Commun. Math. Res.
doi: 10.4208/cmr.2020-0005

Vol. **36**, No. 1, pp. 93-112
March 2020

Finding the Maximal Eigenpair for a Large, Dense, Symmetric Matrix based on Mufa Chen's Algorithm

Tao Tang[1] and Jiang Yang[2,*]

[1] *Division of Science and Technology, BNU-HKBU United International College, Zhuhai, Guangdong, China, & SUSTech International Center for Mathematics, Southern University of Science and Technology, Shenzhen 518055, China.*

[2] *Department of Mathematics & SUSTech International Center for Mathematics, Southern University of Science and Technology, Shenzhen 518055, China.*

Received 15 February 2020; Accepted 5 March 2020

Abstract. A hybrid method is presented for determining maximal eigenvalue and its eigenvector (called eigenpair) of a large, dense, symmetric matrix. Many problems require finding only a small part of the eigenpairs, and some require only the maximal one. In a series of papers, efficient algorithms have been developed by Mufa Chen for computing the maximal eigenpairs of tridiagonal matrices with positive off-diagonal elements. The key idea is to explicitly construct effective initial guess of the maximal eigenpair and then to employ a self-closed iterative algorithm. In this paper, we will extend Mufa Chen's algorithm to find maximal eigenpair for a large scale, dense, symmetric matrix. Our strategy is to first convert the underlying matrix into the tridiagonal form by using similarity transformations. We then handle the cases that prevent us from applying Chen's algorithm directly, e.g., the cases with zero or negative super- or sub-diagonal elements. Serval numerical experiments are carried out to demonstrate the efficiency of the proposed hybrid method.

AMS subject classifications: 60J60, 34L15
Key words: Maximal eigenpair, symmetric matrix, dense matrix, tridiagonal matrix, Householder transformation, complexity, iteration.

*Corresponding author. *Email addresses:* tangt@sustech.edu.cn (T. Tang), yangj7@sustech.edu.cn (J. Yang)

1 Introduction

The best approach for computing all the eigenpairs (eigenvalues and eigenvectors) of a dense symmetric matrix involves three steps:

- *reduction*: reduce the given symmetric matrix A to tridiagonal form T (i.e. nonzero elements only occur on the diagonal and super-/sub-diagonals);
- *solution of tridiagonal eigen-problem*: compute all the eigenpairs of T;
- *back-transformation*: map the eigenvectors of T into those of A.

For an $N\times N$ matrix, the reduction and back-transformation steps require $\mathcal{O}(N^3)$ arithmetic operations each. Note that most algorithms for the tridiagonal eigen-problem also had cubic complexity in the worst case, including the QR algorithm and inverse iteration. As pointed out by [18] the tridiagonal problem can be the computational bottleneck for large problems taking nearly $70\sim80\%$ of the total time to solve the entire dense problem. As a result, numerous methods exist for the numerical computation of the eigenvalues of a real tridiagonal matrix to high accuracy, see, e.g., [2,10,11]. To find eigenvalues of a symmetric tridiagonal matrix typically requires $\mathcal{O}(N^2)$ operations [8], although fast algorithms exist which require $\mathcal{O}(N\ln N)$ [6].

In practice, many problems require finding only a small part of the eigenvalues and eigenvectors; for such problems, finding all the eigenpairs may be time consuming and wasteful. The problem of computing the maximal eigenpairs has been a classical subject, and the methods for this problem that are discussed most are the power method, the inverse method, the Rayleigh quotient method, and some hybrid method, see, e.g., [1,12,13]. Finding the largest eigenpairs has many applications in signal processing, control, and recent development of Google's PageRank algorithm. In a series of papers, Mufa Chen [3–5] developed some efficient algorithms for computing the maximal eigenpairs for tridiagonal matrices. The key idea is to explicitly construct effective initials for the maximal eigenpairs and also employ a self-closed iterative algorithm. The algorithm makes the total number of iterations independent of the size of the matrix.

In [14], the authors extended Chen's algorithm to deal with large scale tridiagonal matrices. By using appropriate scalings and by optimizing numerical complexity, we make the computational cost for each iteration to be $\mathcal{O}(N)$. As a result, it requires $\mathcal{O}(N)$ number of computational cost to obtain the maximal eigenpair.

The main purpose of this paper is to compute the maximal eigenpair for large, dense, symmetric matrices by further extending Chen's algorithm. Namely, for the three steps mentioned above, we first convert the given symmetric matrix to

a similar tridiagonal one by using the Householder algorithm, and then handle the tridiagonal eigenpair problem by extending Chen's algorithm. More precisely, for any full symmetric $N\times N$ matrix A, we first convert it into a symmetric tridiagonal matrix T by a similarity transformation Q:

$$A=Q^TTQ. \tag{1.1}$$

The similarity transformation is implemented by $(N-2)$ Householder transformations H of the form

$$H=I-2\mathbf{u}\mathbf{u}^T, \tag{1.2}$$

where $\mathbf{u}\in\mathbb{R}^N$ is a unit column vector. By making use of the special structure of the Householder transformation, the whole process of the tridiagonalization costs about $\frac{7}{3}N^3$ operations. After obtaining a tridiagonal matrix T, we may not be able to employ Chen's algorithm immediately as the positivity of all super- and sub-diagonal elements may not hold. Therefore, possible cases including zero and negative off-diagonal elements may occur. For the case with zero super- or sub-diagonal elements, we split T into some smaller tridiagonal matrices and then apply Chen's algorithm to each of them. For the negative off-diagonal case, we will introduce a diagonal matrix P depending on the signs of the off-diagonal elements of T. By using the similarity transformation

$$W=P^TTP, \tag{1.3}$$

a matrix suitable for Chen's Algorithm can be obtained. Hence, to obtain the maximal eigenpair for dense symmetric matrices the total number of operations is

$$\frac{7}{3}N^3+p_{\text{ch}}N, \tag{1.4}$$

where p_{ch} is the iteration constant for Chen's algorithm. The biggest advantage of Chen's algorithm is that the constant p_{ch} is quite small (about 10) even for large size matrices. In contrast, the power method, which is the classical method for computing maximal eigenpair, costs $p_{\text{pw}}N^2$ operations at this step. The iteration constant for the power method, p_{pw}, becomes very large for large size matrices, in particular for cases with clustered eigenvalues. The reason is that the convergence rate of the power method depends on the ratio of the second largest eigenvalue over the largest one. In Section 4, our numerical example taken from [15] shows that p_{pw} can be as big as $\mathcal{O}(N)$. Thus, for the cases with clustered eigenvalues, our proposed method has some obvious advantage.

