A mixed precision Jacobi method for the symmetric eigenvalue problem

Zhiyuan Zhang* Zheng-Jian Bai†

November 8, 2022

Abstract

The eigenvalue problem is a fundamental problem in scientific computing. In this paper, we propose a mixed precision Jacobi method for the symmetric eigenvalue problem. We first compute the eigenvalue decomposition of a real symmetric matrix by an eigensolver at low precision and we obtain a low-precision matrix of eigenvectors. Then by using the modified Gram-Schmidt orthogonalization process to the low-precision eigenvector matrix in high precision, a high-precision orthogonal matrix is obtained, which is used as an initial guess for the Jacobi method. We give the rounding error analysis for the proposed method and the quadratic convergence of the proposed method is established under some sufficient conditions. We also present a mixed precision one-side Jacobi method for the singular value problem and the corresponding rounding error analysis and quadratic convergence are discussed. Numerical experiments on CPUs and GPUs are conducted to illustrate the efficiency of the proposed mixed precision Jacobi method over the original Jacobi method.

Keywords. Symmetric eigenvalue problem, singular value problem, Jacobi's method, mixed precision arithmetic, modified Gram-Schmidt method

AMS subject classifications. 65F15, 15A18

1 Introduction

Let A be an $n \times n$ real symmetric matrix. The spectral decomposition of A is given by

$$A = P\Lambda P^T, \quad \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n), \tag{1.1}$$

where $P \in \mathbb{R}^{n\times n}$ is an orthogonal matrix whose ith column is the eigenvector of A associated with the eigenvalue λ_i for $i = 1, \ldots, n$. The eigenvalue problem has widespread applications in scientific computing such as engineering computing [11, 47], numerical partial differential equations (PDEs) [33], and computing chemistry [12], etc.

*School of Mathematical Sciences, Xiamen University, Xiamen 361005, People's Republic of China.

†Corresponding author. School of Mathematical Sciences, Xiamen University, Xiamen 361005, People's Republic of China (zjbai@xmu.edu.cn).

The solution strategy of the symmetric eigenvalue problem depends on the structure of the given symmetric matrix and the desired eigenvalues with or without associated eigenvectors. For example, when only a few extreme eigenvalues of a large, sparse, and symmetric matrix are desired, Lanczos method and Jacobi-Davidison method [22, 41] are recommended. In general, a symmetric matrix can be reduced to tridiagonal form by finite Householder reflections or Givens rotations and there are some popular tridiagonalization based strategies for the symmetric eigenvalue problem [22, 36], e.g., the symmetric QR algorithm [8], the divide-and-conquer method [23], the bisection method [6], Sturm sequence method [24] and the method of multiple relatively robust representations (MR^3) [16]. Compared with these tridiagonalization based algorithm, there is another method directly applied to the original real symmetric matrix, i.e., the Jacobi method. The Jacobi method is a very old method for diagonalizing a real symmetric matrix [31]. Recently, the Jacobi method has received much attention due to its natural suitability for parallel computation [7] and high accuracy in finite precision [15]. Another interesting aspect of Jacobi's method with proper procedure ordering shows sweep-quadratic convergence rate after sufficient iterations [38, 43]. However, the Jacobi method is slower than a method based on tridiagonalization since it is conjectured that $\mathcal{O}(n^3 \log n)$ operations are required for its standard implementation [40]. For an overview of the symmetric eigenvalue problem, one may refer to [36, 45] and [22, Chap.7, Chap,8, Chap.11].

To improve the efficiency of a numerical solver, in many engineering applications, one of the emerging strategies is to combine different precision arithmetics [5]. In the past decades, the most common IEEE 754 floating-point arithmetic in scientific computing has mainly been carried out in double precision (64 bit) and single precision (32 bit) [1]. Theoretically, single precision runs twice as fast as double precision in communication cost and computation cost. And these two formats are supported by most of hardware architectures [3]. Recently, half precision (16-bit) floating point arithmetic has gradually been popular in the machine learning community. Half precision arithmetic is already available in some hardware (e.g., the NVIDIA V100 GPU), which runs faster in machine learning applications and also reduces memory storage and energy consumption. For higher precisions, there exists quadruple precision (128 bit) in some softwares [30] such as Advanpix Multiprecision Computing Toolbox for `MATLAB` [2].

Recently, based on different floating point precisions, many mixed precision algorithms have been proposed [9, 46, 48]. For mixed precision algorithms in numerical linear algebra, one may refer to the two survey papers [3, 30]. For the general eigenvalue problem, an earlier work given by Dongarra et al. [17, 18] was the mixed precision iterative refinement based on Newton's method for computing eigenpairs of a matrix, which was extended to solving the symmetric eigenvalue problem by using the Sherman–Morrison formula [42]. For the symmetric eigenvalue problem, Petschow et al. [37] proposed a mixed precision MR^3-based eigensolver with improved accuracy and negligible performance penalty. Ogita and Aishima [34, 35] developed another novel iterative refinement for the symmetric eigenvalue decomposition. Very most recently, Gao et al. [25] proposed an elaborate mixed precision Jacobi singular value decomposition (SVD) algorithm which can achieve about 2x speedup compared to LAPACK in x86-64 architecture.

In this paper, we propose a mixed precision Jacobi method for the symmetric eigenvalue problem. By using an eigensolver to computing the eigenvalue decomposition of a real symmetric matrix at low precision, we can obtain a low-precision matrix of eigenvector. Then using the modified Gram-Schmidt (MGS) orthogonalization process to the low-precision eigenvector

matrix, a high-precision orthogonal matrix is obtained, which is employed as an initial guess for the Jacobi method. We give the rounding error analysis and some sufficient conditions are provided for guaranteeing the quadratic convergence of the proposed method. We also present a mixed precision one-side Jacobi method for the singular value problem and the corresponding rounding error analysis and quadratic convergence are discussed. Finally, we report some numerical experiments to illustrate the efficiency of the proposed method over the original Jacobi method.

Throughout this paper, we use the following notation. Let $\mathbb{R}^{m\times n}$ be the set of all m-by-n real matrices and $\mathbb{R}^n = \mathbb{R}^{n\times 1}$. I_n is an identity matrix of order n and $\mathbf{e}_s$ is the sth column of I_n. $\mathbf{1}_n$ is an n-vector of all ones. Let $|\cdot|$ be the absolute value of a real number. Let $\|\cdot\|$ and $\|\cdot\|_F$ be the Euclidean vector norm or its induced matrix norm and the Frobenius matrix norm, respectively. The symbol "$\otimes$" means the Kronecker product. The superscript "$.^T$" stands for the transpose of a matrix or vector. For a symmetric matrix $G \in \mathbb{R}^{n\times n}$, we denote by $\lambda_1(G) \geq \lambda_2(G) \geq \cdots \geq \lambda_n(G)$ its eigenvalues, arranged in decreasing order. For a matrix $G \in \mathbb{R}^{m\times n}$, we denote by $\sigma_1(G) \geq \sigma_2(G) \geq \cdots \geq \sigma_{\min\{m,n\}}(G) \equiv \sigma_{\min}(G) \geq 0$ its singular values, arranged in decreasing order. For a matrix $G = (g_{ij}) \in \mathbb{R}^{n\times n}$, let $\mathrm{off}(G) := G - \mathrm{diag}(g_{11}, \ldots, g_{nn})$ and $\mathbf{g}_j$ be the j-th column vector of G for $j = 1, \ldots, n$.

