

Development of powerful algorithm for maximal eigenpair

Mu-Fa Chen and Yue-Shuang Li

(Beijing Normal University)

August 28, 2018

Abstract

Based on a series of recent papers, a powerful algorithm is reformulated for computing the maximal eigenpair of self-adjoint complex tridiagonal matrices. In parallel, the same problem in a particular case for computing the sub-maximal eigenpair is also introduced. The key ideas for each critical improvement are explained. To illustrate and compare these algorithms, some examples are included.

2000 *Mathematics Subject Classification*: 15A18, 65F15, 93E15, 60J27

Key words and phrases. Powerful algorithm, maximal eigenpair, sub-maximal eigenpair, Hermitizable tridiagonal matrix.

1 Introduction. A powerful algorithm and a typical example

Matrix eigenvalues play important role in many areas, not only in mathematics but also in quantum mechanics. In the past 170 years, a large number of publications, as well as libraries have been devoted to the study on eigenproblems. Refer to [17, 14] and the references within. An algorithm which is closely related to the aim of the paper is the so-called Householder's decomposition or reduction (which was listed in [2] as the forth of 10 top algorithms in the 20th century). About the decomposition technique, six algorithms are surveyed in [15], including Householder transformation [1]. It transforms a Hermitian matrix into a real symmetric tridiagonal one. There are three aspects of contributions in the algorithm stated below. The first one is an extension of the Householder transformation from Hermitian to Hermitizable. This is an easier part ([11; Theorem 24]). The second one is reducing further to a birth-death type Q -matrix (that is the tridiagonal matrix having positive sub-diagonals). This enables us to use some long cumulated results in the study on probability theory. As will be seen in Section 4, it takes a long trip to arrive at the present formulation of Algorithm 1 which is very efficient and theoretically complete. The mature algorithm for birth-death type Q -matrix is the third contribution to Algorithm 1. The main aim of the paper is to illustrate the power of the algorithm and to explain the main steps in the development of the algorithm.

To move further, let us define the Hermitizable matrix [11] first. Let

$$E = \{k \in \mathbb{Z}_+ : 0 \leq k < N + 1\}.$$

A matrix $A = (a_{ij} : i, j \in E)$ is called Hermitizable if there exists a positive measure $(\mu_i : i \in E)$ such that

$$\mu_i a_{ij} = \mu_j \bar{a}_{ji}, \quad i, j \in E,$$

where $\bar{a}$ denotes the conjugate of a . With μ at hand, the matrix $(\sqrt{\mu_i} a_{ij} / \sqrt{\mu_j} : i, j \in E)$ becomes Hermitian having the same spectrum as A . As mentioned above, for a Hermitizable matrix, we can transform it into a real symmetric tridiagonal one with the help of Householder transformation. Besides, a reducible tridiagonal matrix can be decomposed into irreducible sub-matrices. Thus, we consider only irreducible tridiagonal one in this paper. In what follows, we focus on the tridiagonal matrices of the following form on finite space $E = \{0, 1, \dots, N\}$.

$$T = \begin{pmatrix} -c_0 & b_0 & & & \\ a_1 & -c_1 & b_1 & & \\ & a_2 & -c_2 & b_2 & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots & b_{N-1} \\ & & & & a_N & -c_N \end{pmatrix}. \quad (1)$$

Here we assume that T is Hermitizable:

$$(c_k) \text{ is real, } (a_k) \text{ and } (b_k) \text{ are complex but } a_{k+1}b_k > 0 \text{ } (0 \leq k < N).$$

As usual, the 'eigenpair' means the twins consisting of an eigenvalue and its eigenvector. In what follows, we simply write the tridiagonal matrix as $T \sim (a_k, -c_k, b_k)$, since such a matrix is determined by the three sequences $\{a_k\}_{k=1}^N$, $\{-c_k\}_{k=0}^N$, and $\{b_k\}_{k=0}^{N-1}$. Now, the powerful algorithm for maximal eigenpair, according to [7-10] and [11; §4], can be stated as follows.

Algorithm 1 Suppose that T is a Hermitizable tridiagonal matrix of form (1). Before computing the maximal eigenpair of T , we need prepare Steps 1 ~ 3.

Step 1. Let

$$m = \sup_{k \in E} (-c_k + |a_k| + |b_k|)^+, \quad x^+ := \max\{x, 0\},$$

and

$$u_k = a_k b_{k-1}, \quad k \in E \setminus \{0\}.$$

Set $\tilde{c}_k = c_k + m$ ($k \in E$) and $\tilde{b}_0 = \tilde{c}_0 > 0$. Next, let

$$\begin{cases} \tilde{b}_k = \tilde{c}_k - u_k/\tilde{b}_{k-1}, & \tilde{a}_k = \tilde{c}_k - \tilde{b}_k, & 1 \leq k < N; \\ \tilde{a}_N = u_N/\tilde{b}_{N-1}. \end{cases}$$

More explicitly,

$$\begin{cases} \tilde{b}_0 = \tilde{c}_0, \\ \tilde{b}_k = \tilde{c}_k - \frac{u_k}{\tilde{c}_{k-1} - \frac{u_{k-1}}{\tilde{c}_{k-2} - \frac{u_{k-2}}{\ddots - \frac{u_2}{\tilde{c}_2 - \frac{u_1}{\tilde{c}_1 - \frac{u_1}{\tilde{c}_0}}}}}}, & 1 \leq k < N, \\ \tilde{a}_k = \tilde{c}_k - \tilde{b}_k, & 1 \leq k < N, \\ \tilde{a}_N = \frac{u_N}{\tilde{b}_{N-1}}. \end{cases}$$

Then the tridiagonal matrix $\tilde{Q} \sim (\tilde{a}_k, -\tilde{c}_k, \tilde{b}_k)$ possesses the properties: both $(\tilde{a}_k)$ and $(\tilde{b}_k)$ are positive, the sum of each row equals zero except the N th row ($\tilde{c}_N \geq \tilde{a}_N$). If $\tilde{c}_N = \tilde{a}_N$, then T has the maximal eigenvalue $\lambda_{\max} = m$ with eigenvector $g_{\max} = h$:

$$h_0 = 1, \quad h_k = h_{k-1} \frac{\tilde{b}_{k-1}}{b_{k-1}}, \quad k \in E \setminus \{0\}.$$

Otherwise go to the next step.

Step 2. Define the symmetric tridiagonal matrix $Q^{\text{sym}} \sim (a_k^{\text{sym}}, -c_k^{\text{sym}}, b_k^{\text{sym}})$ as follows:

$$\begin{aligned} c_k^{\text{sym}} &= \tilde{c}_k, & k \in E; \\ a_k^{\text{sym}} &= b_{k-1}^{\text{sym}} = \sqrt{a_k b_{k-1}}, & k \in E \setminus \{0\}. \end{aligned}$$

Step 3. Define the upper triangular matrix (M_{kj}) and vector (Φ_k) as follows.

$$\begin{aligned} M_{kk} &= 1, \quad M_{kj} = M_{k,j-1} \frac{\tilde{a}_j}{\tilde{b}_{j-1}} \left[= \frac{\tilde{a}_{k+1} \cdots \tilde{a}_j}{\tilde{b}_k \cdots \tilde{b}_{j-1}} \right], & 1 \leq k+1 \leq j \leq N; \\ \Phi_k &= \frac{1}{\tilde{b}_k} + \sum_{k+1 \leq j \leq N} \frac{\tilde{a}_{k+1} \cdots \tilde{a}_j}{\tilde{b}_k \cdots \tilde{b}_j} = \sum_{k \leq j \leq N} \frac{M_{kj}}{\tilde{b}_j}, & 0 \leq k \leq N. \end{aligned}$$

With $(\tilde{a}_k, \tilde{b}_k)$, Q^{sym} , M and Φ at hand, we can now start our iterations. Note that one may use the parallel computing the next step.