The rest of this paper is organized as follows. In Section 2, we will briefly review Mufa Chen's original algorithm designed for tridiagonal matrices. The

main part Section 3 provides a hybrid algorithm for large dense symmetric matrices. The key element is to extend Chen's algorithm to handle negative or zero off-diagonal elements. In Section 4, numerical experiments are carried out to demonstrate the effectiveness of the proposed hybrid method. Some concluding remarks will be provided in the final section.

2 Mufa Chen's algorithm

To better illustrate the proposed hybrid method, we first review Chen's algorithm presented in [3, 4] which seeks maximal eigenpair for tridiagonal matrices with positive off-diagonal elements. Starting from an $(N+1)\times(N+1)$ tridiagonal matrix A and by using a shift $Q:=A-mI$, where I is the identity matrix and

$$m=\max_{0\le i\le N}\sum_{j=1}^{N}a_{i,j}, \tag{2.1}$$

we may assume that

$$Q=\begin{pmatrix} -(b_0+c_0) & b_0 & 0 & 0 & \cdots \\ a_1 & -(a_1+b_1+c_1) & b_1 & 0 & \cdots \\ 0 & a_2 & -(a_2+b_2+c_2) & b_2 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & a_N & -(a_N+c_N) \end{pmatrix}, \tag{2.2}$$

where $a_i,b_i>0$, $c_i\ge 0$ but $c_j\not\equiv 0$. As in [3], define, for $1\le n\le N$,

$$\mu_0=1,\quad \mu_n=\mu_{N-1}\frac{b_{N-1}}{a_n}, \tag{2.3}$$

$$r_0=1+\frac{c_0}{b_0},\quad r_n=1+\frac{a_n+c_n}{b_n}-\frac{a_n}{b_n r_{N-1}}, \tag{2.4}$$

$$h_0=1,\quad h_n=h_{N-1}r_{N-1}. \tag{2.5}$$

Furthermore, define

$$\phi_n=\sum_{k=n}^{N}\frac{1}{h_k h_{k+1}\mu_k b_k},\quad 0\le n\le N \tag{2.6}$$

with $h_{N+1}=c_N h_N+a_N(h_N-h_{N-1})$ and $b_N=1$.

Theorem 2.1. ([3]) *For a given tridiagonal matrix A, define $m,(a_i,b_i,c_i)$ as in* (2.1)-(2.2). *Set*

$$(\tilde{v_0})_i = h_i\sqrt{\phi_i},\quad 0\le i\le N;\quad v_0=\frac{\tilde{v}_0}{\|\tilde{\mathbf{v}}_0\|_2}, \tag{2.7}$$

where h_i,ϕ_i are defined by (2.3)-(2.6). *Furthermore, let*

$$z_0=\frac{1}{\delta_0};\quad \delta_0=\max_{0\le n\le N}\left[\sqrt{\phi_n}\sum_{k=0}^{n}\mu_k h_k^2\sqrt{\phi_k}+\frac{1}{\sqrt{\phi_n}}\sum_{j=n+1}^{N}\mu_j h_j^2\phi_j^{3/2}\right]. \tag{2.8}$$

With the initial values v_0 and shift δ_0, we perform the shifted inverse power method on matrix Q given in (2.2), *which produces a vector sequence of $\{v_k\}$ and associated Rayleigh quotient $\{z_k\}$. Then $m-z_k$ converges to the largest eigenvalue of A and v_k converges to the corresponding eigenvector.*

Although the above theorem gives a useful initial eigenpair approximation, using the inverse iteration in general requires many iterations. This in turn slows down the convergence of the computation. To improve this, a more robust and accurate method was introduced in [4]. First, we take the similarity transformation on Q using the diagonal matrix $\mathrm{Diag}(h_i)$, whose main diagonal is by the vector h, i.e.,

$$\tilde{Q}=\mathrm{Diag}(h_i)^{-1}Q\mathrm{Diag}(h_i). \tag{2.9}$$

It is easy to check that $\tilde{Q}$ is of the following form

$$\tilde{Q}=\begin{pmatrix} -\tilde{b}_0 & \tilde{b}_0 & 0 & 0 & \cdots \\ \tilde{a}_1 & -(\tilde{a}_1+\tilde{b}_1) & \tilde{b}_1 & 0 & \cdots \\ 0 & \tilde{a}_2 & -(\tilde{a}_2+\tilde{b}_2) & \tilde{b}_2 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \tilde{a}_N & -(\tilde{a}_N+\tilde{b}_N) \end{pmatrix}, \tag{2.10}$$

where $\tilde{a}_i,\tilde{b}_i>0$. We then define

$$\nu_0=1,\quad \nu_n=\nu_{N-1}\frac{\tilde{b}_{N-1}}{\tilde{a}_n},\quad 1\le n\le N, \tag{2.11}$$

$$\varphi_n=\sum_{k=n}^{N}\frac{1}{\nu_k\tilde{b}_k},\quad 0\le n\le N. \tag{2.12}$$

Similar as before, we define

$$\tilde{\delta}_0=\max_{0\le k\le N}\left[\sqrt{\varphi_k}\sum_{i=0}^{k}\nu_i\sqrt{\varphi_i}+\frac{1}{\sqrt{\varphi_k}}\sum_{j=k+1}^{N}\nu_j\varphi_j^{3/2}\right]. \tag{2.13}$$

With the above preparations, we are now ready to state Chen's algorithm:

- Choose the initial eigenpair using (2.12) and (2.13): $\boldsymbol{\omega}^{(0)}=\sqrt{\boldsymbol{\varphi}}$,

$$\mathbf{v}^{(0)}=\frac{\boldsymbol{\omega}^{(0)}}{\|\boldsymbol{\omega}^{(0)}\|_2},\quad z^{(0)}=\frac{1}{\tilde{\delta}_0}.$$

- For $n=1,2,\cdots$, solve the linear equation

$$\left(-\tilde{Q}-z^{(n-1)}I\right)\boldsymbol{\omega}=\mathbf{v}^{(n-1)} \tag{2.14}$$

  and then define $\mathbf{v}^{(n)}=\boldsymbol{\omega}/\|\boldsymbol{\omega}\|_2$.

- Update $z^{(n)}=1/\tilde{\delta}_n$ with

$$\tilde{\delta}_n=\max_{0\le k\le N}\frac{1}{v_k^{(n)}}\left[\varphi_k\sum_{i=0}^{k}\nu_i v_i^{(n)}+\sum_{j=k+1}^{N}\nu_j\varphi_j v_j^{(n)}\right] \tag{2.15}$$

  until $\left|z^{(n)}-z^{(n-1)}\right|$ is smaller than some given tolerance.

