional one given in [9, pp. 234–237, 243, 244],

$$\widehat{U}_k = I_{N-k} - \beta v v^H, \quad \beta = \frac{2}{v^H v}, \tag{12a}$$

$$0 \neq v = x \pm \mathrm{e}^{\mathrm{i}\theta}\|x\|_2 e_k, \quad x = x^{(k)}(k+1:N). \tag{12b}$$

The differences stay at two aspects: (a) β in (12) is always set to be real, while α in (10) may be real or complex; (b) the resulted vector of (12) is real or complex, while the resulted vector of (10) is set real. If we use alternatively the traditional Householder transformation (12) in Algorithm 15, then we get a complex tridiagonal matrix. Thus, we need to generate not more than $N-1$ necessary rotations to transform conjugate complex (not real) entries on upper and down subdiagonals to real numbers.

Example 16 *In this example, we reduce an Hermitian matrix to a real symmetric tridiagonal matrix by Algorithm* 15. *We compare the Householder transformation* (10) *with the traditional Householder transformation* (12)*, denoted by HR and HC, respectively. The testing matrix is produced by the command 'rand' in MatLab and the elements of the matrices are chosen from* $[0, 10]$. *More precisely, let* $A_1 = 10 \cdot \mathtt{rand(N)}$ *and* $A_2 = 10 \cdot \mathtt{rand(N)}$*, where* N *is the matrix size. Then we take*

$$A = \frac{A_1 + A_1^T}{2} + \mathrm{i} \cdot \frac{A_2 - A_2^T}{2}.$$

Proof We apply the Householder-based tridiagonalization on A in three cases: $N = 1200,\ 2500$, and 5000. For each case, the whole procedure is carried out for 100 times and the average values of results are output. The numerical results are given in Tables 1 and 2. Note that the output display format is chosen as the scaled fixed point format with 5 digits. For instance, 4.5937e+2 represents 4.5937×10^2. Table 1 presents the *average* CPU times to work out the Householder transformation. The symbols 'CPU-U' and 'CPU' refer to the CPU times in seconds with and without constructing the unitary matrix U, respectively. Both of them are tested by the MatLab command `cputime`. Table 2 displays the accuracy of the methods, in which we adopt some *average* errors:

- err-U: $\|UU^H - I\|_\infty / \|U\|_\infty$,
- err-UAU-T: $\|UAU^H - T\|_\infty / \|A\|_\infty$,
- err-A-UTU: $\|U^H TU - A\|_\infty / \|A\|_\infty$,

where U is a unitary matrix arising in the Householder transformation and I is the identity matrix. Here, for vector $x = [x_1, \ldots, x_N]^*$ and matrix $A = [a_{ij}] \in \mathbb{C}^{N\times N}$, we define

$$\|x\|_\infty = \max_{i=1,\ldots,N} |x_i| \quad \text{and} \quad \|A\|_\infty = \max_{i=1,\ldots,N} \sum_{j=1}^{N} |a_{ij}|.$$

Table 1 Comparison of HC and HR with respect to CPU time for Example 16

N	CPU-U		CPU	
	HC	HR	HC	HR
1200	2.8727e+2	2.8064e+2	1.1235e+2	1.1122e+2
2500	2.8840e+3	2.8689e+3	1.1577e+3	1.1514e+3
5000	2.1321e+4	2.1264e+4	9.2617e+3	9.2376e+3

Table 2 Comparison of HC and HR with respect to accuracy for Example 16

N	err-U		err-UAU-T		err-A-UTU	
	HC	HR	HC	HR	HC	HR
1200	2.5599e-15	2.5140e-15	4.1728e-14	3.9616e-14	2.3152e-14	2.0504e-14
2500	3.3557e-15	3.2936e-15	7.9696e-14	7.4699e-14	4.6707e-14	3.8897e-14
5000	4.4415e-15	4.4061e-15	1.5283e-13	1.4406e-13	9.0779e-14	7.7541e-14

From the numerical results in Tables 1 and 2, we see that the performances of HR and HC are comparable with each other. HR saves a little bit of CPU times costed by HC. For instance, when $N = 1200$, HR saves 2.31% CPU times of computing T and U together, and 1.00% CPU times of computing T, respect to HC. Two average residual errors of HR are smaller than those of HC. Taking

$N = 1200$ for instance, HR improve 'err-U', 'err-UAU-T', and 'err-A-UTU' by 1.79%, 5.06%, and 11.4%, respectively. These numerical results successfully indicate the high efficiency of the proposed Householder transformation (HR), which directly transforms an Hermitian matrix into a real tridiagonal matrix with nonnegative subdiagonals. □

Top k eigenpairs

Now, we concentrate on computing top k eigenpairs of large scale matrices. After Hermitizing and tridiagonalizing, we reduce an Hermitizable matrix A to an isospectral symmetric tridiagonal matrix T. The core work of computing top k eigenpairs of A becomes the calculation of top k eigenpairs of T.