Step 4. For given $v^{(k)}$ ($k \geq 0$), define

$$\zeta_k = \sup_{0 \leq n \leq N} \frac{1}{\sqrt{\tilde{b}_n} v_n^{(k)} - \sqrt{\tilde{a}_{n+1}} v_{n+1}^{(k)}} \sum_{j=0}^n v_j^{(k)} \sqrt{\frac{M_{jn}}{\tilde{b}_n}}, \quad k \geq 0, \quad (2)$$

with a convention that $\tilde{a}_{N+1} = 0$. As in [11; §4], choose

$$w^{(0)} = \sqrt{\Phi}, \quad v^{(0)} = w^{(0)} / \sqrt{w^{(0)} * w^{(0)}}, \quad z^{(0)} = \zeta_0^{-1},$$

where ζ_0 is defined by (2) with $k = 0$. For each $k \geq 1$, solve $w^{(k)}$:

$$(-Q^{\text{sym}} - z^{(k-1)}I)w^{(k)} = v^{(k-1)} \quad (3)$$

and define

$$v^{(k)} = \frac{w^{(k)}}{\sqrt{w^{(k)} * w^{(k)}}}, \quad z^{(k)} = \frac{1}{\zeta_k},$$

where ζ_k is defined by (2). Then $(v^{(k)}, z^{(k)})$ converges to the maximal eigenpair of Q^{sym} .

Step 5. To go back to the original matrix T , denote its maximal eigenpair by $(\lambda_{\max}(T), g_{\max})$. Then we have

$$\lambda_{\max}(T) = m - \lim_{k \rightarrow \infty} z^{(k)}, \quad g_{\max} = \lim_{k \rightarrow \infty} \text{diag}(d)v^{(k)},$$

where $\text{diag}(d)$ is the diagonal matrix having diagonal elements (d_k) :

$$d_0 = 1, \quad d_k = d_{k-1} \sqrt{\frac{a_k}{b_{k-1}}}, \quad k \in E \setminus \{0\}.$$

Alternatively,

$$d_0 = 1, \quad d_k = \sqrt{\prod_{j=1}^k \frac{a_j}{b_{j-1}}}, \quad k \in E \setminus \{0\}.$$

Remark 2 We use the following Thomas algorithm to solve equation (3).

Thomas algorithm ([13]). Given a tridiagonal matrix $T \sim (a_k, -c_k, b_k)$, a constant shift z and a vector v . Define

$$d_i = \begin{cases} \frac{b_0}{z - c_0}, & \text{if } i = 0; \\ \frac{b_i}{z - c_i - a_i d_{i-1}}, & \text{if } i = 1, 2, \dots, N-1. \end{cases}$$

Next, define

$$\xi_i = \begin{cases} \frac{v_0}{c_0 - z}, & \text{if } i = 0; \\ \frac{v_i + a_i \xi_{i-1}}{c_i - z + a_i d_{i-1}}, & \text{if } i = 1, 2, \dots, N. \end{cases}$$

Then, the solution w to the equation

$$(-T - zI)w = v \quad \text{on } E.$$

is given as follows.

$$\begin{cases} w_N = \xi_N, \\ w_i = \xi_i - d_i w_{i+1}, \quad i = N-1, N-2, \dots, 1, 0. \end{cases}$$

To illustrate the power of Algorithm 1, we choose the following typical example, taken from [11; §4] and [18; §3.3].

Example 3 Consider the following matrix on E :

$$Q = \begin{pmatrix} -3 & 2 & & & & \\ 1 & -3 & 2 & & & \\ & 1 & -3 & 2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & 2 \\ & & & & 1 & -3 \end{pmatrix}.$$

We are going to use three algorithms to compute the maximal eigenpair of this matrix Q . The first one is Algorithm 1, the others are also recent. One can see the difference of their effectiveness and then understand a part of the developments of the algorithm.

(a) For the matrix Q , we have $m = 0$. After the preparations of Steps 1 ~ 3 in Algorithm 1, we start the iterations in Step 4. The computing results for different N are given in Table 1.

Table 1. Outputs for different N by Algorithm 1

$N+1$	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$	$z^{(3)}$
8	0.253835	0.33544	0.342107	0.342148
16	0.182046	0.21533	0.219673	0.219732
50	0.171577	0.175993	0.176912	0.176937
100	0.171573	0.172686	0.172934	0.172941
500	0.171573	0.171618	0.171628	
1000	0.171573	0.171584	0.171587	
5000	0.171573	0.171573 (1.597s)		
10000	0.171573	0.171573 (6.578s)		
150000	0.171573	0.171573 (29.160s)		

Here, we mention that the numerical experiments in this article are fulfilled on a PC with Intel(R) Core(TM)i5-5200 CPU @2.20 GHz and 4.00 GB RAM using MATLAB (R2014a).

For this example, with the button 'Run and time' in Matlab, we record the running time needed to get the results for larger N and the time 's' denotes seconds. From Table 1, we see that the six precisely significant digits are achieved with no more than three steps. Actually, when $N \geq 4500$, up to six precisely significant digits, the initial $z^{(0)}$ already coincides with λ_0 .

(b) For comparison, as in [11; Algorithm 27], we take $z^{(k)} = \delta_k^{-1}$ instead of $z^{(k)} = \zeta_k^{-1}$ in Algorithm 1 to compute the same example, where

$$\delta_k = \sup_{0 \leq n \leq N} \frac{1}{v_n^{(k)}} \left[\Phi_n \sum_{0 \leq i \leq n} v_i^{(k)} \sqrt{M_{in}} + \sum_{n+1 \leq j \leq N} \sqrt{M_{nj}} \Phi_j v_j^{(k)} \right], \quad k \geq 0, \sum_{\emptyset} := 0.$$

The computing results are given in Table 2.

Table 2. Outputs for different N using δ_k instead of ζ_k

$N+1$	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$	$z^{(3)}$
8	0.304256	0.340851	0.342146	0.342148
16	0.195163	0.217878	0.219722	0.219732
50	0.171632	0.17606	0.176916	0.176937
100	0.171573	0.17269	0.172934	0.172941
500	0.171573	0.171618	0.171628	
1000	0.171573	0.171584	0.171587	
5000	0.171573	0.171573 (2.814s)		
10000	0.171573	0.171573 (15.927s)		
150000	0.171573	0.171573 (96.483s)		

Comparing Table 1 with Table 2, we know that both the use of ζ_k and δ_k need no more than three steps to get the expected results. But the use of ζ_k saves much time. The reason is clear: ζ_k uses a single summation and δ_k uses a double summation. Their computation complexity are $O(N)$ and $O(N^2)$, respectively.

(c) We now compare the use of Q^{sym} in Step 3 of Algorithm 1 with the earlier algorithm presented in [9; §A.4]. The computing results are given in Table 3.

Table 3. Outputs using the algorithm in [9; §A.4]

$N+1$	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$	$z^{(3)}$
8	0.304256	0.340851	0.342146	0.342148
16	0.195163	0.217878	0.219722	0.219732
50	0.171632	0.17606	0.176916	0.176937
100	0.171573	0.17269	0.172934	0.172941
500	0.171573	0.171618	0.171628	
1000	0.171573	0.171584	0.171587	
1023	0.171573	0.171584	0.171586	

Actually, as the problem mentioned in [11; §4], the curve of the initial vector $w^{(0)}$ goes down rapidly at the beginning smaller intervals, and become too small at the end intervals, due to the limitation of computer for calculation accuracy, we can only get the expected results up to 1023 with the algorithm in [9; §A.4].