- Output the largest eigenpair as $\lambda_{\max}(A)=m-z^{(n)}$, $\mathbf{g}=\mathrm{Diag}(h_i)\mathbf{v}^{(n)}$.

Note that the key contribution of the above algorithm is the use of the iteration (2.15), which was first introduced in [4].

3 An extension to dense symmetric matrices

In this section, we will extend Chen's algorithm to the general symmetric matrices. The first step is to convert the symmetric matrices into tridiagonal ones by similarity transformation, which is implemented by the Householder transformation (HT).

3.1 Tridiagonalization

We first introduce

$$H_{\mathbf{w}}=I-2\mathbf{w}\mathbf{w}^T,\quad \text{with}\quad \mathbf{w}^T\mathbf{w}=1,\quad \mathbf{w}\in\mathbb{R}^N. \tag{3.1}$$

Lemma 3.1. *For any $\mathbf{u}\in\mathbb{R}^N$ and $\mathbf{v}\in\mathbb{R}^N$ such that $\|\mathbf{u}\|_2=\|\mathbf{v}\|_2$, there exists a Householder transformation H such that*

$$\mathbf{v}=H\mathbf{u},$$

where

$$\mathbf{w}=\frac{\mathbf{u}-\mathbf{v}}{\|\mathbf{u}-\mathbf{v}\|_2},\quad \textit{and}\quad H=I-2\mathbf{w}\mathbf{w}^T. \tag{3.2}$$

It is easy to check that an HT defined by (3.2) is symmetric and orthogonal, i.e.,

$$H=H^T,\quad H^TH=I. \tag{3.3}$$

Now we use HTs to convert any symmetric matrix A into tridiagonal form. First, we split A as

$$A=\begin{pmatrix} a_{11} & \mathbf{r}_1^T \\ \mathbf{r}_1 & A_{22}\end{pmatrix}, \tag{3.4}$$

with

$$\mathbf{r}_1=(a_{21},a_{31},\cdots,a_{N1})^T,\quad A_{22}=\left(a_{ij}\right)_{i,j=2}^N. \tag{3.5}$$

We define

$$\mathbf{v}_1=(\sigma_1,0,\cdots,0)^T,\quad \text{where}\quad \sigma_1=-\text{sign}(a_{21})\|\mathbf{r}_1\|_2. \tag{3.6}$$

Note that if $\text{sign}(a_{21})=0$, we can just take $\sigma_1=\|\mathbf{r}_1\|_2$. It can be verified that

$$\|\mathbf{v}_1\|_2=\|\mathbf{r}_1\|_2. \tag{3.7}$$

From Lemma 3.1 we can find an Householder transformation H_1 such that

$$H_1\mathbf{r}_1=\mathbf{v}_1,$$

where

$$\mathbf{u}_1=\mathbf{r}_1-\mathbf{v}_1,\quad \text{and}\quad H_1=I-2\frac{\mathbf{u}_1\mathbf{u}_1^T}{\mathbf{u}_1^T\mathbf{u}_1}. \tag{3.8}$$

We let

$$U_1=\begin{pmatrix} 1 & \\ & H_1\end{pmatrix}, \tag{3.9}$$

which gives that

$$A^{(2)}=U_1AU_1=\begin{pmatrix} a_{11} & \mathbf{v}_1^T \\ \mathbf{v}_1 & H_1A_{22}^{(1)}H_1\end{pmatrix}:=\begin{pmatrix} \eta_1 & \mathbf{v}_1^T \\ \mathbf{v}_1 & A_2\end{pmatrix}. \tag{3.10}$$

It is easy to check that U_1 is a HT and the associating reflection vector is defined as

$$\mathbf{w_1}=\begin{pmatrix} 0 \\ \mathbf{u}_1\end{pmatrix},\quad \text{and}\quad U_1=I-2\mathbf{w_1}\mathbf{w_1}^T.$$

Using the same procedure above on A_2 and following similar processes yield U_2, U_3, $\cdots$, U_{N-2} such that

$$A^{(N-2)} = \left(\prod_{j=1}^{N-2} U_{N-1-j}\right) A \prod_{j=1}^{N-2} U_j = \begin{pmatrix} \eta_1 & \sigma_1 & & & \\ \sigma_1 & \eta_2 & \sigma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \sigma_{N-2} & \eta_{N-1} & \sigma_{N-1} \\ & & & \sigma_{N-1} & \eta_N \end{pmatrix} := B. \tag{3.11}$$

Let $U = U_1 U_2 \cdots U_{N-2}$. It is cleat that U is an orthogonal matrix, and the following result holds

$$B = U^T A U, \tag{3.12}$$

where we have used the fact that each U_j is symmetric and orthogonal.

Remark 3.1. We point out that $\sigma_1 = -\text{sign}(a_{21})\|\mathbf{r}_1\|_2$ is chosen to reduce possible numerical overflow due to (3.8). More precisely, note that

$$\mathbf{u}_1 = (a_{21} + \text{sign}(a_{21})\|\mathbf{r}_1\|_2, a_{31}, \cdots, a_{N1})^T, \tag{3.13}$$

which gives $\|\mathbf{u}_1\|_2 \geq |a_{21}| + \|\mathbf{r}_1\|_2$. However, if we take the opposite sign, i.e., $\sigma_1 = \text{sign}(a_{21})\|\mathbf{r}_1\|_2$, then the estimate will change to $\|\mathbf{u}_1\|_2 \geq ||a_{21}| - \|\mathbf{r}_1\|_2|$, which indicates that in some possible cases instability may occur in (3.8) as $\|\mathbf{u}_1\|_2$ may become very small.