The rest of the paper is organized as follows. In Section 2 we review some error analysis results on classical numerical methods for the eigenvalue problem and the singular value problem. In Section 3 we briefly review the Jacobi method for the symmetric eigenvalue problem. In Section 4 we propose a mixed precision Jacobi method for the symmetric eigenvalue problem. We also discuss the rounding error analysis and quadratic convergence of the proposed method. In Section 5 we present a mixed precision one-sided Jacobi method for the singular value problem. In Section 6 we present some numerical tests to demonstrate the efficiency of the proposed methods. Some concluding remarks are given in Section 7.

2 Preliminaries

In this section, we review some error analysis results on some numerical methods for the symmetric eigenvalue problem and the singular value problem. We first recall the following error bounds for the MGS method [28, Theorem 19.13].

Lemma 2.1 *Let $A \in \mathbb{R}^{m\times n}$ with $\mathrm{rank}(A) = n$. Suppose the MGS method computes the approximate QR factorization $A \approx \hat{Q}\hat{R}$ in precision υ, where $\hat{R} \in \mathbb{R}^{n\times n}$ is upper triangular and $\hat{Q} \in \mathbb{R}^{m\times n}$. Then there exist constants $\eta_i \equiv \eta_i(m,n)$ for $i = 1, 2, 3$ such that*

$$\|A - \hat{Q}\hat{R}\| \leq \eta_1\|A\|\upsilon, \quad \|\hat{Q}^T\hat{Q} - I_n\| \leq \eta_2\kappa(A)\upsilon,$$

and $\hat{Q} + \delta\hat{Q}$ is orthogonal with $\|\delta\hat{Q}\| \leq \eta_3\kappa(A)\upsilon$, where $\kappa(A) = \sigma_1(A)/\sigma_{\min}(A)$ is the condition number of A.

On the error bounds for the symmetric eigenvalue problem, we have the following result [4, pp.104-105].

Lemma 2.2 *Let $A \in \mathbb{R}^{n\times n}$ be a real symmetric matrix. The computed symmetric eigenvalue decomposition* (SED) *$A \approx \hat{P}\hat{\Lambda}\hat{P}^T$ with $\hat{P} \in \mathbb{R}^{n\times n}$ and $\hat{\Lambda} = \mathrm{diag}(\hat{\lambda}_1, \ldots, \hat{\lambda}_n) \in \mathbb{R}^{n\times n}$ via any*

eigensolver in LAPACK or EISPACK in precision υ is nearly the exact symmetric Schur decomposition of $A+E$, i.e.,

$$A+E=(\hat{P}+\delta\hat{P})\hat{\Lambda}(\hat{P}+\delta\hat{P})^T, \tag{2.1}$$

where $\|E\|\le p(n)\,\|A\|\,\upsilon$ and $\hat{P}+\delta\hat{P}$ is orthogonal with $\|\delta\hat{P}\|\le p(n)\upsilon$. Here, $p(n)$ is a modestly growing function of n.

On the error bounds for the singular value problem, we have the following result [4, pp.112–113].

Lemma 2.3 *Let $A\in\mathbb{R}^{m\times n}$ be a real matrix ($m\ge n$). The computed SVD $A\approx\hat{U}\hat{\Sigma}\hat{V}^T$ with $\hat{U}\in\mathbb{R}^{m\times m}$, $\hat{V}\in\mathbb{R}^{n\times n}$, and $\hat{\Sigma}=\mathrm{diag}(\hat{\sigma}_1,\ldots,\hat{\sigma}_n)\in\mathbb{R}^{m\times n}$ via any SVD solver in LAPACK, LINPACK or EISPACK in precision υ is nearly the exact SVD of $A+E$, i.e.,*

$$A+E=(\hat{U}+\delta\hat{U})\hat{\Sigma}(\hat{V}+\delta\hat{V})^T \tag{2.2}$$

where $\|E\|\le p(m,n)\|A\|\upsilon$ and $\hat{U}+\delta\hat{U}$ and $\hat{V}+\delta\hat{V}$ are both orthogonal with $\|\delta\hat{U}\|\le p(m,n)\upsilon$ and $\|\delta\hat{V}\|\le p(m,n)\upsilon$. Here, $p(m,n)$ is a modestly growing function of m and n.

Finally, we recall the following perturbation bound for the eigenvalues of a real symmetric matrix [22, pp.442].

Lemma 2.4 *If A and $A+E$ are $n\times n$ real symmetric matrices, then*

$$|\lambda_k(A+E)-\lambda_k(A)|\le\|E\|,$$

for $k=1,\ldots,n$.

3 Jacobi's method

In this section, we briefly review Jacobi's method for computing the symmetric eigenvalue decomposition of a real symmetric matrix.

Let A be an $n\times n$ real symmetric matrix. The Jacobi method aims to construct a sequence of orthogonal updates $A^{(k+1)}=J_k^TA^{(k)}J_k$ such that the off-diagonal entries of $A^{(k+1)}$ are closer to zeros than $A^{(k)}$, where $A^{(0)}=A$ and J_k is a Jacobi rotation. When $\mathrm{off}(A^{(k)})$ is close to the zero matrix sufficiently, a computed eigenvalue decomposition of the original matrix A is available.

Define a Jacobi rotation $J(i,j;\theta)$ by

$$J(i,j;\theta)=I_n+[\mathbf{e}_i,\mathbf{e}_j]\begin{bmatrix}c-1 & s\\ -s & c-1\end{bmatrix}\begin{bmatrix}\mathbf{e}_i^T\\ \mathbf{e}_j^T\end{bmatrix}, \tag{3.1}$$

where $c=\cos\theta\in\mathbb{R}$ and $s\in\mathbb{R}$ with $c^2+s^2=1$. Then we have the following result [22, §8.5].

Lemma 3.1 *Let A be an $n\times n$ real symmetric matrix. Then, for any index pair (i,j) with $1\le i<j\le n$, there exists a Jacobi rotation $J=J(i,j;\theta)$ defined by (3.1) such that, for the updated matrix $B=J^TAJ$,*

$$b_{ij}=b_{ij}=0,\quad b_{ii}^2+b_{jj}^2=a_{ii}^2+a_{jj}^2+2a_{ij}^2,$$

where $c=\cos\theta=(1+t^2)^{-1/2}$ and $s=tc$ with $|\theta|\le\pi/4$. Here, $t=1/(\mu+\sqrt{1+\mu^2})$ if $\mu\ge0$ and $t=1/(\mu-\sqrt{1+\mu^2})$ if $\mu<0$, where $\mu=(a_{jj}-a_{ii})/(2a_{ij})$. If $a_{ij}=0$, then we set $(c,s)=(1,0)$.