(d) It is the position to compare Algorithm 1 with the known ones. As well known, the maximal eigenpair has a very wide applications, such as Google's PageRank, the input-output method in economic optimization. Refer to [7-10] for more information. From Wikipedia or textbooks, we have learnt that there are mainly two algorithms for computing the maximal eigenpair: Power Iteration and Rayleigh Quotient Iteration. The former one is the simplest one and has a wide application. But its convergence speed is very slow and hence is almost impractical. The second one is a cubic algorithm, provided the initials are sharp enough. However, it is a dangerous algorithm, as will be discussed in detail in §4. A related algorithm is the famous QR Iteration, which is used to compute the eigensystem of a matrix. This algorithm is widely used in practice. The price is the limitation of the scale of the matrix. As a illustration, we now compare Algorithm 1 with the function 'eig' contained in Matlab. Table 4 presents the different time needed to get the eigenvalues for different N in Example 3.

Table 4. Outputs and time needed using Algorithm 1 and eig

$N + 1$	Algorithm 1	eig
100	-0.172941 (0.271s)	-0.172941 (0.193s)
500	-0.171628 (0.239s)	-0.172726 (0.166s)
550	-0.171587 (0.289s)	-5.85152 (0.570s)

The function 'eig' is mature for computing all the eigenvalues of a matrix. From Table 4, we know that when $N \geq 500$, the outputs are incorrect. Thus, 'eig' is efficient for small matrix but it is not stable for large matrices.

In formula (3) of Algorithm 1, we have used a standard dual technique: replacing Q^{sym} by $-Q^{\text{sym}}$ and then $-\lambda_{\max}(Q^{\text{sym}}) = \lambda_{\min}(-Q^{\text{sym}})$. In general, for each constant $r \neq 0$ and a linear operator L , we have

$$\text{Sp}(rL) = r\text{Sp}(L), \quad \text{Sp} := \text{Spectrum}.$$

If moreover, r is real and L is self-adjoint on some inner product space, then so is rL and we again have the identity of their spectrums. Certainly, such a constant transform does not change the corresponding eigenvectors. For the present tridiagonal case $T \sim (a_k, -c_k, b_k)$, if $r < 0$, we can say a little more. Without loss of generality, assume that $r = -1$.

Remark 4 Let $T \sim (a_k, -c_k, b_k)$ and define $T^\sim \sim (a_k, c_k, b_k)$. Then

$$\text{Sp}(-T) = \text{Sp}(T^\sim),$$

due to the fact that $-T$ and $T^\sim$ here have the same $\tilde{Q}$ defined in Algorithm 1 and hence have the same spectrum.

From [7-11], one sees the successive modifications to achieve Algorithm 1. Based on the sharp estimates of maximal eigenvalue [4], the efficient initials were introduced in [7; §3] for Rayleigh Quotient Iteration. To avoid the dangerous region, the modified algorithm by redefining $z_k = \delta_k^{-1}$ was presented in [9; §A.4]. In the following-up article [10; §2], an explicit representation of the solution to tridiagonal equation was proposed. To balance the sharpness and the complexity, several methods to improve the algorithm were proposed in [11]. We will outline this trip in Section 4. Besides, based on [7, 9, 10], Tang and Yang [18] simplified the computational complexity and proved that the total cost for computing is $O(N)$. In the next section, we introduce additional examples to illustrate the power of the algorithm.

2 Examples

In this section, the examples are taken from the papers [4, 7, 9, 11], except Example 6 which is newly added. To compare the results with Algorithm 1, let us look at the following example taken from [9; Appendix A.4] first.

Example 5 ([9; Example A3]) Let

$$T = \begin{pmatrix} 2.334 & 0.9962 & & & & \\ 0.5142 & 2.6725 & 0.1111 & & & \\ & 0.2115 & 2.263 & 0.1405 & & \\ & & 0.8442 & 2.8457 & 0.7595 & \\ & & & 0.2347 & 2.2257 & 0.0781 \\ & & & & 0.9837 & 2.1582 \end{pmatrix}.$$

Then the eigenvalues of T are

$$3.26753, 3.16247, 2.40182, 2.12632, 1.80416, 1.73679.$$

For the matrix T , we have $m = 4.4494$. The outputs using Algorithm 1 are given in Table 5.

Table 5. The outputs by Algorithm 1

$m - z^{(0)}$	$m - z^{(1)}$	$m - z^{(2)}$	$m - z^{(3)}$	$m - z^{(4)}$
3.41401	3.28721	3.26957	3.26757	3.26753

Comparing the results here with those in [9; Example A3], we know that Algorithm 1 is rather effective. In the next two examples, we compute the maximal eigenvalue of complex matrices using Algorithm 1.

Example 6 Let

$$T = \begin{pmatrix} -2 & 2+i & & & \\ 2^2(2-i) & -1 & 3+i & & \\ & 2^2(3-i) & -3 & 1+2i & \\ & & 1-2i & -2 & 2+3i \\ & & & 2-3i & -4 & 2-i \\ & & & & 4+2i & 3 \end{pmatrix}.$$

Then the eigenvalues of T are

$$-10.0244, -7.26296, -2.65371, 0.243075, 4.50158, 6.19640.$$

For the complex matrix T , we have $m = \sqrt{5} + 4\sqrt{10} - 3$. The outputs using Algorithm 1 are given in Table 6.

Table 6. The outputs by Algorithm 1

$m - z^{(0)}$	$m - z^{(1)}$	$m - z^{(2)}$	$m - z^{(3)}$	$m - z^{(4)}$
7.31593	6.34282	6.20382	6.19644	6.19640

Example 7 ([11; Example 19]) Let

$$T = \begin{pmatrix} -1 & 2+i & & & \\ 2^2(2-i) & -1 & 2^4(2+i) & & \\ & 6^2(2-i) & -1 & 3^4(2+i) & \\ & & 12^2(2-i) & -1 & 4^4(2+i) \\ & & & 20^2(2-i) & -1 \end{pmatrix}.$$

Then the eigenvalues of T are

$$-756.391, -52.0308, -1, 50.0308, 754.391.$$

For the complex matrix T , we have $m = 400\sqrt{5} - 1$. The outputs using Algorithm 1 are given in Table 7.

Table 7. The outputs by Algorithm 1

$m - z^{(0)}$	$m - z^{(1)}$	$m - z^{(2)}$
773.836	754.594	754.391

To explicitly illustrate the power of the algorithm, we compute three real examples using $z^{(k)} = \zeta_k^{-1}$ and $z^{(k)} = \delta_k^{-1}$, respectively, in Algorithm 1. The following three examples are birth-death matrices taken from [4; §3] satisfying that $c_0 = b_0$ and $c_k = a_k + b_k$, $k \in E \setminus \{0\}$. Which means $m = 0$.

Example 8 ([4; Example 3.5]) Let $b_k = 2$ ($k \geq 0$), $a_k = k$ ($k \geq 1$). Then for this example, when N is large enough, $\lambda_0 = 1$. For different N , the outputs are given in Tables 8 and 9, using ζ_k and δ_k , respectively.

Table 8. The outputs for different N using ζ_k

$N+1$	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$	$z^{(3)}$	$z^{(4)}$
8	0.836045	0.992135	1.01307	1.01355	
16	0.834044	0.977037	0.999391	1.00011	
32	0.83404	0.976651	0.999245	0.999999	1.0
50	0.83404	0.97665	0.999245	0.999999	1.0
100	0.83404	0.97665	0.999245	0.999999	1.0

Table 9. The outputs for different N using δ_k instead of ζ_k

$N+1$	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$	$z^{(3)}$
8	0.943632	1.00843	1.01353	1.01355
16	0.918635	0.99231	1.00004	1.00011
32	0.917966	0.99197	0.999921	1.0
50	0.917965	0.991969	0.999921	1.0
100	0.917965	0.991969	0.999921	1.0

Comparing Table 8 with Table 9, we know that when $N \geq 32$, the use of ζ_k needs one more step than that of δ_k to get the expected results. But usually, the time spent in Table 8 is shorter than that of Table 9. See for instance in the next example.