Remark 3.2. To save the storages of U_j, we only need to keep the refection vectors $\mathbf{w}_j$ associating with U_j, which yields $\mathcal{O}(N^2)$ storage. Meanwhile, it only takes $\mathcal{O}(N)$ instead of $\mathcal{O}(N^2)$ operations when performing the product of an Householder transformation with a vector:

$$H_{\mathbf{w}}\mathbf{v} = \left(I - 2\mathbf{w}\mathbf{w}^T\right)\mathbf{v} = \mathbf{v} - 2\left(\mathbf{w}^T\mathbf{v}\right)\mathbf{w}. \tag{3.14}$$

3.2 Zero off-diagonal elements

After tridiagonalization, the super-/sub-diagonal elements may not be all positive which prevent us from using Chen's algorithm directly. We first fix the case

that the off-diagonals have zero elements. Without loss of generality, we assume that there is only one zero element: $\sigma_k = 0$. In this case, let us split B as

$$B = \begin{pmatrix} B_1 & \\ & B_2 \end{pmatrix}, \tag{3.15}$$

where

$$B_1 = \begin{pmatrix} \eta_1 & \sigma_1 & & & \\ \sigma_1 & \eta_2 & \sigma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \sigma_{k-2} & \eta_{k-1} & \sigma_{k-1} \\ & & & \sigma_{k-1} & \eta_k \end{pmatrix}, \quad B_2 = \begin{pmatrix} \eta_{k+1} & \sigma_{k+1} & & & \\ \sigma_{k+1} & \eta_{k+2} & \sigma_{k+2} & & \\ & \ddots & \ddots & \ddots & \\ & & \sigma_{N-2} & \eta_{N-1} & \sigma_{N-1} \\ & & & \sigma_{N-1} & \eta_N \end{pmatrix}.$$

Lemma 3.2. *For B given by* (3.15)*, assume that $(\lambda, \mathbf{p})$ and $(\mu, \mathbf{q})$ are eigenpairs of B_1 and B_2 respectively. Then $(\lambda, \tilde{\mathbf{p}})$ and $(\mu, \tilde{\mathbf{q}})$ are eigenpairs of B, where*

$$\tilde{\mathbf{p}} = \begin{pmatrix} \mathbf{p} \\ \mathbf{0}_{(N-k)\times 1} \end{pmatrix}, \quad \tilde{\mathbf{q}} = \begin{pmatrix} \mathbf{0}_{k\times 1} \\ \mathbf{q} \end{pmatrix}. \tag{3.16}$$

Based on the above lemma, we can derive following proposition easily.

Proposition 3.1. *Given an block diagonal matrix B*

$$B = \begin{pmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_k \end{pmatrix}. \tag{3.17}$$

If $(\lambda, \mathbf{p})$ is an eigenpair of B_j, then $(\lambda, \tilde{\mathbf{p}})$ is an eigenpair of B with

$$\tilde{\mathbf{p}} = \left(\mathbf{0}_1, \cdots, \mathbf{0}_{j-1}, \mathbf{p}, \mathbf{0}_{j+1}, \cdots, \mathbf{0}_k\right)^T, \tag{3.18}$$

where $\mathbf{0}_i$ has the same size as row size of B_i for each i.

Hence, when multi zero off-diagonal elements occur, we can apply the algorithm to B_j respectively.

3.3 Negative off-diagonal elements

If the off-diagonals have negative elements (but all no zero), we define

$$P=\text{diag}(p_1,p_2,\cdots,p_N), \tag{3.19}$$

where

$$p_1=1, \quad \text{and} \quad p_k=\frac{\text{sign}(\sigma_{k-1})}{p_{k-1}} \quad \text{for} \quad 2\leq k\leq N. \tag{3.20}$$

It is seen that p_k is either 1 or -1. Hence, it is easy to verify that P is symmetric and orthogonal, i.e., $P^T=P$ and $P^TP=I$, and satisfies

$$P^TBP=\begin{pmatrix} \eta_1 & |\sigma_1| & & & & \\ |\sigma_1| & \eta_2 & |\sigma_2| & & & \\ & \ddots & \ddots & \ddots & & \\ & & |\sigma_{N-2}| & \eta_{N-1} & |\sigma_{N-1}| & \\ & & & |\sigma_{N-1}| & \eta_N & \end{pmatrix}.$$

Clearly, with this similarity transformation, we can guarantee that all off-diagonal elements are positive.

3.4 Complexity of storage and operations

We first present our hybrid scheme, Algorithm 1, which can be used to compute the maximal eigenpair.

We now discuss the complexity of the algorithm:

- In Step **1**, the whole process of tridiagonalization takes $\mathcal{O}(N^3)$ operations and $\mathcal{O}(N^2)$ storage for storing $\mathbf{w}_1,\cdots,\mathbf{w}_{N-2}$ associated with the Householder transformations $U_1,\cdots,U_{N-2}$. Even though the cost of operations looks very expensive, the tridiagonalization is necessary for getting all eigenpairs. Note in Matlab this is done by using the function `eig(·)`. Moreover, the resulting tridiagonal matrix will be useful to compute all the eigenpairs if we are not just interested in the maximal eigenpair.
- In Step **2**, it does not increase the operation cost.
- In Step **3**, the simple similarity transformation only takes N operations.
- In Step **4**, the optimal operation cost $\mathcal{O}(pN)$ is achieved in [14], where p is the number of the iterations in Chen's algorithm. It is shown by numerical examples that this iteration number is small even for very large matrices.

Algorithm 1 A hybrid method for computing the maximal eigenpair

Given: a symmetric matrix $A\in\mathbb{R}^{N\times N}$.

Step 1. Following Section 3.1 to find HTs of $U_1,\cdots,U_{N-2}$ and using these HTs to convert A into a tridiagonal matrix B.
Step 2. **If** some off-diagonal elements in B are zero, then the splitting techniques in Section 3.2 is used.
Step 3. **If** some off-diagonal elements in B are negative, then the similarity transformation defined in Section 3.3 is used.
Step 4. Apply Chen's algorithm in Section 2 to get the maximal eigenpair for the tridiagonal matrices in **Steps 1-3**.
Step 5. Take the back-transformations of ones in **Step 1** and **Step 3** to recover the maximal eigenpair of A.

- In Step **5**, the inverse similarity transformation in Step **3** only takes N operations and it takes $N(N-2)$ operations for the series inverse similarity transformation in Step **1**, see Remark 3.2.

4 Numerical experiments

In this section, we will take several numerical examples to demonstrate the performance of the extension of the proposed hybrid algorithm.

Example 4.1. Consider the signed Gauss-Laguerre quadrature matrix with a sign factor γ_j in front of β_j given below

$$G_{N+1}^s=\begin{pmatrix}\alpha_0 & \gamma_0\beta_0 & 0 & 0 & \cdots\\ \gamma_0\beta_0 & \alpha_1 & \gamma_1\beta_1 & 0 & \cdots\\ 0 & \gamma_1\beta_1 & \alpha_2 & \gamma_2\beta_2 & \cdots\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \gamma_{N-1}\beta_{N-1} & \alpha_N\end{pmatrix}, \tag{4.1}$$

where

$$\alpha_i=2i+1+\alpha,\quad \beta_i=\sqrt{(i+1)(i+1+\alpha)},\quad i\geq 0,$$

and α is chosen as $\alpha=-0.25$.