From Lemma 3.1, we observe that the updated matrix $B = J^T AJ$ agrees with A except in rows and columns i and j and

$$\|\mathrm{off}(B)\|_F^2 = \|B\|_F^2 - \sum_{i=1}^{n} b_{ii}^2 = \|A\|_F^2 - \sum_{i=1}^{n} a_{ii}^2 + (a_{ii}^2 + a_{jj}^2 - b_{ii}^2 - b_{jj}^2) = \|\mathrm{off}(A)\|_F^2 - 2a_{ij}^2. \quad (3.2)$$

To minimize $\|\mathrm{off}(B)\|_F$, a classical strategy is to choose the index (i, j) such that the off-diagonal element a_{ij} has the largest absolute value, i.e., $|a_{ij}| = \max_{p\neq q} |a_{pq}|$. This leads to the classical Jacobi algorithm, which is stated in Algorithm 3.1.

Algorithm 3.1 Classical Jacobi's method for the symmetric eigenvalue decomposition.

Require: A symmetric matrix $A \in \mathbb{R}^{n\times n}$ and a tolerance $\epsilon > 0$. Let $P = I_n$.

1: **while** $\|\mathrm{off}(A)\|_F > \epsilon \|A\|_F$ **do**
2: Choose the index (i, j) such that $|a_{ij}| = \max_{p\neq q} |a_{pq}|$.
3: Compute a cosine-sine group (c, s) as in Lemma 3.1.
4: Set $A = J(i, j, \theta)^T AJ(i, j, \theta)$ and $P = PJ(i, j, \theta)$.
5: **end while**

Let $A^{(k)}$ be the matrix A after k Jacobi updates. Then Algorithm 3.1 converges linearly in the sense that [22, §8.5]

$$\|\mathrm{off}(A^{(k)})\|_F^2 \leq \left(1 - \frac{1}{N}\right)^k \|\mathrm{off}(A^{(0)})\|_F^2,$$

where $N = n(n-1)/2$. Here, we refer to N Jacobi updates as a *sweep*. However, the quadratic convergence of Algorithm 3.1 is established in [39] in the sense that

$$\|\mathrm{off}(A^{(k+N)})\|_F \leq \alpha \cdot \|\mathrm{off}(A^{(k)})\|_F^2 \quad \text{for some constant } \alpha > 0.$$

We note that it is expensive to find the optimal index (i, j). A feasible alternative is to update A by rows or columns. This is the so-called *cyclic Jacobi* algorithm. Here, we describe a special row-cyclic Jacobi algorithm as Algorithm 3.2 [21].

Remark 3.2 *When A is positive definite, one may replace the stopping criterion in Line 4 of Algorithm 3.2 by $|a_{ij}| \leq \nu\sqrt{a_{ii}a_{jj}}$ for achieving accuracy proportional to the condition number of A [15].*

On Algorithm 3.2, we have the following quadratic convergence [43, 44].

Lemma 3.3 *Let $A^{(k)}$ be the matrix A after k Jacobi updates in Algorithm 3.2 with $A^{(0)} = A$. Suppose*

$$\min_{\lambda_i(A^{(0)})\neq\lambda_j(A^{(0)})} |\lambda_i(A^{(0)}) - \lambda_j(A^{(0)})| \geq d > 0.$$

Then we have

$$\|\mathrm{off}(T^{(N)})\|_F \leq \frac{\sqrt{34/9}}{d}\|\mathrm{off}(A^{(k)})\|_F^2$$

provided that $\|\mathrm{off}(A^{(k)})\|_F < d/(4\sqrt{2})$, where $N = n(n-1)/2$.

One may also refer to [26, 44] for the quadratic convergence of the general cyclic Jacobi method for a real symmetric matrix with distinct eigenvalues.

Algorithm 3.2 Cyclic Jacobi's method for the symmetric eigenvalue decomposition.

Require: A symmetric matrix $A \in \mathbb{R}^{n\times n}$ and a tolerance $\epsilon > 0$ and threshold $\nu > 0$. Let $P = I_n$.

while $\|\mathrm{off}(A)\|_F > \epsilon\|A\|_F$ **do**
  for $i = 1, \ldots, n-1$ **do**
    for $j = i+1, \ldots, n$ **do**
      if $|a_{ij}| \le \nu \min\{|a_{ii}|, |a_{jj}|\}$ **then**
        Set $a_{ij} = 0$ and $a_{ji} = 0$.
      else
        Compute a cosine-sine group (c, s) as in Lemma 3.1.
        Set $A = J(i, j, \theta)^T A J(i, j, \theta)$ and $P = PJ(i, j, \theta)$.
      end if
    end for
  end for
end while

4 A mixed precision Jacobi method for the SED

In this section, we propose a mixed precision Jacobi algorithm for computing the eigenvalue decomposition of an n-by-n real symmetric matrix A. We first compute an approximate eigenvalue decomposition $A \approx ZDZ^T$ in low precision υ, where $Z \in \mathbb{R}^{n\times n}$ and $D = \mathrm{diag}(d_1, \ldots, d_n)$. To improve the orthogonality of Z, we use the MGS method to Z in high precision $\omega \ll \upsilon$ and we obtain a high accuracy orthogonal matrix $Q \in \mathbb{R}^{n\times n}$. One may employ Householder orthogonalization to orthogonalize Z, which is less computationally efficient than MGS [22, §5.2.9]. Then we use the orthogonal matrix Q as the initial guess for the Jacobi method for computing the eigenvalue decomposition of A.

Based on the above analysis, we present our mixed precision Jacobi algorithm in Algorithm 4.1.

We observe from Lemma 3.3 that, to take advantage of the quadratic convergence of the Jacobi method as much as possible, it is desired that the off-diagonal entries of the matrix $Q^T AQ$ in Step 3 of Algorithm 4.1 is small enough.

In the rest of this section, we provide an error bound for the Frobenius norm of the off-diagonal entries of the matrix $Q^T AQ$ and give some sufficient conditions to guarantee the quadratic convergence of Algorithm 4.1. We first give an error bound between Z and Q, which are generated in Steps 1–2 of Algorithm 4.1.