Example 9 ([4; Example 3.7]) Let

$$b_k = (k+1)^4, \quad a_k = k(k-1/2)(k^2 + 3k + 3),$$

when N is large enough, $\lambda_0 = 1/2$. For different N , the outputs are given in Tables 10 and 11, using ζ_k and δ_k , respectively.

Table 10. The outputs for different N using ζ_k

$N+1$	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$	$z^{(3)}$
8	0.416918	0.627353	0.633461	0.633466
50	0.35708	0.541483	0.548162	0.548169
100	0.347301	0.526674	0.533345	0.533353
500	0.335057	0.507877	0.514489	0.514498
1000	0.332284	0.503583	0.510173	0.510182
5000	0.328655	0.497945	0.504504	0.504512 (2.009s)
10000	0.327808	0.496625	0.503175	0.503184 (8.973s)

Table 11. The outputs for different N using δ_k

$N+1$	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$
8	0.616418	0.63343	0.633466
50	0.530028	0.548113	0.548169
100	0.515265	0.533295	0.533353
500	0.496589	0.514436	0.514498
1000	0.492332	0.51012	0.510182
5000	0.486749	0.50445	0.504512 (3.383s)
10000	0.485443	0.503121	0.503184 (24.602s)

For this example, we record the time needed to get the results for larger N . To get the expected results, from Tables 10 and 11, it follows that the use of ζ_k needs one more step than that of δ_k , but saves much time. The next example exhibits the same phenomenon.

Example 10 ([4; Example 3.6] and [7; Example 7]) Let

$$b_k = (k+1)^2, \quad a_k = k^2,$$

when N is large enough, $\lambda_0 = 1/4$. For different N , the outputs are given in Tables 12 and 13, using ζ_k and δ_k , respectively.

Table 12. The outputs for different N using ζ_k

$N+1$	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$	$z^{(3)}$
8	0.406762	0.514094	0.525176	0.525268
100	0.304993	0.36995	0.376269	0.376383
500	0.279999	0.333226	0.33823	0.338329
1000	0.273336	0.322412	0.327148	0.32724
5000	0.26322	0.304128	0.308454	0.308529 (2.088s)
7500	0.261484	0.300603	0.304845	0.304918 (4.711s)
10000	0.260397	0.298305	0.302489	0.302561 (8.516s)

Table 13. The outputs for different N using δ_k

$N+1$	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$	$z^{(3)}$
8	0.485985	0.52415	0.525267	0.525268
100	0.348549	0.374848	0.376378	0.376383
500	0.310195	0.33686	0.33832	0.338329
1000	0.299089	0.325735	0.327229	0.32724
5000	0.281156	0.306874	0.308514	0.308529 (4.281s)
7500	0.277865	0.303213	0.304903	0.304918 (11.077s)
10000	0.275762	0.300821	0.302545	0.302561 (55.355s)

Examples 8, 9 and 10 illustrate the reason for choosing $z^{(k)} = 1/\zeta_k$ from the computational point of view. To conclude this section, we study one more example.

Example 11 ([7; Example 22]) Let

$$T = \begin{pmatrix} -5 & 5 & & & \\ 3 & -7 & 4 & & \\ & 2 & -3 & 1 & \\ & & 10 & -16 & 6 \\ & & & 11 & -11 - b_4 \end{pmatrix}.$$

For the matrix T , we have $m = 0$. For different N , the outputs using Algorithm 1 are given in Table 14.

Table 14. The outputs for different N

b_4	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$
0.01	0.000143394	0.000278683	0.000278686
1	0.0130396	0.0244922	0.0245175
100	0.102368	0.182367	0.182819
10^6	0.109962	0.19468	0.195145

The number of iterations is the same as those given in [7], except when $b_4 = 0.01$, for which the use of Algorithm 1 requires one more step.

From the above examples, it follows that Algorithm 1 is efficient for computing the maximal eigenpair. Furthermore, there is a natural way to study the next to the maximal eigenpair, which is the topic in the next section.

3 Conservative case

In this section, we continue the study on a special case in Algorithm 1. That is $\tilde{c}_N = \tilde{a}_N$. Then the matrix $\tilde{Q}$ constructed in Algorithm 1 is conservative and has a trivial eigenvalue 0. The aim of this section is to study its sub-maximal eigenpair. By using a shift mI , if necessary, we deal with the tridiagonal Q -matrix of the following form:

$$Q = \begin{pmatrix} -c_0 & b_0 & & & \\ a_1 & -c_1 & b_1 & & \\ & a_2 & -c_2 & b_2 & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots & b_{N-1} \\ & & & & a_N & -c_N \end{pmatrix}, \quad (4)$$

where $a_k > 0$ ($1 \leq k \leq N$), $b_k > 0$ ($0 \leq k < N$) and $c_k = a_k + b_k$ (with $a_0 := 0$ and $b_N := 0$). Clearly, the maximal eigenvalue for this matrix is $\lambda_0 = 0$ with constant eigenvector $\mathbb{1}$. Now, we compute the sub-maximal eigenvalue $\lambda_1(Q)$ by the following algorithm, which is essentially harder but formally the

same as Algorithm 1, except the use of a new auxiliary tridiagonal matrix $\tilde{Q}$ on $E_1 := \{k \in \mathbb{Z}_+ : 1 \leq k \leq N\}$. Recall that $E = E_1 \cup \{0\}$. The next result is due to [11; §4].

Algorithm 12 Suppose that Q is a tridiagonal matrix of form (4) with an auxiliary condition that $c_k = a_k + b_k$, $k \in E$. Define a measure μ corresponding to Q as follows:

$$\mu_0 = 1, \quad \mu_n = \mu_{n-1} \frac{b_{n-1}}{a_n}, \quad 1 \leq n \leq N. \quad (5)$$

Before computing the sub-maximal eigenpair of Q , we need prepare Steps 1 ~ 3.

Step 1. Define $\tilde{Q} \sim (\tilde{a}_k, -\tilde{c}_k, \tilde{b}_k)$ on E_1 as follows. First, set $\tilde{b}_1 = a_1 + b_0$. Next, define

$$\begin{cases} \tilde{b}_k = a_k + b_{k-1} - \frac{a_{k-1}b_{k-1}}{\tilde{b}_{k-1}}, & \tilde{a}_k = a_k + b_{k-1} - \tilde{b}_k, & 2 \leq k < N; \\ \tilde{c}_k = a_k + b_{k-1}, & & 1 \leq k \leq N; \\ \tilde{a}_N = \frac{a_{N-1}b_{N-1}}{\tilde{b}_{N-1}}. \end{cases}$$

This is essentially different from Step 1 in Algorithm 1, in fact it uses a dual technique. The other Steps are similar to those in Algorithm 1.