Note that for any tridiagonal matrix T, we can find a shift m such that the diagonal entries of $T - mI$ are negative. Without loss of generality, we build our algorithm on the following irreducible tridiagonal matrix:

$$T = \begin{bmatrix} -c_0 & b_0 & & & 0 \\ a_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & b_{N-1} \\ 0 & & & a_N & -c_N \end{bmatrix}, \tag{13}$$

where the sequences $\{b_j\}_{j=0}^{N-1}$ and $\{a_j\}_{j=1}^{N}$ are positive and $\{c_j\}_{j=0}^{N}$ is nonnegative. Let

$$E = \{j \in \mathbb{Z} \colon 0 \leqslant j < N+1\} \quad (N \leqslant \infty).$$

We may write $T \sim (a_j, -c_j, b_j)$ for simplicity. If $a_j = b_{j-1}$ $(j \in E \setminus \{0\})$, then T is symmetric.

We now present a new two-stage method of computing top k eigenpairs of the irreducible tridiagonal matrix T. In the first stage, we compute the largest eigenpair, denoted by (λ_1, g_1), of T, for instance, applying Algorithm 1 in [7]. In the second stage, we compute the other top $k-1$ eigenpairs, denoted by $(\lambda_2, g_2), \ldots, (\lambda_k, g_k)$, applying Algorithm 13 in this paper and the inverse iteration. These two stages are not separated strictly, but are jointed tightly. For convenience, we gather above steps into Algorithm 17 with a subroutine in Algorithm 18.

Algorithm 17 (Computing top k eigenpairs) Suppose that $T \sim (a_j, -c_j, b_j)$ is an irreducible tridiagonal matrix of the form (13). The following algorithm computes top k eigenpairs of T, denoted by $(\lambda_1, g_1), \ldots, (\lambda_k, g_k)$.

Step 1 Let $a_0 = 0$, $b_N = 0$,

$$m = \sup_{j \in E}(a_j + b_j - c_j)^+, \quad x^+ = \max\{x, 0\},$$

and

$$u_j = a_j b_{j-1}, \quad j \in E \setminus \{0\}.$$

Step 2 *Specific isospectral transformation.* Set $\widetilde{c}_j = c_j + m$ $(j \in E)$ and $\widetilde{b}_0 = \widetilde{c}_0$. Let

$$\widetilde{b}_j = \widetilde{c}_j - \frac{u_j}{\widetilde{b}_{j-1}}, \quad \widetilde{a}_j = \widetilde{c}_j - \widetilde{b}_j, \quad 1 \leqslant j < N, \quad \widetilde{a}_N = \frac{u_N}{\widetilde{b}_{N-1}}.$$

Then the tridiagonal matrix

$$\widetilde{T} \sim (\widetilde{a}_j, -\widetilde{c}_j, \widetilde{b}_j)$$

possesses the properties: both $(\widetilde{a}_j)$ and $\big(\widetilde{b}_j\big)$ are positive, the sum of each row equals zero except the $(N+1)$ th row $(\widetilde{c}_N \geqslant \widetilde{a}_N)$.

Step 3 *Symmetrizing.* Define the symmetric tridiagonal matrix

$$T^{\mathrm{sym}} \sim (a_j^{\mathrm{sym}}, -c_j^{\mathrm{sym}}, b_j^{\mathrm{sym}})$$

as follows:

$$c_j^{\mathrm{sym}} = \widetilde{c}_j \ (j \in E), \quad a_j^{\mathrm{sym}} = b_{j-1}^{\mathrm{sym}} = \sqrt{u_j} \ (j \in E \setminus \{0\}).$$

Step 4 *Computing the maximal eigenpair.* If $\widetilde{c}_N = \widetilde{a}_N$, then T^{sym} has the maximal eigenvalue $\lambda_1^{\mathrm{sym}} = 0$ with eigenvector $g_1^{\mathrm{sym}} = \sqrt{\widetilde{\mu}}$:

$$\widetilde{\mu}_0 = 1, \quad \widetilde{\mu}_j = \widetilde{\mu}_{j-1} \frac{\widetilde{b}_{j-1}}{\widetilde{a}_j}, \quad j \in E \setminus \{0\}. \tag{14}$$