Lemma 4.1 *Suppose that $Z \in \mathbb{R}^{n\times n}$ in Step 1 of Algorithm 4.1 is computed by any eigensolver in LAPACK or EISPACK in precision υ and $Q \in \mathbb{R}^{n\times n}$ in Step 2 of Algorithm 4.1 is computed by using the MGS method to Z in precision ω ($\omega \ll \upsilon$). Then there exist constants $h_i \equiv h_i(n)$ for $i = 1, 2$ such that*

$$\|Z - Q\|_F \le h_1\upsilon + h_2\omega.$$

Proof. By Lemma 2.2, there exists a matrix $\delta Z \in \mathbb{R}^{n\times n}$ such that $Z + \delta Z$ is orthogonal, where

Algorithm 4.1 A mixed precision Jacobi method for the SED.

Require: A symmetric matrix $A \in \mathbb{R}^{n\times n}$, a tolerance $\epsilon > 0$ and a threshold $\nu > 0$.
Ensure: The orthogonal matrix $P \in \mathbb{R}^{n\times n}$ and the symmetric matrix $T \in \mathbb{R}^{n\times n}$ with $\|\mathrm{off}(T)\|_F \le \epsilon\|A\|_F$.

$[Z, \sim] = \mathrm{eig}\,(A)$ ▷ Symmetric QR factorization in low precision υ
$Q = \mathrm{MGS}\,(Z)$ ▷ Modified Gram-Schmidt orthogonalization in high precision ω
Set $T = Q^T AQ$ and $P = Q$
while $\|\mathrm{off}(T)\|_F > \epsilon\|A\|_F$ **do**
  for $i = 1, \ldots n-1$ **do**
    for $j = i+1, \ldots, n$ **do**
      if $|t_{ij}| \le \nu \min\{|t_{ii}|, |t_{jj}|\}$ **then**
        Set $t_{ij} = 0$ and $t_{ji} = 0$.
      else
        Compute a cosine-sine group (c, s) as in Lemma 3.1 with $A = T$.
        Set $T = J(i, j, \theta)^T TJ(i, j, \theta)$ and $P = PJ(i, j, \theta)$.
      end if
    end for
  end for
end while

$\|\delta Z\| \le p(n)\upsilon$. Thus,

$$\|Z\| \le \|Z + \delta Z\| + \|\delta Z\| = 1 + \|\delta Z\| \le 1 + p(n)\upsilon \equiv c_1. \tag{4.1}$$

This, together with the orthogonality of $Z + \delta Z$, yields

$$\begin{cases} \lambda_{\max}(Z^T Z) \le 1 + (2\|\delta Z\|\|Z\| + \|\delta Z\|^2) \le 1 + \beta_1, \\ \lambda_{\min}(Z^T Z) \ge 1 - \left(2\|\delta Z\|\|Z\| + \|\delta Z\|^2\right) \ge 1 - \beta_1, \end{cases}$$

provided that $\beta_1 = (2c_1 + p(n)\upsilon)p(n)\upsilon < 1$. Hence, Z is nonsingular and $\mathrm{rank}(Z) = n$. In addition,

$$\kappa(Z) = \frac{\sigma_{\max}(Z)}{\sigma_{\min}(Z)} = \sqrt{\frac{\lambda_{\max}(Z^T Z)}{\lambda_{\min}(Z^T Z)}} \le \sqrt{\frac{1+\beta_1}{1-\beta_1}}. \tag{4.2}$$

It follows from Lemma 2.1 and (4.1) that there exists an upper triangular matrix $R \in \mathbb{R}^{n\times n}$ such that

$$Z = QR + F_1, \quad \|F_1\| \le \eta_1 \|Z\|\, \omega \le \eta_1 c_1 \omega \equiv c_2\omega, \tag{4.3}$$

where Q is such that

$$Q^T Q = I + F_2, \quad \|F_2\| \le \eta_2 \kappa(Z)\omega \le \sqrt{\frac{1+\beta_1}{1-\beta_1}} \cdot \eta_2\omega \equiv c_3\omega. \tag{4.4}$$

We now give an upper bound and a lower bound to each diagonal entry of R. From (4.4) we have

$$\|Q\| \le \sqrt{1 + \|F_2\|} \le 1 + \|F_2\|\,. \tag{4.5}$$

Using (4.3) and (4.4) we have

$$Q^T Z = R + F_2 R + Q^T F_1,$$

which implies that

$$\|R\| \le \|Q\| \, \|Z\| + \|F_2\| \, \|R\| + \|Q\| \, \|F_1\| \, .$$

Thus,

$$\begin{aligned} \|R\| &\le \frac{\|Q\|}{1 - \|F_2\|} \left(\|Z\| + \|F_1\|\right) \le \frac{1 + \|F_2\|}{1 - \|F_2\|} \left(\|Z\| + \|F_1\|\right) \\ &\le \left(1 + \frac{2\,\|F_2\|}{1 - \|F_2\|}\right) \left(\|Z\| + \|F_1\|\right) \\ &\le \left(1 + \frac{2\beta_2}{1 - \beta_2}\right)(1 + \eta_1\omega)(1 + p(n)\upsilon) \equiv 1 + g_1\upsilon, \end{aligned} \tag{4.6}$$

provided that $\beta_2 = c_3\omega < 1$, where

$$g_1 = \eta_1\omega\upsilon^{-1} + (1 + \eta_1\omega)p(n) + \frac{2c_3\omega\upsilon^{-1} + 2\eta_1\beta_2\omega\upsilon^{-1} + 2\beta_2 p(n) + 2\eta_1\beta_2 p(n)\omega}{1 - \beta_2}.$$

Similarly, we have by (4.3),

$$(Z + \delta Z)R^{-1} = Q + (F_1 + \delta Z)R^{-1}.$$

This, together with the orthogonality of $Z + \delta Z$, yields

$$\|R^{-1}\| = \|(Z + \delta Z)R^{-1}\| \le \|Q\| + \left(\|F_1\| + \|\delta Z\|\right) \|R^{-1}\|.$$

Using (4.3)–(4.5) we have

$$\|R^{-1}\| \le \frac{\|Q\|}{1 - \left(\|F_1\| + \|\delta Z\|\right)} \le \frac{1 + \|F_2\|}{1 - \left(\|F_1\| + \|\delta Z\|\right)} \le \frac{1 + c_3\omega}{1 - \beta_3} \equiv 1 + g_2\upsilon, \tag{4.7}$$

provided that $\beta_3 = c_2\omega + p(n)\upsilon < 1$, where $g_2 = ((c_2 + c_3)\omega\upsilon^{-1} + p(n))/(1 - \beta_3)$.