Step 2. Define the tridiagonal matrix $Q^{\text{sym}} \sim (a_k^{\text{sym}}, -c_k^{\text{sym}}, b_k^{\text{sym}})$ on E_1 :

$$\begin{aligned} c_k^{\text{sym}} &= \tilde{c}_k, & k \in E_1; \\ a_k^{\text{sym}} &= b_{k-1}^{\text{sym}} = \sqrt{a_{k-1}b_{k-1}}, & k \in E_1 \setminus \{1\}. \end{aligned}$$

Step 3. Define the upper triangular matrix (M_{kj}) and vector (Φ_k) on E_1 :

$$\begin{aligned} M_{kk} &= 1, \quad M_{kj} = M_{k,j-1} \frac{\tilde{a}_j}{\tilde{b}_{j-1}} \left[= \frac{\tilde{a}_{k+1} \cdots \tilde{a}_j}{\tilde{b}_k \cdots \tilde{b}_{j-1}} \right], & 2 \leq k+1 \leq j \leq N; \\ \Phi_k &= \frac{1}{\tilde{b}_k} + \sum_{k+1 \leq j \leq N} \frac{\tilde{a}_{k+1} \cdots \tilde{a}_j}{\tilde{b}_k \cdots \tilde{b}_j} = \sum_{k \leq j < N+1} \frac{M_{kj}}{\tilde{b}_j}, & 1 \leq k \leq N. \end{aligned}$$

With $(\tilde{a}_k, \tilde{b}_k)$, Q^{sym} , M and Φ at hand, we can now start our iterations.

Step 4. For given $v^{(k)}$ ($k \geq 0$), define

$$\zeta_k = \sup_{1 \leq n \leq N} \frac{1}{\sqrt{\tilde{b}_n v_n^{(k)} - \sqrt{\tilde{a}_{n+1} v_{n+1}^{(k)}}}} \sum_{j=1}^n v_j^{(k)} \sqrt{\frac{M_{jn}}{\tilde{b}_n}}, \quad k \geq 0. \quad (6)$$

with a convention that $\tilde{a}_{N+1} = 0$. As in [11; Algorithm 29], choose $w_i^{(0)} = \sqrt{\Phi_i}$, $i \in E_1$. Define

$$v^{(0)} = w^{(0)} / \sqrt{w^{(0)*} w^{(0)}}, \quad z^{(0)} = \zeta_0^{-1},$$

where ζ_0 is defined by (6) with $k = 0$. For each $k \geq 1$, solve $w^{(k)}$:

$$(-Q^{\text{sym}} - z^{(k-1)}I)w^{(k)} = v^{(k-1)} \quad \text{on } E_1, \quad (7)$$

and define

$$v^{(k)} = \frac{w^{(k)}}{\sqrt{w^{(k)} * w^{(k)}}}, \quad z^{(k)} = \frac{1}{\zeta_k},$$

where ζ_k is defined by (6). Let $(\lambda_1^{\text{sym}}, g^{\text{sym}})$ be the minimal eigenpair of $-Q^{\text{sym}}$. Then

$$\lambda_1^{\text{sym}} = \lim_{k \rightarrow \infty} z^{(k)}, \quad g^{\text{sym}} = \lim_{k \rightarrow \infty} v^{(k)}.$$

Furthermore, denote the sub-maximal eigenpair of Q by $(-\lambda_1, f)$, then

$$\lambda_1 = \lambda_1^{\text{sym}} = \lim_{k \rightarrow \infty} z^{(k)}, \quad f = \mathcal{M}^{-1}g = \lim_{k \rightarrow \infty} (\mathcal{M}^{-1}g^{(k)}),$$

where $\mathcal{M}$ is a matrix on $E \times E$ (recall that $E = E_1 \cup \{0\}$) of the following form:

$$\mathcal{M} = \begin{pmatrix} \mu_0 & \mu_1 & \mu_2 & \cdots & \mu_N \\ & \mu_1 & \mu_2 & \cdots & \mu_N \\ & & \mu_2 & \cdots & \mu_N \\ & & & \ddots & \vdots \\ 0 & & & & \mu_N \end{pmatrix},$$

μ is defined by (5), g and $(g^{(k)})$ are vectors on E satisfying that

$$g_0 = 0, \quad g|_{E_1} = \text{diag}(d1)g^{\text{sym}},$$

and

$$g_0^{(k)} = 0, \quad g^{(k)}|_{E_1} = \text{diag}(d1)v^{(k)}.$$

Here, $\text{diag}(d1)$ is a diagonal matrix on E_1 having diagonal elements $(d1_k)$:

$$d1_1 = 1, \quad d1_k = d1_{k-1} \sqrt{\frac{b_{k-1}}{a_{k-1}}}, \quad k \in E_1 \setminus \{1\},$$

Alternatively,

$$d1_1 = 1, \quad d1_k = \sqrt{\prod_{j=2}^k \frac{b_{j-1}}{a_{j-1}}}, \quad k \in E_1 \setminus \{1\}.$$

Remark 13 We also use Thomas algorithm to solve equation (7).

For the remainder of this section, we introduce some examples to illustrate the power of Algorithm 12. The next two examples are closely related to Examples 3 and 10, respectively.

Example 14 Consider the matrix

$$Q = \begin{pmatrix} -2 & 2 & & & \\ 1 & -3 & 2 & & \\ & 1 & -3 & 2 & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & -3 & 2 \\ & & & & 1 & -1 \end{pmatrix}.$$

The computing results for different N using Algorithm 12 are given in Table 15.

Table 15. The outputs for different N by Algorithm 12

$N+1$	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$	$z^{(3)}$
8	0.284681	0.37967	0.386839	0.386874
16	0.184834	0.221395	0.225864	0.225920
50	0.171578	0.176176	0.177128	0.177154
100	0.171573	0.172709	0.172961	0.172969

Example 15 ([7; Example 25]) Consider the matrix

$$Q = \begin{pmatrix} -1 & 1 & & & \\ 1 & -5 & 2^2 & & \\ & 2^2 & -13 & 3^2 & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & -N^2 - (N-1)^2 & N^2 \\ & & & & N^2 & -N^2 \end{pmatrix}.$$

For different N , the outputs using Algorithm 12 are given in Table 16.

Table 16. The outputs for different N by Algorithm 12

$N+1$	$z^{(0)}$	$z^{(1)}$	$z^{(2)}$	$z^{(3)}$
8	0.60449	0.80318	0.820402	0.820539
16	0.481611	0.638168	0.650021	0.650141
50	0.379585	0.494848	0.5035	0.503596
100	0.344814	0.444154	0.451895	0.451977

Example 16 ([7; Example 26]) Let

$$Q = \begin{pmatrix} -5 & 5 & & & \\ 3 & -7 & 4 & & \\ & 2 & -3 & 1 & \\ & & 10 & -16 & 6 \\ & & & 11 & -11 \end{pmatrix}.$$

The eigenvalues of $-Q$ are as follows:

$$22.348, \quad 10.6857, \quad 5.92951, \quad 3.03673, \quad 0.$$

The outputs using Algorithm 12 are given in Table 17.

Table 17. The outputs by Algorithm 12

$z^{(0)}$	$z^{(1)}$	$z^{(2)}$	$z^{(3)}$
2.4898	2.97585	3.03569	3.03673

Comparing the results here with the outputs given in [7], we need one more iteration in using Algorithm 12. This often happens for small N . The reason we do not go to large scale matrices is that Algorithm 12 should be as powerful as Algorithm 1, since they have the same computation complexity $O(N)$.

The story of the development as well as intrinsic points are presented in the next section.

4 Development and proofs

Let us begin this section with the well-known Rayleigh Quotient Iteration.

Rayleigh Quotient Iteration (abreav. **RQI**) For a given real matrix A with nonnegative off-diagonal elements on $E \times E$, let $(\lambda_{\max}, g_{\max}(A))$ be the maximal eigenpair of A and $(z^{(0)}, v^{(0)})$ be an approximation of $(\lambda_{\max}, g_{\max}(A))$. At the k th step ($k \geq 1$), solve the linear equation in $w^{(k)}$:

$$(z^{(k-1)}I - A)w^{(k)} = v^{(k-1)},$$

where I is the identity matrix and define

$$v^{(k)} = \frac{w^{(k)}}{\sqrt{w^{(k)*}w^{(k)}}}, \quad z^{(k)} = v^{(k)*}Av^{(k)}.$$

Then $v^{(k)}$ converges to $g_{\max}$ and $z^{(k)}$ converges to $\lambda_{\max}(A)$ as $k \rightarrow \infty$, provided $(z^{(0)}, v^{(0)})$ is close enough to $(\lambda_{\max}(A), g_{\max})$.