More economically,

$$g_1^{\mathrm{sym}}(0) = 1, \quad g_1^{\mathrm{sym}}(j) = g_1^{\mathrm{sym}}(j-1) \frac{\tilde{b}_{j-1}}{\sqrt{u_j}}, \quad j \in E \setminus \{0\}. \tag{15}$$

Otherwise, set $\widetilde{b}_N = \widetilde{c}_N - \widetilde{a}_N$. The jth approximation of the maximal eigenpair is computed by Algorithm 18. Then $(-z_j, v^{(j)})$ converges to the maximal eigenpair of T^{sym}:

$$\lambda_1^{\mathrm{sym}} = -\lim_{j\to\infty} z_j, \quad g_1^{\mathrm{sym}} = \lim_{j\to\infty} v^{(j)}.$$

Step 5 *Computing the subsequent eigenpairs.* Compute $k-1$ eigenvalues, $\lambda_2^{\mathrm{sym}}, \ldots, \lambda_k^{\mathrm{sym}}$, of T^{sym} by the bisection method (Algorithm 13).

For each j $(2 \leqslant j \leqslant k)$, compute the eigenvector g_j^{sym} corresponding to λ_j^{sym} of T^{sym}, by the inverse iteration with the shift λ_j^{sym} and an initial vector $v^{(0)}$. The initial vector $v^{(0)}$ is generated as follows: choose a vector $x^{(j)} \notin \mathrm{span}\{g_1^{\mathrm{sym}}, \ldots, g_{j-1}^{\mathrm{sym}}\}$, and compute $v^{(0)}$:

$$w^{(0)} = x_j - \sum_{i=1}^{j-1} [(g_i^{\mathrm{sym}})^H x_j] g_i, \quad v^{(0)} = \frac{w^{(0)}}{\sqrt{(w^{(0)})^* w^{(0)}}}.$$

The modified version of the Gram-Schmidt method [9, pp. 254, 255] is used in the practical implementation.

Step 6 *Returning to the original top k eigenpairs.* To go back to the original matrix T, the top k eigenpairs are

$$\lambda_j = \lambda_j^{\rm sym} + m, \quad g_j = {\rm diag}(h^\mu) g_j^{\rm sym},$$

where ${\rm diag}(h^\mu)$ is the diagonal matrix having diagonal elements (h_j^μ):

$$h_0^\mu = 1, \quad h_j^\mu = h_{j-1}^\mu \frac{\sqrt{u_j}}{b_{j-1}}, \quad j \in E \setminus \{0\}. \tag{16}$$

Algorithm 18 (The jth approximation of the maximal eigenpair) With computed $\widetilde{a}_j$'s and $\widetilde{b}_j$'s, this algorithm computes the jth approximation of the maximal eigenpair: $(z_j, v^{(j)})$.

Step 1 Define the upper triangle matrix (M_{ij}) and the vector (Φ_i) as follows:

$$M_{ii} = 1, \quad M_{ij} = M_{i,j-1} \frac{\widetilde{a}_j}{\widetilde{b}_{j-1}} = M_{i,j-1} \frac{u_j}{\widetilde{b}_{j-1}^2}, \quad 1 \leqslant i+1 \leqslant j \leqslant N,$$
$$\Phi_i = \sum_{i \leqslant j \leqslant N} \frac{M_{ij}}{\widetilde{b}_j}, \quad 0 \leqslant i \leqslant N.$$

Step 2 Choose

$$w^{(0)} = \sqrt{\Phi}, \quad v^{(0)} = \frac{w^{(0)}}{\sqrt{w^{(0)*} w^{(0)}}}.$$

Step 3 For a computed vector $v^{(j)}$ $(j \geqslant 0)$, compute ζ_j:

$$\zeta_j = \sup_{0 \leqslant n \leqslant N} \frac{1}{\sqrt{\widetilde{b}_n} v_n^{(j)} - \sqrt{\widetilde{a}_{n+1}} v_{n+1}^{(j)}} \left(\sum_{i=0}^{n} v_i^{(j)} \sqrt{\frac{M_{in}}{\widetilde{b}_i}} \right), \quad j \geqslant 0,$$

with assuming $\widetilde{a}_{N+1} = 0$.