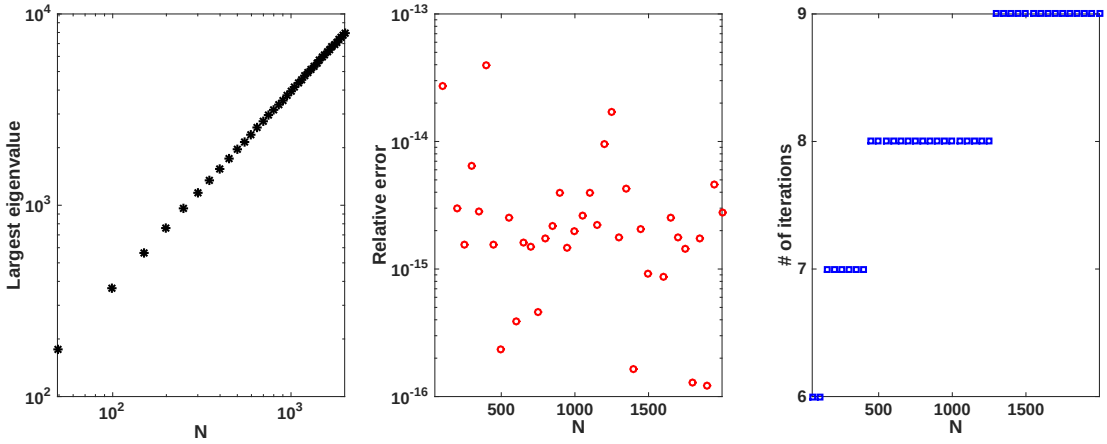

Figure 1: (Example 4.1) the largest eigenvalue in the signed Gauss-Laguerre quadrature versus the matrix size N (left), the corresponding relative errors (middle), and the total number of iterations (right).

If we choose the sign factor $\gamma_j \equiv 1$, the matrix becomes the original Gauss-Laguerre quadrature matrix given in [9], which is tested in our precious work [14]. In this example, we set $\gamma_j = (-1)^j$. Hence, the off-diagonal elements can be negative. The original Chen's algorithm cannot be applied directly as it requires the positivity of super-/sub-diagonal elements. Using the technique in (3.19)-(3.20), we define

$$P = \mathrm{diag}(1,1,-1,-1,1,1,-1,-1,\cdots)^T,$$

which ends up with

$$G_{N+1} := P^T G^s_{N+1} P = \begin{pmatrix} \alpha_0 & \beta_0 & 0 & 0 & \cdots \\ \beta_0 & \alpha_1 & \beta_1 & 0 & \cdots \\ 0 & \beta_1 & \alpha_2 & \beta_2 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \beta_{N-1} & \alpha_N \end{pmatrix}. \tag{4.2}$$

In Fig. 1, we display the growth of the largest eigenvalue with respect to the matrix size N. It is noticed that the eigenvalue grows almost quadratically with respect to N. By use of the method given by Chen [4] together with scaling and recurrence techniques proposed in [14], the corresponding approximation errors can reach the machine accuracy after 9 iterations for $N = 1500$.

We further increase the value of N to 10000, and the corresponding numerical results are presented in Table 1. It is observed that very accurate approximations

Table 1: (Example 4.1) Numerical approximation for the largest eigenvalue of the signed Gauss-Laguerre quadrature problem with $N=10000$. The exact value of the largest eigenvalue is $\lambda_{\max}=3.987464700035208e+04$.

# iteration	Numerical	Absolute Error	Relative Error
0	3.989034126621150e+04	15.69	3.94e-04
3	3.988599366128229e+04	11.35	2.85e-04
6	3.987752624280322e+04	2.88	7.22e-05
9	3.987464796093082e+04	9.61e-04	2.41e-08
11	3.987464700035210e+04	5.82e-11	1.46e-15

are obtained after 10 iterations even for such a large size. In fact, as the largest eigenvalue is of order 10^4 it takes a couple of more iterations for this case.

Example 4.2. Consider a symmetric random matrix taken as

$$H=\frac{1}{2\sqrt{n}}\begin{bmatrix} \chi_{N-1} & N(0,2) & & & & & \\ N(0,2) & \chi_{N-2} & N(0,2) & & & & \\ & N(0,2) & \chi_{n-3} & N(0,2) & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & N(0,2) & \chi_{n-k} & N(0,2) & \\ & & & & N(0,2) & \chi_{n-k-1} & \end{bmatrix}, \tag{4.3}$$

where $N(0,2)$ is a zeros-mean Gaussian with variance 2, and χ_r is the square-root of a χ^2-distributed number with r degrees of freedom. We take $n=10^6$ and $k=1000$.

Note that $N(0,2)$ may contain nonpositive numbers, so the techniques proposed in *Steps 2* and *3* in Algorithm 1 must be used. In Fig. 2, we first show the elements and the distribution of eigenvalues. It is observed that indeed some of the off-diagonal elements are nonpositive. It is further seen that the main diagonal are close to 1, and the eigenvalues are densely clustered in $(0.997,1.002)$. It would take thousands of iterations if the power method is used since the ratio of the second largest eigenvalue over the largest eigenvalue is very close to 1. We have tested these random matrices tens of times. Each test takes about $15\sim20$ iterations to achieve $10\sim15$ significant digits. Table 2 presents one typical test for the largest eigenvalue approximation.

In the following two examples, two kinds of dense symmetric matrices will be considered.