In what follows, we will often use some probabilistic idea. Our first object is the Q -matrix:

$$Q = (q_{ij} : i, j \in E),$$

where $q_{ij} \geq 0$ for every pair $i \neq j$ and $\sum_{j \in E} q_{ij} \leq 0$ for every $i \in E$. For simplicity, we assume at the moment that

$$0 < \lambda_0 < |\lambda_1| \leq |\lambda_2| \leq \cdots,$$

where $\{\lambda_j\}$ is the sequence of the eigenvalues of $-Q$. Suppose that the orthogonal eigenvectors corresponding to $\{\lambda_j\}$ are $\{g_j\}$. We now introduce RQI for Q -matrix.

RQI for Q-matrix The algorithm is almost the same as the original one except A is replaced by $-Q$, at the same time, the original iteration equation is replaced by

$$(-Q - z^{(k-1)}I)w^{(k)} = v^{(k-1)}.$$

We know that Rayleigh Quotient Iteration (RQI) is fast for computing the maximal eigenpair but can be dangerous because there are many pitfalls. To get the maximal eigenpair, there is a strict restriction for the initial $(z^{(0)}, v^{(0)})$. Fortunately, some basic analytic estimates of the maximal eigenvalue and some mimic of its eigenvector were obtained for many years of accumulation in the study of stochastic stability speed. See [3, 4] for instance. Based on this, Algorithm 1 has been developed in [7-11], and then Algorithm 12 is presented in [11]. Fig.1 exhibits the diagram of the development from RQI to Algorithm 1.

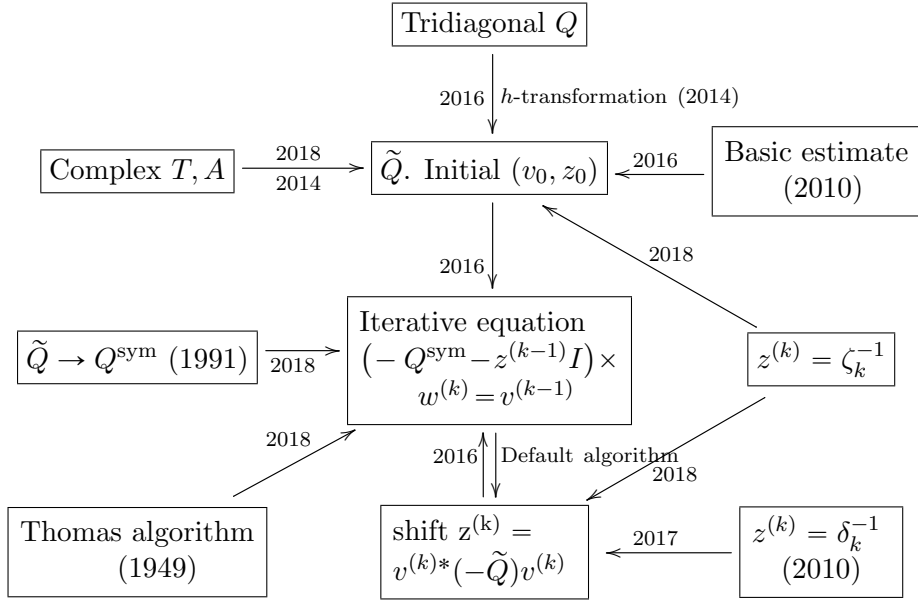


Fig.1 Diagram of development of Algorithm 1.

For the remainder of this section, we are going to explain this diagram by six steps:

- Isospectral operators. $Q \xrightarrow{2014} \tilde{Q}, T \xrightarrow{2018} \tilde{Q}$.
- Initial (v_0, z_0) .
- Rayleigh quotient $\rightarrow \delta_k^{-1}$.
- Coupling of non-symmetric Q and symmetrized matrix Q^{sym} .
- Computation complexity.
- The Thomas algorithm.

4.1 Isospectral operators. $Q \xrightarrow{2014} \tilde{Q}, T \xrightarrow{2018} \tilde{Q}$.

Before moving further, let us recall the so-called birth-death Q -matrix.

Definition 17 The matrix $Q \sim (a_k, -c_k, b_k)$ is called birth-death Q -matrix on E if

$$a_k > 0, \quad b_k > 0, \quad c_k \geq a_k + b_k.$$

Let T be an Hermitizable tridiagonal matrix of form (1), and let $\tilde{Q}$ be the one defined by Algorithm 1. Next, we recall that an isospectral transformation (h -transformation) was introduced in [6] which allows us to study the Q -matrix having arbitrary diagonals. Then, an explicit construction of isospectral transformation in tridiagonal case (3 steps) was presented in [5], which is the h -transformation used in [7].

In the study on the sub-maximal real part of eigenvalues, a question appears: when a complex matrix has real spectrum? This triggered the study on Hermitizable complex matrix [11; §2]. In which the direct explicit representation of isospectral transformation in tridiagonal case is deduced [11; §3]. This completes the proof of $T \rightarrow \tilde{Q}$, that is quite hard. Furthermore, in terms of a modified Householder transformation, it can be proved that a general Hermitizable complex matrix is also isospectral to a birth-death matrix $\tilde{Q}$ [11; Theorem 24]. For tridiagonal matrix, the transformation can be explained roughly by Theorem 18.

Theorem 18 Assume T is a complex matrix of form (1). Set $A = T - mI$, where $m = \sup_{k \in E} (-c_k + |a_k| + |b_k|)^+$. Let h be A -harmonic on $E \setminus \{N\}$ with $h_0 = 1$, i.e.,

$$(Ah)(k) = 0, \quad k \in \{0, 1, \dots, N-1\}.$$

Define $\tilde{Q} = \text{diag}(h)^{-1} A \text{diag}(h)$, where $\text{diag}(h)$ is the diagonal matrix having diagonal elements (h_k) , then $\tilde{Q}$ is a birth-death matrix of the form $\tilde{Q} \sim (\tilde{a}_k, -\tilde{c}_k, \tilde{b}_k)$ as in Step 1 of Algorithm 1.

Proof. Since $\tilde{Q} = \text{diag}(h)^{-1} A \text{diag}(h)$, we have

$$\tilde{c}_k = c_k + m, \quad \tilde{a}_k = \frac{h_{k-1}}{h_k} a_k, \quad \tilde{b}_k = \frac{h_{k+1}}{h_k} b_k. \quad (8)$$

According to

$$(Ah)(k) = 0, \quad k \in \{0, 1, \dots, N-1\},$$

we get

$$a_k \frac{h_{k-1}}{h_k} = \tilde{c}_k - b_k \frac{h_{k+1}}{h_k}.$$

Combining this with (8), we have

$$\text{conservativity : } \tilde{a}_k = \tilde{c}_k - \tilde{b}_k, \quad 1 \leq k < N,$$

and

$$\text{invariance : } \tilde{a}_k \tilde{b}_{k-1} = a_k b_{k-1} = |a_k b_{k-1}| = u_k, \quad 1 \leq k \leq N,$$

where u is the one given in Algorithm 1. Thus,

$$\tilde{b}_0 = \tilde{c}_0, \quad \tilde{b}_k = \tilde{c}_k - a_k \frac{b_{k-1}}{\tilde{b}_{k-1}} = \tilde{c}_k - \frac{u_k}{\tilde{b}_{k-1}}, \quad 1 \leq k \leq N,$$

and

$$\tilde{a}_N = \frac{u_N}{\tilde{b}_{N-1}}.$$

Besides, according to [11; Theorem 15], the sequences $\{\tilde{a}_k\}$ and $\{\tilde{b}_k\}$ are positive, provided $m < \infty$. $\square$

Remark 19 For the matrix $\tilde{Q}$ in Theorem 18, the sum of each row equals zero, except the N th row which is not positive.