Step 4 With setting $z_j = 1/\zeta_j$, solve $w^{(j+1)}$:

$$(-T^{\rm sym} - z^{(j)} I) w^{(j+1)} = v^{(j)}.$$

Step 5 Compute $v^{(j+1)}$:

$$v^{(j+1)} = \frac{w^{(j+1)}}{\sqrt{w^{(j+1)*} w^{(j+1)}}}.$$

Remark that if $\widetilde{c}_N = \widetilde{a}_N$, the way of computing the maximal eigenpair, (λ_1, g_1), in Algorithm 17, is different from that in Algorithm 1 in [7]. But their

results are the same: if $\widetilde{c}_N = \widetilde{a}_N$, then Algorithm 17 shows that T has the maximal eigenvalue $\lambda_1 = m$ with eigenvector $g_1 = h$:

$$h_0 = 1, \quad h_j = h_{j-1}\frac{\widetilde{b}_{j-1}}{b_{j-1}}, \quad j \in E \setminus \{0\}. \tag{17}$$

These two algorithms are coincided since

$$h_j^{\mu} = \frac{h_j}{\sqrt{\widetilde{\mu}_j}}, \quad j \in E \setminus \{0\}.$$

With applying equations (16) and (17), one can simplify the computation of g_1^{sym} in (14) into the economical formula in (15).

One important point is that the non-symmetric matrix $\widetilde{T}$ and the symmetric matrix T^{sym} are coupled together in Algorithm 17. This coupling idea has been introduced in [5; Section 4] and [7; Section 4.4]. For a short explanation, the spectrum of $T - mI$ is transformed (in Step 2) to the one of $\widetilde{T}$, which is a birth-death Q-matrix (see [7; Definition 7]); and then the non-symmetric matrix $\widetilde{T}$ is symmetrized to a symmetric matrix T^{sym} in Step 3. The maximal pair is computed by coupling $\widetilde{T}$ and T^{sym}; see Algorithm 18 and Step 4 of Algorithm 17. The subsequent eigenpairs are computed by applying Algorithm 13 and the iteration method on T^{sym}. We have two reasons: one is that T^{sym} is symmetric and has eigenvectors which are orthogonal to each other; another one is that T^{sym} has been generated during the computation of the maximal eigenpairs, and it does not rise extra computational flops. At last, the top k eigenpairs are generated for the original matrix T.

Clearly, we do not have to go to the last Step 6 every time when using Algorithm 17. Actually, for very large N, this step is risky. Especially for non-symmetric matrices, the overflow happens very often. This is due to the limitation of the accuracy of the machine. At this time, if you still want to calculate, then one may use the above iterative formula (17). Because the sequences $\{\tilde{b}_k\}$ and $\{b_k\}$ are known, and so is the ratio $\tilde{b}_j/b_j$ which does not overflow (see [6] for more related details on Hermitizable complex tridiagonal matrices).

Example 19 *In this example, we compute top k eigenpairs of a symmetric tridiagonal matrix of the form* (13)*,*

$$T \sim (a_j, -c_j, b_j) \in \mathbb{R}^{N\times N},$$

where a_j and c_j are random positive integers which are less than or equal to N, generated by the command 'randi' in MatLab, and $b_{j-1} = a_j$ for $j = 1, \ldots, N$.

Proof We apply Algorithm 17 and the MatLab functions, `eig` and `eigs`, to compute top k eigenvalues and corresponding eigenvectors of T, respectively. For comparison, we set $k = 3$, and $N = 5000$, 10000, 15000, 20000. We run the whole process of each case for 1000 times and take the average numerical results.

Note that `eig` computes all eigenpairs, and `eigs` is called by the command `eigs(T,k,'LA')`. For the iterative methods (Algorithm 17 and `eigs`), the convergence tolerance is set as `tol` $= 10^{-10}$ and the maximal number of iterations is 100.

Let V denote the matrix consisting of eigenvectors and D the diagonal matrix with eigenvalues on the diagonal. The numerical results are given in Table 3, in which 'CPU' refers to the CPU time in seconds, 'Res' represents the relative residual error $\|TV - VD\|_\infty / \|T\|_\infty$, and 'NaN' refers to no answer.