If $g_2\upsilon < 1$, then

$$1 - g_2\upsilon \le \frac{1}{1 + g_2\upsilon} \le \frac{1}{\|R^{-1}\|} \le r_{ii} \le \|R\| \le 1 + g_1\upsilon, \ i = 1, 2, \ldots, n. \tag{4.8}$$

Next, we bound the spectral norm of the strictly upper triangular part of R. Substituting (4.3) and (4.4) into $(Z + \delta Z)^T(Z + \delta Z) = I$ yields

$$I - R^T R = R^T F_2 R + (F_1 + \delta Z)^T Q R + R^T Q^T (F_1 + \delta Z) + (F_1 + \delta Z)^T (F_1 + \delta Z).$$

Then, by (4.3), (4.4), (4.5), and (4.6) we have

$$\begin{aligned} \|I - R^T R\| &\le \|R\|^2 \|F_2\| + 2\left(\|F_1\| + \|\delta Z\|\right) \|Q\| \|R\| + \left(\|F_1\| + \|\delta Z\|\right)^2 \\ &\le (1 + g_1\upsilon)^2 c_3\omega + 2(c_2\omega + p(n)\upsilon)\sqrt{1 + \beta_2}(1 + g_1\upsilon) + (c_2\omega + p(n)\upsilon)^2 \\ &\equiv c_4\upsilon, \end{aligned}$$

where $c_4 = c_3(1+g_1\upsilon)^2\omega\upsilon^{-1} + 2(c_2\omega\upsilon^{-1}+p(n))\sqrt{1+\beta_2}(1+g_1\upsilon) + \beta_3(c_2\omega\upsilon^{-1}+p(n))$. This, together with (4.7), yileds

$$\left\|R^{-1}-R^T\right\| \le \left\|I-R^TR\right\|\left\|R^{-1}\right\| \le c_4(1+g_2\upsilon)\upsilon \equiv c_5\upsilon,$$

which implies that

$$|r_{ij}| \le \left\|R^{-1}-R^T\right\| \le c_5\upsilon \quad \forall i<j. \tag{4.9}$$

It follows from (4.8) and (4.9) that

$$\|R-I\|_F = \sqrt{\sum_{i=1}^{n}(r_{ii}-1)^2 + \sum_{i<j} r_{ij}^2} \le \upsilon\sqrt{n(\max\{g_1,g_2\})^2 + \frac{n(n-1)}{2}c_5^2} \equiv c_6\upsilon.$$

Therefore, by (4.3), (4.4), and (4.5) we have

$$\begin{aligned}\|Z-Q\|_F &\le \|Q\|\,\|R-I\|_F + \|F_1\|_F \le \sqrt{1+\|F_2\|}\,\|R-I\|_F + \sqrt{n}\,\|F_1\| \\ &\le \sqrt{1+c_3\omega}\cdot c_6\upsilon + \sqrt{n}\cdot c_2\omega \equiv h_1\upsilon + h_2\omega.\end{aligned}$$

□

On the Frobenius norm of the off-diagonal entries of the matrix Q^TAQ generated in Step 3 of Algorithm 4.1, we have the following result.

Theorem 4.2 *Suppose that $Z \in \mathbb{R}^{n\times n}$ in Step 1 of Algorithm 4.1 is computed by any eigensolver in LAPACK or EISPACK in precision υ and $Q \in \mathbb{R}^{n\times n}$ in Step 2 of Algorithm 4.1 is computed by using the MGS method to Z in precision ω ($\omega \ll \upsilon$). Then there exists a constant $\gamma_1 \equiv \gamma_1(n)$ such that*

$$\left\|\mathrm{off}(Q^TAQ)\right\|_F \le \gamma_1\|A\|\upsilon.$$

Proof. By Lemma 2.2 we know there exists a symmetric matrix $E \in \mathbb{R}^{n\times n}$ such that $A+E$ admits the following exact eigenvalue decomposition:

$$A+E = (Z+\delta Z)D(Z+\delta Z)^T, \quad D = \mathrm{diag}(d_1,\ldots,d_n) \in \mathbb{R}^{n\times n}, \tag{4.10}$$

where $\|E\| \le p(n)\|A\|\upsilon$ and $Z+\delta Z$ is orthogonal with $\|\delta Z\| \le p(n)\upsilon$. We note that Q is computed by using the MGS method to Z with precision ω. It follows that

$$\begin{aligned}&Q^TAQ - D = (Q-Z+Z)^T A\,(Q-Z+Z) - D \\ =\ &(Q-Z)^T A\,(Q-Z) + (Q-Z)^T AZ + Z^T A\,(Q-Z) + (Z^TAZ - D).\end{aligned} \tag{4.11}$$

On the other hand, using (4.10) we have

$$D = (Z+\delta Z)^T(A+E)(Z+\delta Z),$$

i.e.,

$$D - Z^TAZ = Z^TA\delta Z + \delta Z^TAZ + \delta Z^TA\delta Z + (Z+\delta Z)^TE(Z+\delta Z).$$

Thus,

$$\begin{aligned}\left\|Z^TAZ-D\right\|_F &\leq 2\|A\|_F\|Z\|\|\delta Z\|+\|A\|_F\|\delta Z\|^2+\|E\|_F\\ &\leq \|A\|_F(2\|Z\|+\|\delta Z\|)\|\delta Z\|+\|E\|_F.\end{aligned}$$

By Lemma 4.1 and using (4.1) and (4.11), we have

$$\begin{aligned}&\left\|\mathrm{off}(Q^TAQ)\right\|_F \leq \left\|Q^TAQ-D\right\|_F\\ \leq\ & \|(Q-Z)^TA(Q-Z)\|_F+2\|(Q-Z)^TAZ\|_F+\|Z^TAZ-D\|_F\\ \leq\ & \|A\|\|Q-Z\|_F^2+2\|A\|\|Z\|\|Q-Z\|_F+\|A\|_F(2\|Z\|+\|\delta Z\|)\|\delta Z\|+\|E\|_F\\ \leq\ & \|A\|\|Q-Z\|_F(\|Q-Z\|_F+2\|Z\|)+\sqrt{n}\|A\|(2\|Z\|+\|\delta Z\|)\|\delta Z\|+\sqrt{n}\|E\|\\ \leq\ & (2c_1+h_1\upsilon+h_2\omega)\|A\|(h_1\upsilon+h_2\omega)+\sqrt{n}\|A\|(2c_1+p(n)\upsilon)p(n)\upsilon+\sqrt{n}\|A\|p(n)\upsilon\\ \equiv\ & \gamma_1\|A\|\upsilon, \end{aligned}\tag{4.12}$$

where $\gamma_1=(2c_1+h_1\upsilon+h_2\omega)(h_1+h_2\omega\upsilon^{-1})+\sqrt{n}p(n)(1+2c_1+p(n)\upsilon)$. □

The following theorem provides a sufficient condition for the quadratic convergence of Algorithm 4.1.

Theorem 4.3 *Let $T^{(k)}$ be the matrix T after k Jacobi updates in Algorithm 4.1 with $T^{(0)}=Q^TAQ$. Suppose $\omega \ll \upsilon$ and*

$$\min_{\lambda_i(T^{(0)})\neq\lambda_j(T^{(0)})}|\lambda_i(T^{(0)})-\lambda_j(T^{(0)})|\geq d>0.$$

If $\gamma_1\|A\|\upsilon < d/(4\sqrt{2})$ for some constant $\gamma_1=\gamma_1(n)$, then we have

$$\|\mathrm{off}(T^{(N)})\|_F \leq \frac{\sqrt{34/9}}{d}\|\mathrm{off}(T^{(0)})\|_F^2,$$

where $N=n(n-1)/2$.