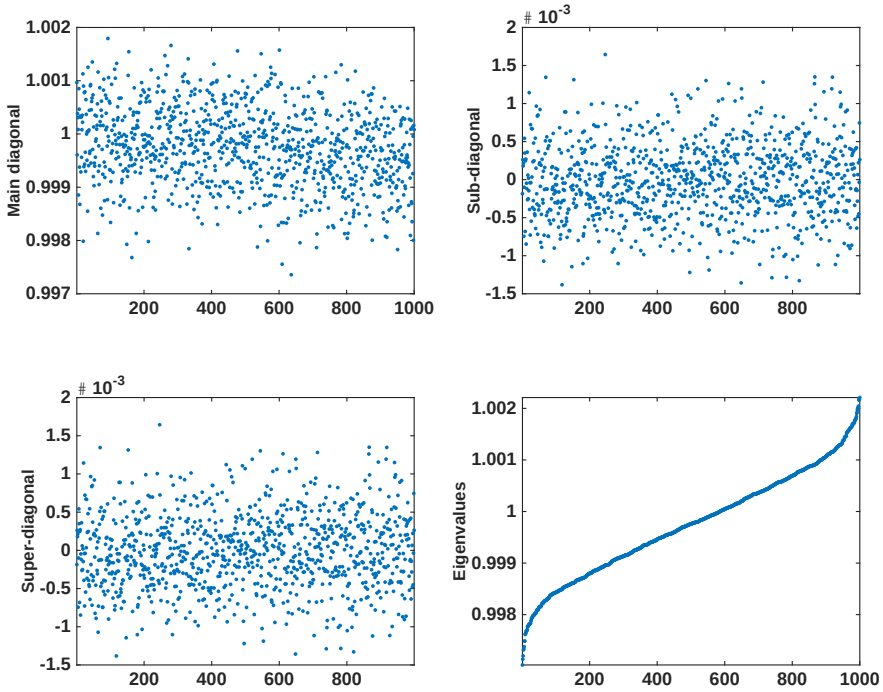

Figure 2: (Example 4.2) the distribution of elements and eigenvalues.

Table 2: (Example 4.2) The approximations of the largest eigenvalue in each iteration.

# iteration	Numerical	Error
1	1.002470857648490	6.878357132089619e-05
6	1.002454385910175	5.231183300624664e-05
11	1.002430848213373	2.877413620439029e-05
16	1.002408106105916	6.032028747160467e-06
21	1.002402074077465	2.955413691552167e-13

Example 4.3. Consider the classical Hilbert matrix $H=(H_{ij})\in\mathbb{R}^{N\times N}$ given as follows.

$$H_{ij}=(i+j-1)^{-1},\quad 1\leq i,j\leq N. \tag{4.4}$$

The Hilbert matrix comes from the least square approximation of functions on $(0,1)$ by polynomials with the natural basis $\{1,x,\cdots,x^{N-1}\}$, i.e.,

$$\min_{a_0,a_1,\cdots,a_{N-1}} E(a_0,a_1,\cdots,a_{N-1})=\min_{a_0,a_1,\cdots,a_{N-1}}\frac{1}{2}\int_0^1(f(x)-P_N(x))^2dx, \tag{4.5}$$

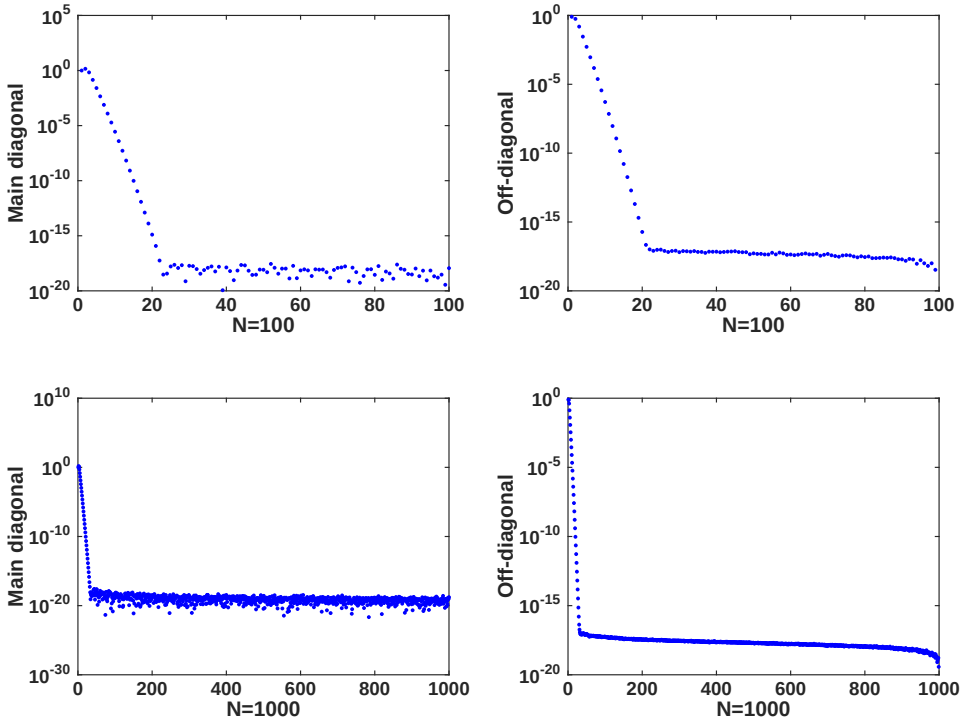

Figure 3: (Example 4.3) elements after tridiagonalization: upper row for $N=100$, bottom row for $N=1000$.

where $P_N(x)=a_0+a_1x+\cdots+a_{N-1}x^{N-1}$. The ith normal equation $\partial E/\partial a_{i-1}=0$ leads to

$$\sum_{j=1}^{N}\int_0^1 x^{i+j-2}dx a_{j-1}=\int_0^1 x^{i-1}f(x)dx, \tag{4.6}$$

further amounting to

$$\sum_{j=1}^{N}(i+j-1)^{-1}a_j=\int_0^1 x^{i-1}f(x)dx=:f_i. \tag{4.7}$$

This yields a linear system

$$H\mathbf{a}=\mathbf{f}, \tag{4.8}$$

where H is the Hilbert matrix. It is well known that the Hilbert matrix is highly ill-conditioned.

We first perform the tridiagonalization process to the Hilbert matrix, and the elements of the main diagonal and the super-/sub-diagonal are plotted in Fig. 3.

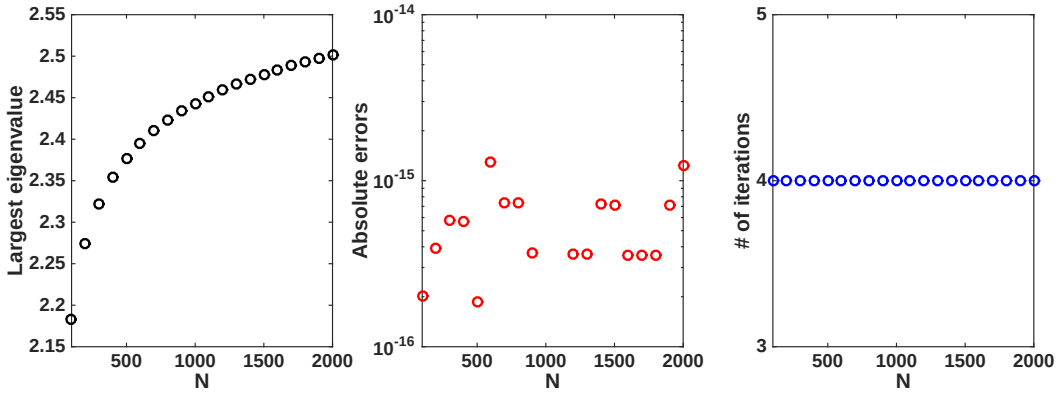

Figure 4: (Example 4.3) numerical performances of Algorithm 1 for Hilbert matrices.