Table 3 Numerical results of Example 19: CPU times and residual errors

N	Algorithm 17		`eigs`		`eig`	
	CPU	Res	CPU	Res	CPU	Res
5000	2.2367	4.2206e-11	6.4821e-1	3.3668e-12	2.1670	6.5837e-15
10000	6.8966	7.7877e-11	9.3459e-1	3.4931e-12	5.1046	7.1151e-15
15000	7.9176e+1	5.8389e-11	2.4787	3.5428e-12	4.5517e+1	7.1183e-15
20000	3.6226e+1	9.0249e-11	3.3654e-1	NaN	2.2656e+1	6.7109e-15

From the numerical results, it follows that Algorithm 17 and the MatLab function `eig` correctly compute the top 3 eigenpairs in all cases, and they cost almost same CPU times. The MatLab function `eigs` successfully obtains the top 3 eigenpairs in the first three cases. But when $N = 20000$, it fails to converge at one time of 1000 tests and no result is obtained. Thus, we can conclude that the proposed new method is more reliable than the MatLab function `eigs`. Moreover, Algorithm 17 also has the comparable level of accuracy with the MatLab function `eigs` in the first three cases. □

Example 20 *In this example, we compute top k eigenpairs of an Hermitizable tridiagonal matrix of the form* (13),

$$T \sim (a_j, -c_j, b_j) \in \mathbb{R}^{N\times N},$$

where $a_j \equiv a$, $b_j \equiv b$, and $c_j \equiv c$ are positive. We test the following two cases:

Case 1 $a = 2$, $b = 1$, *and* $c = 3$;

Case 2 $a = 1$, $b = 2$, *and* $c = 3$.

Proof Let $(\lambda_j^{\text{exact}}, g_j^{\text{exact}})$ and (λ_j, g_j) denote the exact and computed eigenpairs of $T \sim (a, -c, b)$, respectively. They are defined by

$$\lambda_j^{\text{exact}} = 2\sqrt{ab}\cos\frac{j\pi}{N+1} - c,$$
$$g_j^{\text{exact}}(i) = \left(\sqrt{\frac{a}{b}}\right)^i \sin\frac{ij\pi}{N+1}, \quad i = 1, \ldots, N.$$

For comparison, we define the error of the computed eigenvalues by

$$\text{ERR} = \max_{j=1,\ldots,k} |\lambda_j - \lambda_j^{\text{exact}}|.$$

Let s_j denote the sign changing time of the jth eigenvector and the vector $\mathrm{SCT} = [s_1, \ldots, s_k]$ denote the sign changing times of each of the top k eigenvalues. Remember that the jth exact eigenvector has $j - 1$ times of sign changing.

In the two cases listed above, we set $k = 3$ and apply Algorithm 17 and the MatLab functions, `eig` and `eigs`, to compute the top k eigenvalues and corresponding eigenvectors of T, respectively. Note that `eig` computes all eigenpairs and `eigs` is called by the command `eigs(T,k,'LR')`. For the iterative methods, Algorithm 17 and `eigs`, the convergence tolerance is set as $\mathtt{tol} = 10^{-8}$ and the maximal number of iterations is 300.

From the numerical results in Tables 4 and 5, we see that Algorithm 17 correctly computes the top k eigenpairs in all cases; but both `eig` and `eigs` fail in most of cases, even when the size of T is very small. We underline the values of ERR which are larger than 10^{-6}, and also underline the values of SCTs if they are not correct. The notation '–' means that it is not necessary to compute.

Table 4 Numerical results of Example 20: errors of eigenpairs in Case 1

N	Algorithm 17		`eigs`		`eig`	
	ERR	SCT	ERR	SCT	ERR	SCT
51	4.4409e-16	[0,1,2]	3.4233e-7	[0,1,2]	6.7230e-11	[0,1,2]
52	3.6082e-16	[0,1,2]	5.8039e-6	[0,1,2]	1.4893e-10	[0,1,2]
83	1.8874e-15	[0,1,2]	5.0430e-5	[0,1,2]	5.4506e-7	[0,1,2]
84	1.4710e-15	[0,1,2]	1.0336e-2	[0,1,1]	1.0819e-6	[0,1,2]
103	1.3323e-15	[0,1,2]	–	–	1.0916e-4	[0,1,2]
104	1.9706e-15	[0,1,2]	–	–	5.6538	[102,102,103]
10000	3.3307e-16	[0,1,2]	–	–	–	–
20000	1.6037e-13	[0,1,2]	–	–	–	–

Table 5 Numerical results of Example 20: errors of eigenpairs in Case 2

N	Algorithm 17		`eigs`		`eig`	
	ERR	SCT	ERR	SCT	ERR	SCT
44	1.6653e-16	[0,1,2]	1.8505e-7	[0,1,2]	4.1633e-16	[0,1,2]
45	2.7756e-16	[0,1,2]	5.6594e-6	[0,1,2]	6.6613e-16	[0,1,2]
104	1.9706e-15	[0,1,2]	1.0453e-3	[0,1,2]	1.0112e-11	[0,1,2]
105	1.8041e-15	[0,1,2]	7.7746e-3	[0,2,2]	3.3805e-11	[0,1,2]
106	1.5821e-15	[0,1,2]	–	–	1.2713e-11	[1,3,2]
160	9.9920e-16	[0,1,2]	–	–	3.2369e-9	[30,33,36]
161	7.4940e-16	[0,1,2]	–	–	5.6561	[160,160,157]
10000	3.3307e-16	[0,1,2]	–	–	–	–
20000	1.6037e-13	[0,1,2]	–	–	–	–