Proof. By Theorem 4.2 we have

$$\|\mathrm{off}(T^{(0)})\|_F=\left\|\mathrm{off}(Q^TAQ)\right\|_F\leq\gamma_1\|A\|\upsilon$$

for some constant $\gamma_1\equiv\gamma_1(n)$. Therefore, we have $\|\mathrm{off}(T^{(0)})\|_F < d/(4\sqrt{2})$ provided that $\gamma_1\|A\|\upsilon < d/(4\sqrt{2})$. The theorem is established by using Lemma 3.3. □

In Theorem 4.3, the quadratic convergence of Algorithm 4.1 is established under the assumption that $\gamma_1\|A\|\upsilon < d/(4\sqrt{2})$ for some constant $\gamma_1=\gamma_1(n)$ and there exists a clear gap between the distinct eigenvalues of the matrix $T^{(0)}=Q^TAQ$. In the following theorem, we establish the quadratic convergence of Algorithm 4.1 under the assumption that there exists a clear gap between the distinct eigenvalues of the original matrix A.

Theorem 4.4 *Let $T^{(k)}$ be the matrix T after k Jacobi updates in Algorithm 4.1 with $T^{(0)} = Q^T AQ$. Suppose $\omega \ll \upsilon$ and*

$$\min_{\lambda_i(A))\neq\lambda_j(A)} |\lambda_i(A) - \lambda_j(A)| \geq d > 0.$$

If $2\gamma_2\|A\|\omega \leq \rho_1 d$ for some constant $\gamma_2 = \gamma_2(n)$ and $0 < \rho_1 < 1$, and $\gamma_1\|A\|\upsilon < (1-\rho_1)d/(4\sqrt{2})$ for some constant $\gamma_1 = \gamma_1(n)$, then we have

$$\|\mathrm{off}(T^{(N)})\|_F \leq \frac{\sqrt{34/9}}{(1-\rho_1)d}\|\mathrm{off}(T^{(0)})\|_F^2.$$

Proof. By Lemma 2.1, there exists a matrix $\delta Q \in \mathbb{R}^{n\times n}$ such that $Q + \delta Q$ is orthogonal with

$$\|\delta Q\| \leq \eta_3\kappa(Z)\omega \leq \sqrt{\frac{1+\beta_1}{1-\beta_1}} \cdot \eta_3\omega \equiv \eta_4\omega, \tag{4.13}$$

where the second inequality follows from (4.2). We note that

$$(Q+\delta Q)^T A(Q+\delta Q) = Q^T AQ + Q^T A\delta Q + \delta Q^T AQ + \delta Q^T A\delta Q.$$

By hypothesis, $T^{(0)} = Q^T AQ$. It follows from Lemma 2.4, (4.4) and (4.5) that, for any $1 \leq k \leq n$,

$$\begin{aligned} |\lambda_k(A) - \lambda_k(T^{(0)})| &\leq 2\|A\|\|Q\|\|\delta Q\| + \|A\|\|\delta Q\|^2 \leq \|A\|(2\sqrt{1+\|F_2\|} + \|\delta Q\|)\|\delta Q\| \\ &\leq \|A\|(2\sqrt{1+c_3\omega} + \eta_4\omega)\eta_4\omega \equiv \gamma_2\|A\|\omega. \end{aligned}$$

Let $\lambda_i(A)$ and $\lambda_j(A)$ be arbitrary two distinct eigenvalues of A. Then we have

$$\begin{aligned} |\lambda_i(A) - \lambda_j(A)| &\leq |\lambda_i(A) - \lambda_i(T^{(0)})| + |\lambda_i(T^{(0)}) - \lambda_j(T^{(0)})| + |\lambda_j(T^{(0)}) - \lambda_j(A)| \\ &\leq |\lambda_i(T^{(0)}) - \lambda_j(T^{(0)})| + 2\gamma_2\|A\|\omega, \\ |\lambda_i(A) - \lambda_j(A)| &\geq ||\lambda_i(T^{(0)}) - \lambda_j(T^{(0)})| - |\lambda_i(A) - \lambda_i(T^{(0)})| - |\lambda_j(T^{(0)}) - \lambda_j(A)| \\ &\geq |\lambda_i(T^{(0)}) - \lambda_j(T^{(0)})| - 2\gamma_2\|A\|\omega. \end{aligned}$$

If $2\gamma_2\|A\|\omega \leq \rho_1 d$ for some $0 < \rho_1 < 1$, then we have

$$\frac{1}{1+\rho_1}|\lambda_i(T^{(0)}) - \lambda_j(T^{(0)})| \leq |\lambda_i(A) - \lambda_j(A)| \leq \frac{1}{1-\rho_1}|\lambda_i(T^{(0)}) - \lambda_j(T^{(0)})|.$$

By hypothesis, $\min_{\lambda_i(A))\neq\lambda_j(A)} |\lambda_i(A) - \lambda_j(A)| \geq d > 0$. Thus,

$$\min_{\lambda_i(T^{(0)})\neq\lambda_j(T^{(0)})} |\lambda_i(T^{(0)}) - \lambda_j(T^{(0)})| \geq (1-\rho_1) \min_{\lambda_i(A))\neq\lambda_j(A)} |\lambda_i(A) - \lambda_j(A)| \geq (1-\rho_1)d.$$

The theorem follows from Theorem 4.3 provided that $\gamma_1\|A\|\upsilon < (1-\rho_1)d/(4\sqrt{2})$. □

In Theorem 4.4, we need an additional condition that $2\gamma_2\|A\|\omega \leq \rho_1 d$ for some constant $\gamma_2 = \gamma_2(n)$ and $0 < \rho_1 < 1$. In fact, it is much weaker than the condition that $\gamma_1\|A\|\upsilon < d/(4\sqrt{2})$ for some constant $\gamma_1 = \gamma_1(n)$ since $\omega \ll \upsilon$. Therefore, the later is an essential condition for the quadratic convergence of Algorithm 4.1 under the assumption that there exists a clear gap between the distinct eigenvalues of the original matrix A.

5 A mixed precision one-side Jacobi method for the SVD

In this section, we propose a mixed precision one-side Jacobi method for computing the SVD of a real matrix. As we know, the one-side Jacobi method for the singular value problem was originally mentioned in [27].

We first describe the one-side Jacobi algorithm. Let A be an m-by-n real matrix ($m \geq n$). The one-side Jacobi algorithm aims to o construct a sequence of orthogonal updates $A^{(k+1)} = A^{(k)}J_k$ such that the columns of $A^{(k+1)}$ are mutually orthogonal sufficiently, where $A^{(0)} = A$ and J_k is a Jacobi rotation defined by (3.1). Then, the computed SVD of the original matrix A is available by columns scaling of the updated $A^{(k+1)}$.