Now, for a given Hermitizable tridiagonal complex matrix T , in terms of Step 1, we can transform its spectrum to the one for a birth-death Q -matrix $\tilde{Q}$. After completing this procedure, we need only to deal with the maximal eigenpair computation for the birth-death Q -matrix $\tilde{Q}$. In what follows, unless otherwise stated, we omit ' $\sim$ ' in the notation of $\tilde{Q}$, $\tilde{a}_k$, $\tilde{c}_k$, $\tilde{b}_k$. About the study on this topic, there is a long accumulation on the estimation of stability speed. For example, in [4], we have the following result

$$\delta_k^{-1} \uparrow \leq \lambda_0 \leq \delta_k'^{-1},$$

where

$$\delta_k = \sup_{0 \leq n \leq N} \frac{1}{v_n^{(k)}} \left[\varphi_n \sum_{0 \leq i \leq n} \mu_i v_i^{(k)} + \sum_{n+1 \leq j \leq N} \mu_j \varphi_j v_j^{(k)} \right], \quad k \geq 0, \quad (9)$$

and $\varphi_i = \sum_{j=i}^N (\mu_j b_j)^{-1}$, $i \in E$.

4.2 Initial (v_0, z_0) .

The choose of initial (v_0, z_0) plays rather important role for RQI. Since for larger matrices, there are a lot of eigenvalues, for which the algorithm may go to pitfalls. In [7], we choose $z_0 = \delta_1^{-1}$,

$$v_0 = \frac{w_0}{\|w_0\|}, \quad w_0 = \sqrt{\varphi}.$$

Here, (v_0, z_0) is formally different from the representation of that in Algorithm 1, which will be explained later. Based on the estimation of (9), the initial is already good enough so that, as in [7], the shift z_k is setting to be the Rayleigh quotient in latter iteration:

$$z_k = v^{(k)*} (-Q) v^{(k)}, \quad k \geq 1.$$

4.3 Rayleigh quotient $\rightarrow \delta_k^{-1}$.

RQI can be used in many cases due to its simplicity, the danger going into pitfalls is its limitation. Consider the symmetrizable Q for example, we usually have

$$v^{(k)*}(-Q)v^{(k)} \geq \lambda_0.$$

There may be an eigenvalue $\lambda' (> \lambda_0)$ nearer $v^{(k)*}(-Q)v^{(k)}$ than λ_0 , then the algorithm will converge to $\lambda' \neq \lambda_0$, i.e. falling into the pitfall λ' . Based on this, in [9, 10], we suggested replacing Rayleigh quotient $v^{(k)*}(-Q)v^{(k)}$ by δ_n^{-1} . Since there is no other eigenvalues of $-Q$ in the interval $(0, \lambda_0^{-1})$, by basic estimates, our modified algorithm becomes safe, never falling into a pitfall.

4.4 Coupling of non-symmetric Q and symmetrized matrix Q^{sym} .

In [18] and [11; §3], the computational trouble caused by non-symmetric matrix was illustrated in detail. On one hand, in non-symmetric case, the eigenvector we concerned increases (or decreases) very fast, which is the problem that the computer cannot deal with. On the other hand, if we replace Q by Q^{sym} , the spectrum remains, but it happens often that the sum of some of the first N rows is not zero, which makes the use of $\{\delta_k\}$ and $w^{(0)}$ meaningless. If we now use the isospectral transformation in Section 4.1 on Q^{sym} , then we return to the non-symmetric Q . This leads to an unsolvable circulation. To be brief, the advantage of Q is that for which we have good estimates δ_k and $w^{(0)}$, and the advantage of Q^{sym} is that for which there are many mature algorithms. However, these two are not compatible. This hard problem troubled us for many years. The method we used here is to “let the two get married”, that means to use their two advantages simultaneously. For mathematical details, see [11; §4]. Clearly, this coupling technique should be meaningful for other algorithms of non-symmetric and symmetric ones.

In order to go back to the original matrix T , let $(\lambda_{\max}(T), g_{\max})$ denote its maximal eigenpair and (z, v) be the output from the last iteration in Step 4. Then we have

$$\lambda_{\max}(T) \approx m - z, \quad g_{\max} \approx \text{diag}(h)\text{diag}(\tilde{\mu})^{-1/2}v.$$

By (8), $h_k/h_{k-1} = \tilde{b}_{k-1}/b_{k-1}$, combining this with $h_0 = 1$, we have

$$h_k = h_{k-1} \frac{\tilde{b}_{k-1}}{b_{k-1}} = \frac{\tilde{b}_0 \tilde{b}_1 \cdots \tilde{b}_{k-1}}{b_0 b_1 \cdots b_{k-1}}.$$

By the definition of $\tilde{\mu}$, we have

$$\tilde{\mu}_k = \frac{\tilde{b}_0 \tilde{b}_1 \cdots \tilde{b}_{k-1}}{\tilde{a}_1 \tilde{a}_2 \cdots \tilde{a}_k}.$$

Hence, for $k \geq 1$, we have

$$\frac{h_k}{\sqrt{\tilde{\mu}_k}} = \frac{\tilde{b}_0 \tilde{b}_1 \cdots \tilde{b}_{k-1}}{b_0 b_1 \cdots b_{k-1}} \sqrt{\frac{\tilde{a}_1 \tilde{a}_2 \cdots \tilde{a}_k}{\tilde{b}_0 \tilde{b}_1 \cdots \tilde{b}_{k-1}}} = \frac{\sqrt{u_1 u_2 \cdots u_k}}{b_0 b_1 \cdots b_{k-1}} = \sqrt{\prod_{j=1}^k \frac{a_j}{b_{j-1}}}.$$

Then, we get the last assertion of Algorithm 1.

4.5 Computation Complexity

Two improvements based on the computation complexity are made to arrive at Algorithm 1. The first one is the use of the matrix M and the vector Φ stated in Step 3 of Algorithm 1, which are used to leveling the quantities we required and then fasten the computations.

The second change is replacing δ_k in Sections 4.2 and 4.3 by ζ_k appeared in Step 4 of the algorithm. This is somehow strange since in general, the estimates using ζ_k is poorer than the use of δ_k (refer to [4; (2.22)]):

$$\zeta_k^{-1} \leq \delta_k^{-1} \leq \lambda_0.$$

Hence, we usually do not use ζ_k to estimate λ_0 (refer to [4; Theorem 3.2] for instance). However, such a replacement is reasonable since in the expression of δ_k , one uses double summation, it has complexity $O(N^2)$, but ζ_k uses only a single summation, having complexity $O(N)$. Therefore, the use of ζ_k saves much computation time especially for large N .

4.6 The Thomas algorithm

The algorithm goes back to L.H. Thomas (1949), refer to [16] for a short history. The algorithm is designed especially for tridiagonal systems. It is famous since it is an unusual $O(N)$ -algorithm (see [12]). Nevertheless, it is not surprising that the algorithm may not be stable, such as when the matrix is symmetric positive definite, as pointed in [13]. Very fortunately, even though our Q^{sym} is symmetric positive definite, but we can still use the algorithm without trouble since at each step of the iterations, it keeps us in the safe region. See [11] for more details.