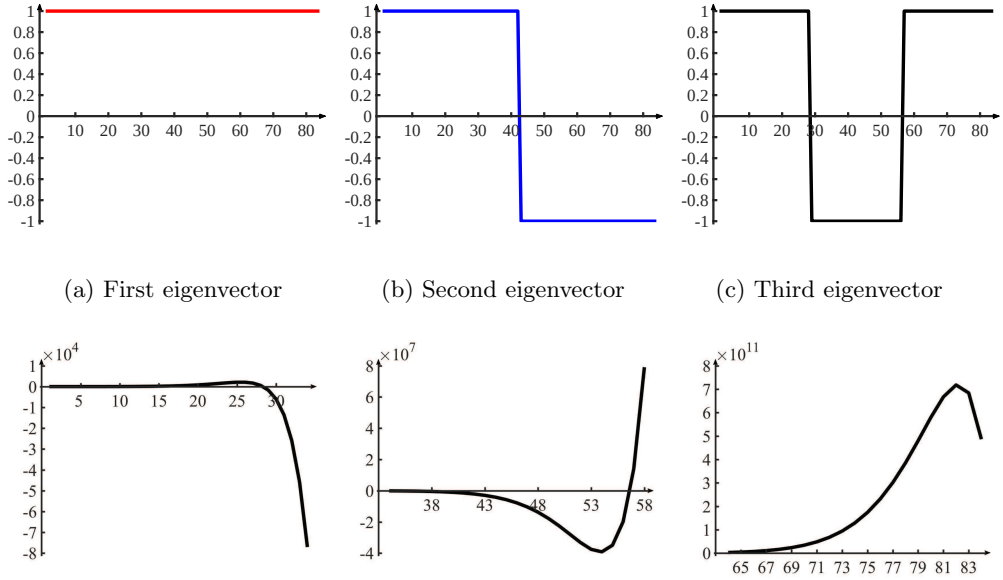
Now, we give a detailed analysis as follows.

• `Algorithm 17` computes correctly the top k eigenpairs in both Cases 1 and 2. The maximal errors of the computed top k eigenvalues are always less than 10^{-6}, and the times of sign changing of the computed eigenvectors are also correct, that is, SCT $= [0, 1, 2]$.

• `eigs` computes correctly the top k eigenpairs when $N \leqslant 51$ in Case 1 (or $N \leqslant 44$ in Case 2). The maximal errors of the computed top k eigenvalues become larger than 10^{-6} when $N \geqslant 52$ in Case 1 (or $N \geqslant 45$ in Case 2). Moreover, the times of sign changing of the computed eigenvectors become wrong when $N \geqslant 84$ in Case 1 (or $N \geqslant 105$ in Case 2).

• `eig` computes correctly the top k eigenpairs when $N \leqslant 83$ in Case 1 (or $N \leqslant 105$ in Case 2). The maximal errors of the computed top k eigenvalues become larger than 10^{-6} when $N \geqslant 84$ in Case 1 (or $N \geqslant 161$ in Case 2). Moreover, the times of sign changing of the computed eigenvectors become wrong when $N \geqslant 104$ in Case 1 (or $N \geqslant 106$ in Case 2).

For Case 1 with $N = 84$, the signs of the top $k = 3$ exact and computed eigenvectors are shown in Figs. 2–5. In Fig. 2 (a)–(c), the signs of the first, second and third maximal exact eigenvectors (from left to right) are drawn; and the times of sign changing are 0, 1 and 2, respectively. That is SCT $= [0, 1, 2]$. In Fig. 2 (d)–(f), three parts of the third eigenvector are drawn. The two points at which the signs of the third eigenvector change are shown in Fig. 2 (d) and (e).