On how to orthogonalize two columns of A, we have the following result [14].

Lemma 5.1 *Let A be an $m\times n$ real matrix ($m \geq n$). For any index pair (i, j) with $1 \leq i < j \leq n$, if $\mathbf{a}_i^T\mathbf{a}_j \neq 0$, then there exists a Jacobi rotation $J = J(i, j; \theta)$ defined by (3.1) such that, for the updated matrix $B = AJ$,*

$$\mathbf{b}_i^T\mathbf{b}_j = 0, \quad b_{ii}^2 + b_{jj}^2 = a_{ii}^2 + a_{jj}^2 + 2a_{ij}^2,$$

where $c = \cos\theta = (1 + t^2)^{-1/2}$ and $s = tc$ with $|\theta| \leq \pi/4$. Here, $t = 1/(\mu + \sqrt{1 + \mu^2})$ if $\mu \geq 0$ and $t = 1/(\mu - \sqrt{1 + \mu^2})$ if $\mu < 0$, where $\mu = (\mathbf{a}_j^T\mathbf{a}_j - \mathbf{a}_i^T\mathbf{a}_i)/(2\mathbf{a}_i^T\mathbf{a}_j)$.

In fact, the Jacobi rotation $J = J(i, j; \theta)$ defined by Lemma 5.1 is such that the off-diagonal entries (i, j) and (j, i) in the symmetric matrix $B^TB = J^TA^TAJ$ are zeros. Here, a special row-cyclic one-side Jacobi algorithm for the SVD is sated as Algorithm 5.1.

Algorithm 5.1 Cyclic one-side Jacobi's method for the SVD.

Require: A matrix $A \in \mathbb{R}^{m\times n}$ ($m \geq n$) with $\mathrm{rank}(A) = n$, a tolerance $\epsilon > 0$, and a threshold $\nu > 0$. Let $V = I_n$.

```
while ‖off(A^T A)‖_F > ε‖A^T A‖_F do
  for i = 1, ..., n − 1 do
    for j = i + 1, ..., n do
      if |a_i^T a_j| > ν‖a_i‖‖a_j‖ then
        Compute a cosine-sine group (c, s) as in Lemma 5.1.
        Set A = AJ(i, j, θ) and V = VJ(i, j, θ).
      end if
    end for
  end for
end while
```

For a reliable and accurate implementation of Algorithm 5.1, one may refer to [19].

As in Section 4, we propose a mixed precision one-sided Jacobi algorithm for computing the SVD of a real matrix with full column rank, which is stated as Algorithm 5.2.

Remark 5.2 *In Algorithms 5.1–5.2, the given matrix A is assumed to be of full column rank. In fact, these algorithms can be used to compute the SVD of a general matrix [27]. When $m \gg n$, one may use the rank-revealing QR factorization as a preconditioner for these algorithms [20].*

Algorithm 5.2 A mixed precision one-side Jacobi method for the SVD.

Require: A matrix $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ with $\mathrm{rank}(A) = n$, a tolerance $\epsilon > 0$, and a threshold $\nu > 0$.

Ensure: The orthonormal matrix $U \in \mathbb{R}^{m\times n}$ and the orthogonal matrix $V \in \mathbb{R}^{n\times n}$ and the diagonal matrix $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$.

1: $[Y, \sim, Z] = \mathrm{svd}\,(A)$ ▷ SVD in low precision υ
2: $Q = \mathrm{MGS}\,(Z)$ ▷ Modified Gram-Schmidt orthogonalization in high precision ω
3: Set $C = AQ$ and $V = Q$ ▷ Matrix multiplication
4: **while** $\|\mathrm{off}(C^TC)\|_F > \epsilon\|C^TC\|_F$ **do**
5: **for** $i = 1, \ldots, n-1$ **do**
6: **for** $j = i+1, \ldots, n$ **do**
7: **if** $|\mathbf{c}_i^T\mathbf{c}_j| > \nu\|\mathbf{c}_i\|\|\mathbf{c}_j\|$ **then**
8: Compute a cosine-sine group (c, s) as in Lemma 5.1 with $A = C$.
9: Set $C = CJ(i, j, \theta)$ and $V = VJ(i, j, \theta)$.
10: **end if**
11: **end for**
12: **end for**
13: **end while**
14: Reorder the columns of $[C^T, V^T]^T$ with $\|\mathbf{c}_1\| \geq \cdots \geq \|\mathbf{c}_n\| > 0$.
15: Compute $\sigma_j = \|\mathbf{c}_j\|$ and $\mathbf{u}_j = \mathbf{c}_j/\sigma_j$ for $j = 1, \ldots, n$.
16: Set $U = [\mathbf{u}_1, \ldots, \mathbf{u}_n]$, $V = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$, and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$.

In the rest of this section, we give an error analysis to the orthogonalization of the columns of AQ generated in Step 3 of Algorithm 5.2 and provide some sufficient conditions to guarantee the quadratic convergence of Algorithm 5.2. We first give an error bound between Z and Q, which are generated in Steps 1–2 of Algorithm 5.2.

Lemma 5.3 *Suppose that $Z \in \mathbb{R}^{n\times n}$ in Step 1 of Algorithm 5.2 is computed by any svd solver in LAPACK, LINPACK or EISPACK in precision υ and $Q \in \mathbb{R}^{n\times n}$ in Step 2 of Algorithm 5.2 is computed by using the MGS method to Z in precision ω $(\omega \ll \upsilon)$. Then there exist constants $f_i \equiv f_i(m, n)$ for $i = 1, 2$ such that*

$$\|Z - Q\|_F \leq f_1\upsilon + f_2\omega.$$

Proof. The lemma follows from the arguments similar to that of Lemma 4.1. □

On the orthogonalization of the columns of AQ generated in Step 3 of Algorithm 5.2, we have the following result.