It is seen that all elements decay exponentially, reaching the machine limit after $N=20$. We then propose to chop super-/sub-diagonal elements to zero when they are smaller than 10^{-15}. In this setting, the original Chen's algorithm cannot be used directly due to zero super-/sub-diagonal elements. To fix this, we apply the splitting technique presented in Section 3.2.

In Fig. 4, we present the numerical results, which contains the numerical errors and number of iterations needed for each N. It is observed that the iteration number stays at the constant number 4, while the absolute errors almost reach the machine accuracy. We emphasize that only 4 iterations are required even the size of the Hilbert matrix is as large as 2000×2000.

Example 4.4. This example comes from [15]. We generate real symmetric and positive definite matrices using the MATLAB function `randsvd`. More precisely, we use Higham's test matrices [16] by the following MATLAB command

$$>>\texttt{A}=\texttt{gallery}(\texttt{randsvd},\texttt{N},-\texttt{cnd},\texttt{mode})$$

The eigenvalue distribution and condition number of A can be controlled by the input arguments $mode\in\{1,2,3,4,5\}$ and $cnd:=\alpha\geq 1$, as described below:

- $mode=1$, one large: $\lambda_1\approx 1$, $\lambda_i\approx\alpha^{-1}$, $i=2,\cdots,N$;
- $mode=2$, one small: $\lambda_N\approx\alpha^{-1}$, $\lambda_i\approx 1$, $i=1,\cdots,N-1$;
- $mode=3$, geometrically distributed: $\lambda_i\approx\alpha^{-(i-1)/(N-1)}$, $i=1,\cdots,N$;
- $mode=4$, arithmetically distributed: $\lambda_i\approx 1-(1-\alpha^{-1})(i-1)/(N-1)$, $i=1,\cdots,N$;

- *mode* = 5, random with uniformly distributed logarithm: $\lambda_i \approx \alpha^{-r(i)}$, $i = 2,\cdots,N$, where $r(i)$ are pseudo-random values drawn from the standard uniform distribution on $(0,1)$.

In this example, we fix *mode* = 3 and *cnd* = 1.5, and will compare our hybrid method Algorithm 1 with the power method. The power method is the classical algorithm for computing the maximal eigenvector:

$$\mathbf{v}^{(k+1)} = \frac{A\mathbf{v}^{(k)}}{\|A\mathbf{v}^{(k)}\|}, \quad k = 1,2,\cdots, \tag{4.9}$$

and similarly for the eigenvalue.

Before presenting the numerical results, let us look at the parameter effects. Since *mode* = 3 and $\alpha = 1.5$, the convergence speed of the power method with p iterations is given by

$$\mathcal{O}\left(\left(\frac{\lambda_2}{\lambda_1}\right)^p\right) \approx 1.5^{-\frac{p}{N-1}}. \tag{4.10}$$

In order to get accurate approximations, we need to take $p > N$. Thus, the total number of computations of the power methods is dominated by $\mathcal{O}(N^2)$ for a full matrix without particular structures.

Fig. 5 compares the numerical performances of Algorithm 1 and the power method. As expected, the power methods take very large number of iterations to achieve satisfactory results. On the other hand, our hybrid method only takes very small number of iterations. Thanks to the code for tridiagonalization given in [17], the CPU time of our methods takes only 3% of the one used by the power method.

5 Concluding remarks

The problem considered in this work is the efficient computation of the maximum eigenvalue and its corresponding eigenvector of a large, dense, symmetric matrix to a prescribed tolerance. Finding the maximal eigenpair can be a tough job. The whole algorithm can be $\mathcal{O}(N^3)$ when N is sufficiently large. As most efficient algorithms to find eigenvalues and eigenvectors use the Householder algorithm, even the most powerful algorithm requires $\mathcal{O}(N^3)$ operation. Knowing that we cannot improve the Householder algorithm, we look for ways to increase the speed for finding the maximal eigenpairs of the corresponding tridiagonal matrices. The complexity of most existing algorithms for this part is proportional to the matrix size N, together with an iteration constant. A significant breakthrough