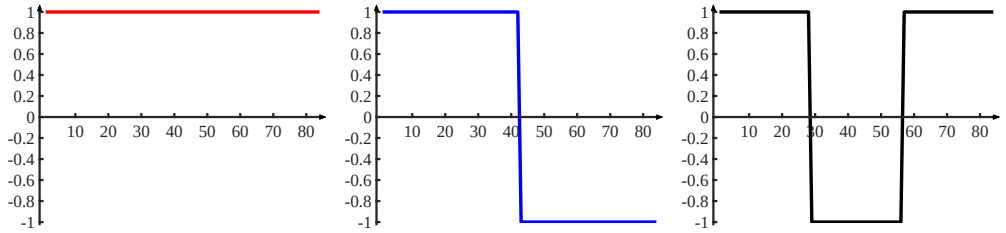


(a) First eigenvector (b) Second eigenvector (c) Third eigenvector

(d) First part from 1 to 34 (e) Second part from 34 to 58 (f) Third part from 64 to 84

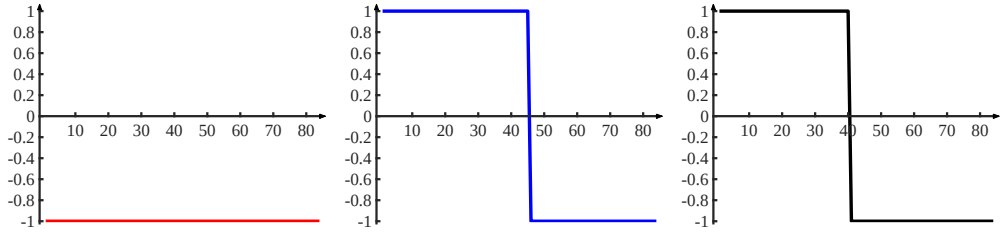
Fig. 2 (a)–(c) The sign of the top $k = 3$ exact eigenvectors of $T \sim (2, -3, 1)$ with $N = 84$. (d)–(f) Three parts of the third exact eigenvector.

Figures 3–5 indicate the signs of the top 3 eigenvectors, computed by Algorithm 17, `eigs`, and `eig`, respectively; and the SCTs are $[0, 1, 2]$, $[0, 1, 1]$, and $[0, 1, 2]$. From Fig. 4, it follows that the sign of the third eigenvector computed by `eigs` is not correct.



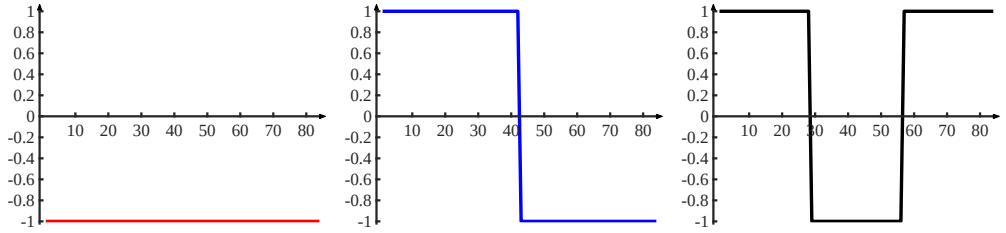
(a) First eigenvector (b) Second eigenvector (c) Third eigenvector

Fig. 3 The sign of the top $k = 3$ eigenvectors computed by Algorithm 17 of $T \sim (2, -3, 1)$ with $N = 84$



(a) First eigenvector (b) Second eigenvector (c) Third eigenvector

Fig. 4 The sign of the top $k = 3$ eigenvectors computed by `eigs` of $T \sim (2, -3, 1)$ with $N = 84$



(a) First eigenvector (b) Second eigenvector (c) Third eigenvector

Fig. 5 The sign of the top $k = 3$ eigenvectors computed by `eig` of $T \sim (2, -3, 1)$ with $N = 84$

As in Example 19, we list the CPU times ('CPU') and the relative residual errors ('Res') of Cases 1 and 2 in Tables 6 and 7, respectively. Comparing these two tables with Tables 4 and 5, respectively, it follows that the error estimates by using 'ERR' plus 'SCT' are more precise than using 'Res'. Note that the CPU times are obtained by running the programs for 100 times and taking the

Table 6 Numerical results of Example 20: CPU times and residual errors in Case 1

N	Algorithm 17		`eigs`		`eig`	
	CPU	Res	CPU	Res	CPU	Res
51	1.5200e-2	2.7062e-16	3.0200e-2	3.2479e-11	4.3000e-3	1.1148e-15
52	8.9000e-3	3.1456e-16	4.6400e-2	7.6735e-11	4.7000e-3	1.1125e-15
83	1.5500e-2	6.0137e-16	6.9300e-2	3.4396e-11	1.5700e-2	1.5400e-15
84	1.5800e-2	6.1987e-16	4.7600e-2	1.9611e-10	1.8100e-2	1.6283e-15
103	3.0800e-2	7.0777e-16	–	–	2.4100e-2	2.6738e-15
104	3.4000e-2	6.5688e-16	–	–	2.7500e-2	7.5496e-15