4.7 Proof of Algorithm 12

From [11; Proof of Algorithm 29], we have $\hat{Q} = \mathcal{M}Q\mathcal{M}^{-1}$, where

$$\hat{Q} = \begin{pmatrix} 0 & 0 & 0 & & & \\ b_0 & -(a_1 + b_0) & a_1 & & & \\ & b_1 & -(a_2 + b_1) & a_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & a_{N-1} \\ & & & & b_{N-1} & -(a_N + b_{N-1}) \end{pmatrix}.$$

Then $\hat{Q}$ and Q have the same spectrum. By removing the first row and the first column of $\hat{Q}$, we obtain a matrix on E_1 :

$$\hat{Q}_1 = \begin{pmatrix} -(a_1 + b_0) & a_1 & & & \\ b_1 & -(a_2 + b_1) & a_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & a_{N-1} \\ & & & b_{N-1} & -(a_N + b_{N-1}) \end{pmatrix}.$$

Now, the spectrum of Q ignoring the trivial eigenvalue 0 coincides with the spectrum of $\hat{Q}_1$. The sub-minimal eigenvalue of $-Q$ coincides with the minimal eigenvalue of $-\hat{Q}_1$. Applying Algorithm 1 to the matrix $\hat{Q}_1$, we get Algorithm 12. Here, we prove the assertion about the sub-maximal eigenvector in Algorithm 12. Let (z, v^{sym}) be the output from the last iteration in Step 4 of Algorithm 12. Note that we develop Algorithm 12 as follows:

$$\hat{Q}_1 \sim (\hat{a}_k, -\hat{c}_k, \hat{b}_k) \longrightarrow \tilde{Q} \sim (\tilde{a}_k, -\tilde{c}_k, \tilde{b}_k) \longrightarrow Q^{\text{sym}} \sim (a_k^{\text{sym}}, -c_k^{\text{sym}}, b_k^{\text{sym}}).$$

Following the proof given in the last part of §4.4 with

$$\frac{h_k}{\sqrt{\mu_k}} = \frac{\tilde{b}_1 \cdots \tilde{b}_{k-1}}{\tilde{b}_1 \cdots \tilde{b}_{k-1}} \sqrt{\frac{\tilde{a}_2 \cdots \tilde{a}_k}{\tilde{b}_1 \cdots \tilde{b}_{k-1}}} = \sqrt{\frac{b_1 \cdots b_{k-1}}{a_1 \cdots a_{k-1}}}, \quad k \geq 2.$$

it follows that the eigenvector of $\hat{Q}_1$ corresponding to λ_1 is

$$\hat{g} \approx \text{diag}(d1)v^{\text{sym}},$$

where $\text{diag}(d1)$ is a diagonal matrix on E_1 having diagonal elements $(d1_k)$:

$$d1_1 = 1, \quad d1_k = \sqrt{\frac{b_1 \cdots b_{k-1}}{a_1 \cdots a_{k-1}}}, \quad k \in E_1 \setminus \{1\}.$$

Thus, the eigenvector of $\hat{Q}$ corresponding to λ_1 is g :

$$g_0 = 0, \quad g|_{E_1} = \hat{g} \approx \text{diag}(d1)v^{\text{sym}}.$$

Combining with $\hat{Q} = \mathcal{M}Q\mathcal{M}^{-1}$, we get the eigenvector of Q corresponding to λ_1 is $\mathcal{M}^{-1}g$.

To conclude this paper, we make two remarks. First, even though we concentrate on tridiagonal matrix here, there are some ways to extend it to the general one, as did in [7; Appendix of §3 or §4], except the general Hermitizable matrices mentioned above. Next, we also have the global algorithms presented in [9] to deal with general matrices, especially for those having nonnegative off-diagonal elements. Here is a very simple example to show the importance of the last algorithms.

Example 20 Study the maximal eigenpair of the matrix

$$A = (a_{kj}), \quad a_{kj} = (2^{k \wedge j} - 1) 2^{-j}, \quad 1 \leq k, j \leq N.$$

Here we adopt three different approaches.

- (a) By using the package ‘Eigensystem’ in Mathematica (version 10.03 or 11.03) on PC, it works well up to $N = 11$. Starting from $N = 12$, the output of the components of the vector have different sign. Since A is positive, this is clearly impossible by the well-known Perron-Frobenius theorem.
- (b) Next, we use the package ‘eig’ in MatLab, the similar result appears. Starting from $N = 173$, the output of the components of the vector have different sign.
- (c) However, when we use [9; Algorithm 5], it is well done up to $N = 7000$.

We do not go further since, at which, the outputs become stable.

Hopefully, this paper proves the value of the algorithms developed in [7-11].

Acknowledgments Research supported in part by National Natural Science Foundation of China (Grant No. 11771046), 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] A.S. Householder. (1958). *Unitary Triangularization of a Nonsymmetric Matrix*. J. Assoc. Comput. Mach 5, 339?-342.
- [2] Cipra, B.A. *The Best of the 20th Century: Editors Name Top 10 Algorithms*. SIAM. News, 33,4.
- [3] Chen, M.F. (2005). *Eigenvalues, Inequalities, and Ergodic Theory*. Springer, London.
- [4] Chen, M.F. (2010). *Speed of stability for birth-death processes*. Front. Math. China 5(3), 379–515.
- [5] Chen, M.F. (2014). *Criteria for discrete spectrum of 1D operators*. Commu Math Stat 2, 279-309.
- [6] Chen, M.F. and Zhang, X. (2014). *Isospectral operators*. Commu Math Stat 2, 17-32.
- [7] Chen, M.F. (2016). *Efficient initials for computing the maximal eigenpair*. Front. Math. China 11(6): 1379–1418. See also volume 4 in the middle of the author’s homepage:

<http://math0.bnu.edu.cn/~chenmf>

A package based on the paper is available on CRAN now (by X.J. Mao). One may check it through the link:

<https://cran.r-project.org/web/packages/EfficientMaxEigenpair/index.html>

The authors’ papers cited in this article can be found from volumes 1-4 in the middle of the homepage above.

- [8] Chen, M.F. (2017a). *The charming leading eigenpair*. Adv. Math. (China) 46(4): 281–297.
- [9] Chen, M.F. (2017b). *Global algorithms for maximal eigenpair*. Front. Math. China 12(5): 1023–1043.
- [10] Chen, M.F. (2017c). *Trilogy on computing maximal eigenpair*. Springer Proceedings as one volume of Lecture Notes in Computer Science series.
- [11] Chen, M.F. (2018). *Hermitizable, Isospectral Complex Matrices or Differential Operators*. Front. Math. China.
- [12] Frolov, A.V. et al. *Thomas algorithm, pointwise version*. https://algorithms.wtf/Thomas_algorithm_pointwise_version
- [13] From Wikipedia. *Tridiagonal matrix algorithm*. https://en.wikipedia.org/wiki/Tri-diagonal_matrix_algorithm
- [14] Golub, G.H. and van der Vorst, H.A. (2000). *Eigenvalue computation in the 20th century*. J. Comput. Appl. Math. 123(1-2): 35-65.
- [15] G.W. Stewart. (2000). *The Decompositional Approach to Matrix Computation*. IEEE. Comput. Sci. Eng. 2(1): 50-59.
- [16] Moler, C. (1996). *Llewellyn Thomas*. https://en.wikipedia.org/wiki/Llewellyn_Thomas
- [17] Van der Vorst, H.A. and Golub, G.H. (1997). *150 Years old and still alive: eigenproblems*. The State of the Art in Numerical Analysis : 93 - 119.
- [18] Tang, T. and Yang, J. (2018). *Computing the maximal eigenpairs of large size tridiagonal matrices with $O(1)$ number of iterations*. Numer. Math. Theor. Meth. Appl. 11(4):877-894.