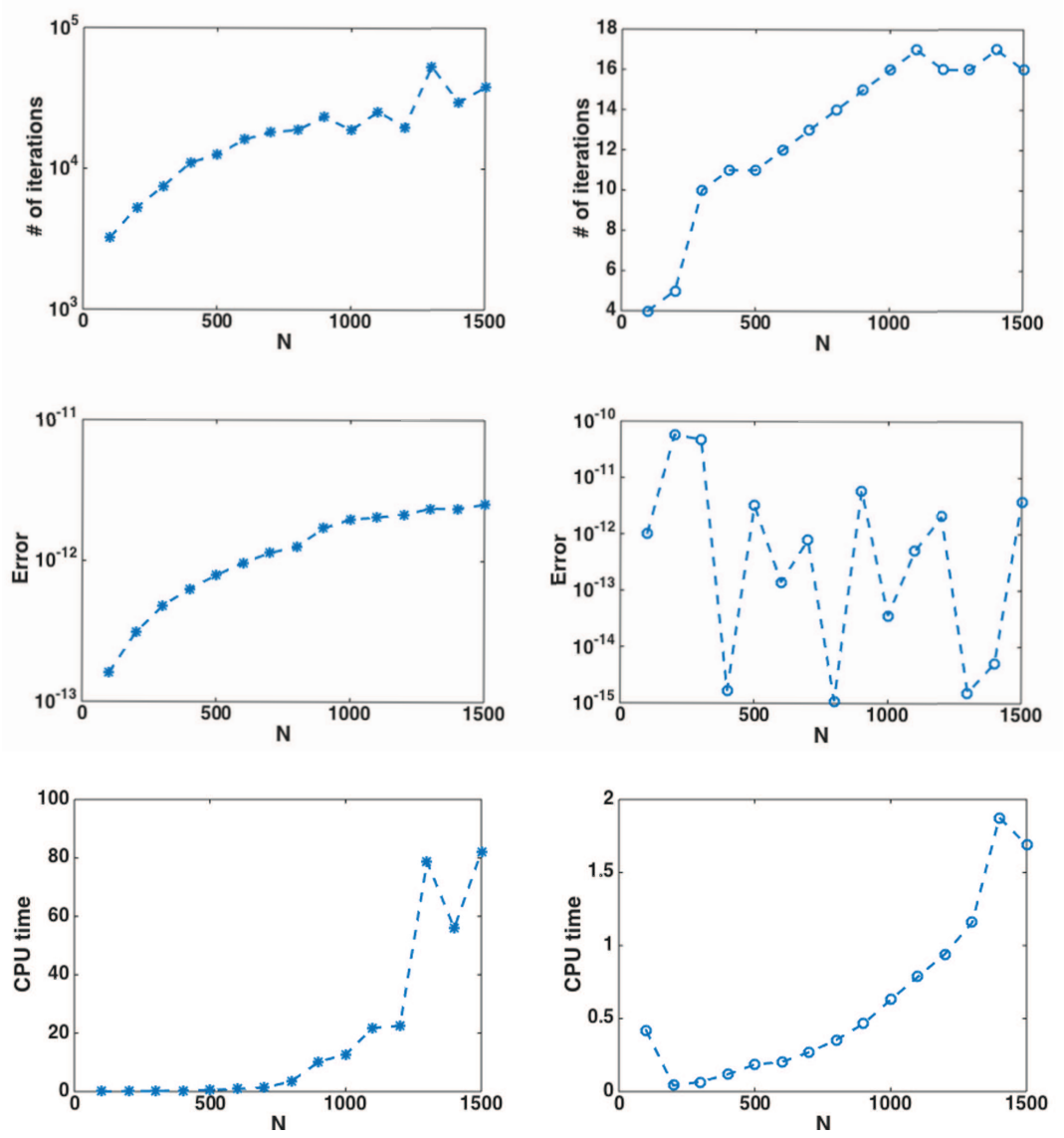

Figure 5: (Example 4.4) Comparisons of numerical performances of the power method (left column) with Algorithm 1 (right column).

on computing maximal eigenpair for tridiagonal matrices has been proposed by Chen recently, which minimizes the cost of the iteration.

Our work in this paper extends Chen's idea to deal with the maximal eigenpair problem for large and dense matrices. Numerical tests and comparisons are made to study the performance of the proposed algorithm, which suggest that our hybrid method is fast and reliable for computing the maximal eigenpairs.

We close this work with two general remarks on the application of the algorithm presented in this paper. The first is that the algorithm can be extended to non-symmetrical cases, as long as the underlying matrix can be tridiagonalized. As demonstrated in [14], Chen's method can deal with non-symmetrical tridiagonal matrices. The second remark is that the present method may be further extended to obtain partial or all eigenpairs by combining some shift inverse techniques. This will be a meaningful future work which can provide an effective numerical algorithm to obtain all eigenpairs with $\frac{7}{3}N^3 + p_{\text{ch}}N^2$ operations, where the constant p_{ch} is small.

Acknowledgments

This work is partially supported by the Special Project on High-Performance Computing of the National Key R&D Program under No. 2016YFB0200604, the National Natural Science Foundation of China (NSFC) Grant No. 11731006, and the NSFC/Hong Kong RRC Joint Research Scheme (NFSC/RGC 11961160718). The work of J. Yang is supported by NSFC-11871264 and Natural Science Foundation of Guangdong Province (2018A0303130123).

References

[1] P. Arbenz, Lecture notes on solving large scale eigenvalue problems, Computer Science Department, ETH, 2016.
`https://people.inf.ethz.ch/arbenz/ewp/Lnotes/lsevp.pdf`

[2] W. Barth, R. S. Martin and J. H. Wilkinson, Calculation of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection, Numer. Math., 9 (1967), pp. 386-393.

[3] M. F. Chen, Efficient initials for computing maximal eigenpair, Front. Math. China, 11 (2016), pp. 1397-1418.

[4] M. F. Chen, Global algorithms for maximal eigenpairs, Front. Math. China, 12(5) (2017), pp. 1023-1043.

[5] M. F. Chen and Y. S. Li, Improved global algorithms for maximal eigenpair, Front. Math. China, 14 (2019), pp. 1077-1116.

[6] E. S Coakley and V. Rokhlin, A fast divide-and-conquer algorithm for computing the spectra of real symmetric tridiagonal matrices, Appl. Comput. Harmon. Anal., 34 (2012), pp. 379-414.

[7] A. Edelman and N. R. Rao, Random matrix theory, Acta Numerica, (2005), pp. 1-65.

[8] D. Gill and E. Tadmor, An $\mathcal{O}(N^2)$ method for computing the eigensystem of $N\times N$ symmetric tridiagonal matrices by the divide and conquer approach, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 161-173.

[9] G. H. Golub and J. H. Welsch, Calculation of Gauss quadrature rules, Math. Comp., 23 (1969), pp. 221-239.

[10] M. Gu and S. C. Eisenstat, A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 172-191.

[11] I. C. F. Ipsen and E. R. Jessup, Solving the symmetric tridiagonal eigenvalue problem on the hypercube, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 203-229.

[12] Y. Saad, Numerical Methods for Large Eigenvalue Problems, SIAM, Philadelphia, 2011.

[13] D. B. Szyld, Criteria for combining inverse and Rayleigh quotient iteration, SIAM J. Numer. Anal., 25 (1988), pp. 1369-1375.

[14] T. Tang and J. Yang, Computing the maximal eigenpairs of large size tridiagonal matrices with $\mathcal{O}(1)$ number of iterations, Numer. Math. Theor. Meth. Appl., 11 (2018), pp. 877-894.

[15] T. Ogita and K. Aishima, Iterative refinement for symmetric eigenvalue decomposition II: Clustered eigenvalues, Jpn. Ind. Appl. Math., 36 (2019), pp. 435-459.

[16] N. J. Higham, Accuracy and Stability of Numerical Algorithms, 2nd edition, SIAM, Philadelphia, 2002.

[17] Tridiagonalization of a Hermitian or symmetric matrix based on Lapack interface, `https://www.mathworks.com/matlabcentral/fileexchange/47803-tridiagonalization-of-a-hermitian-or-symmetric-matrix-based-on-lapack-interface`.

[18] P. Bientinesi, I. S. Dhillon and R. A. van de Geijn, A parallel eigensolver for dense symmetric matrices based on multiple relatively robust representations, SIAM J. Sci. Comput., 27 (2005), pp. 43-66.