Table 7 Numerical results of Example 20: CPU times and residual errors in Case 2

N	Algorithm 17		`eigs`		`eig`	
	CPU	Res	CPU	Res	CPU	Res
44	1.2900e-2	1.7810e-16	2.8100e-2	3.1130e-12	3.6000e-3	1.8735e-15
45	8.9000e-3	3.1456e-16	4.6400e-2	7.6735e-11	4.7000e-3	1.1125e-15
104	3.4500e-2	6.8695e-16	9.9700e-2	7.5717e-11	2.4300e-2	3.2867e-15
105	3.0000e-2	6.1525e-16	6.7300e-2	2.5779e-11	2.0000e-2	1.7139e-15
106	3.2800e-2	6.2913e-16	–	–	2.6600e-2	1.5451e-15
160	3.9500e-2	4.1402e-16	–	–	1.2860e-1	1.7278e-15
161	3.9900e-2	4.0246e-16	–	–	1.2930e-1	2.5905e-15

average values. The relative residual errors of Algorithm 17 are less than those of `eigs` and `eig`. Algorithm 17 costs less CPU times than `eigs` and has comparable speed with `eig`.

From above numerical results, we conclude that Algorithm 17 performs better than `eigs` and `eig`, and is very feasible and reliable to compute the top k eigenpairs of large scale matrices. □

Acknowledgements We are grateful to two anonymous referees for providing many useful comments and suggestions. Acknowledges are given to Ying-Chao Xie and Yue-Shuang Li for fruitful discussion and valuable suggestions. For the algorithms in the paper, a package in MatLab is preparing by Jia and Pang and should appear soon. This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 12090011, 11771046, 11771188, 11771189), the National Key R & D Program of China (No. 2020YFA0712900), the Natural Science Foundation of Jiangsu Province (Grant No. BK20171162), the project from the Ministry of Education in China, and the Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

1. Cao Z H. Matrix Eigenvalue Problem. Shanghai: Shanghai Scientific & Technical Publishers, 1983 (in Chinese)
2. Chen M F. Eigenvalues, Inequalities, and Ergodic Theory. London: Springer, 2005
3. Chen M F. Efficient initials for computing maximal eigenpair. Front Math China, 2016, 11(6): 1379–1418
4. Chen M F. Global algorithms for maximal eigenpair. Front Math China, 2017, 12(5): 1023–1043
5. Chen M F. Hermitizable, isospectral complex matrices or differential operators. Front Math China, 2018, 13(6): 1267–1311
6. Chen M F. On spectrum of Hermitizable tridiagonal matrices. Front Math China, 2020, 15(2): 285–303
7. Chen M F, Li Y S. Development of powerful algorithm for maximal eigenpair. Front Math China, 2019, 14(3): 493–519
8. Chung K L, Yan W M. The complex Householder transform. IEEE Trans Signal Process, 1997, 45(9): 2374–2376
9. Golub G H, Van Loan C F. Matrix Computations. 4th ed. Baltimore: The Johns Hopkins Univ Press, 2013
10. Householder A S. Unitary triangularization of a nonsymmetric matrix. J Assoc Comput Mach, 1958, 5: 339–342
11. Jiang E X. Symmetric Matrix Computation. Shanghai: Shanghai Scientific & Technical Publishers, 1984 (in Chinese)
12. Min C. A new understanding of the QR method. J Korean Soc Ind Appl Math, 2010, 14(1): 29–34
13. Niño A, Muñoz-Caro C, Reyes S. A concurrent object-oriented approach to the eigenproblem treatment in shared memory multicore environments. Lecture Notes in Computer Sci, Vol 6782. Cham: Springer, 2011, 630–642
14. Parlett B N. The Symmetric Eigenvalue Problem. Philadelphia: SIAM, 1998
15. Press W H, Teukolsky S A, Vetterling W T, Flannery B P. Numerical Recipes. The Art of Scientific Computing. 3rd ed. Cambridge: Cambridge Univ Press, 2007
16. Shukuzawa O, Suzuki T, Yokota I. Real tridiagonalization of Hermitian matrices by modified Householder transformation. Proc Japan Acad Ser A, 1996, 72(5): 102–103
17. Wang Z J. Householder transformation for Hermitizable matrix. Master Thesis. Beijing: Beijing Normal Univ, 2018
18. Wilkinson J H. The Algebraic Eigenvalue Problem. Oxford: Oxford Univ Press, 1965