Theorem 5.4 *Suppose that $Z \in \mathbb{R}^{n\times n}$ in Step 1 of Algorithm 5.2 is computed by any svd solver in LAPACK, LINPACK or EISPACK in precision υ and $Q \in \mathbb{R}^{n\times n}$ in Step 2 of Algorithm 5.2 is computed by using the MGS method to Z in precision ω $(\omega \ll \upsilon)$. Then there exists a constant $\zeta_1 \equiv \zeta_1(m, n)$ such that*

$$\|\mathrm{off}((AQ)^T(AQ))\|_F \leq \zeta_1\|A\|^2\upsilon.$$

Proof. By Lemma 2.3, there exists a matrix $E \in \mathbb{R}^{m\times n}$ such that $A+E$ admits the following exact SVD:

$$A + E = (Y + \delta Y)S(Z + \delta Z)^T, \quad S = \mathrm{diag}(s_1, \ldots, s_n) \in \mathbb{R}^{m\times n}, \tag{5.1}$$

where $\|E\| \le p(m,n)\|A\|\upsilon$ and $Y+\delta Y$ and $Z+\delta Z$ are both orthogonal with $\|\delta Y\| \le p(m,n)\upsilon$ and $\|\delta Z\| \le p(m,n)\upsilon$. Using the orthogonality of $Z+\delta Z$ we have

$$\|Z\| \le \|Z + \delta Z\| + \|\delta Z\| = 1 + \|\delta Z\| \le 1 + p(m,n)\upsilon \equiv \tau_1. \tag{5.2}$$

We note that Q is computed by using the MGS method to Z in precision ω. Then we have

$$\begin{aligned} & Q^T A^T AQ - S^T S = (Q - Z + Z)^T A^T A(Q - Z + Z) - S^T S \\ = \;& (Q - Z)^T A^T A(Q - Z) + (Q - Z)^T A^T AZ + Z^T A^T A(Q - Z) + (Z^T A^T AZ - S^T S) \end{aligned} \tag{5.3}$$

On the other hand, it follows from (5.1) that

$$S^T S = (Z + \delta Z)^T (A + E)^T (A + E)(Z + \delta Z),$$

i.e.,

$$\begin{aligned} S^T S - Z^T A^T AZ \;=\; & Z^T A^T A\delta Z + \delta Z^T A^T AZ + \delta Z^T A^T A\delta Z \\ & +(Z + \delta Z)^T (A^T E + E^T A + E^T E)(Z + \delta Z). \end{aligned}$$

Thus,

$$\|Z^T A^T AZ - S^T S\| \le 2\|A\|^2\|Z\|\|\delta Z\| + \|A\|^2\|\delta Z\|^2 + 2\|A\|\|E\| + \|E\|^2.$$

By Lemma 5.3 and using (5.2) and (5.3) we obtain

$$\begin{aligned} & \|\mathrm{off}(Q^T A^T AQ)\|_F \le \|Q^T A^T AQ - S^T S\|_F \\ \le\; & \|A\|^2\|Q - Z\|_F^2 + 2\|A\|^2\|Z\|\|Q - Z\|_F + \|Z^T A^T AZ - S^T S\|_F \\ \le\; & \|A\|^2\|Q - Z\|_F(\|Q - Z\|_F + 2\|Z\|) + \sqrt{n}\|Z^T A^T AZ - S^T S\| \\ \le\; & \|A\|^2\|Q - Z\|_F(\|Q - Z\|_F + 2\|Z\|) + \sqrt{n}\|A\|^2(2\|Z\| + \|\delta Z\|)\|\delta Z\| + \sqrt{n}(2\|A\| + \|E\|)\|E\| \\ \le\; & \|A\|^2(2\tau_1 + f_1\upsilon + f_2\omega)(f_1\upsilon + f_2\omega) + \sqrt{n}\|A\|^2(2\tau_1 + 2 + 2p(m,n)\upsilon)p(m,n)\upsilon \\ \equiv\; & \zeta_1\|A\|^2\upsilon, \end{aligned}$$

where $\zeta_1 = (2\tau_1 + f_1\upsilon + f_2\omega)(f_1 + f_2\omega\upsilon^{-1}) + 2\sqrt{n}(\tau_1 + 1 + p(m,n)\upsilon)p(m,n)$. □

In the following theorem, we give a sufficient condition for the quadratic convergence of Algorithm 5.2.

Theorem 5.5 *Let $C^{(k)}$ be the matrix C after k Jacobi updates in Algorithm 5.2 with $C^{(0)} = AQ$. Suppose $\omega \ll \upsilon$ and*

$$\min_{\sigma_i(C^{(0)}) \ne \sigma_j(C^{(0)})} |\sigma_i^2(C^{(0)}) - \sigma_j^2(C^{(0)})| \ge d > 0.$$

If $\zeta_1\|A\|^2\upsilon < d/(4\sqrt{2})$ for some constant $\zeta_1 = \zeta_1(m,n)$, then we have

$$\|\mathrm{off}((C^{(N)})^T C^{(N)})\|_F \le \frac{\sqrt{34/9}}{d}\|\mathrm{off}((C^{(0)})^T C^{(0)})\|_F^2, \tag{5.4}$$

where $N = n(n-1)/2$.

Proof. By Theorem 5.4 we have

$$\|\mathrm{off}((C^{(0)})^T C^{(0)}\|_F = \|\mathrm{off}(Q^T A^T A Q)\|_F \le \zeta_1 \|A\|^2 \upsilon,$$

for some constant constant $\zeta_1 \equiv \zeta_1(m,n)$. Therefore, we have $\|\mathrm{off}((C^{(0)})^T C^{(0)}\|_F < d/(4\sqrt{2})$ provided that $\zeta_1\|A\|^2\upsilon < d/(4\sqrt{2})$. By Lemma 3.3 we obtain (5.4). □

In Theorem 5.5, the quadratic convergence of Algorithm 5.2 is established under the assumption that $\zeta_1\|A\|^2\upsilon < d/(4\sqrt{2})$ for some constant $\zeta_1 \equiv \zeta_1(m,n)$ and there exists a clear gap between the distinct singular values of the matrix $C^{(0)} = AQ$. In the following theorem, we establish the quadratic convergence of Algorithm 5.2 under the assumption that there exists a clear gap between the distinct singular values of the original matrix A.

Theorem 5.6 *Let $C^{(k)}$ be the matrix C after k Jacobi updates in Algorithm 5.2 with $C^{(0)} = AQ$. Suppose $\omega \ll \upsilon$ and*

$$\min_{\sigma_i(A))\neq\sigma_j(A)} |\sigma_i^2(A) - \sigma_j^2(A)| \ge d > 0.$$

If $2\zeta_2\|A\|^2\omega \le \rho_2 d$ for some constant $\zeta_2 = \zeta_2(m,n)$ and $0 < \rho_2 < 1$ and $\zeta_1\|A\|^2\upsilon < (1-\rho_1)d/(4\sqrt{2})$ for some constant $\zeta_1 = \zeta_1(m,n)$, then we have

$$\|\mathrm{off}((C^{(N)})^T C^{(N)})\|_F \le \frac{\sqrt{34/9}}{(1-\rho_2)d}\|\mathrm{off}((C^{(0)})^T C^{(0)})\|_F^2. \tag{5.5}$$

Proof. By Lemma 2.1, there exists a matrix $\delta Q \in \mathbb{R}^{n\times n}$ such that $Q + \delta Q$ is orthogonal, where $\|\delta Q\| \le \eta_4\omega$ with $\eta_4 = \eta_4(n)$ being defined by (4.13). Then

$$\|Q\| \le \|Q + \delta Q\| + \|\delta Q\| = 1 + \|\delta Q\| \le 1 + \eta_4\omega \equiv \eta_5. \tag{5.6}$$

We note that

$$(Q+\delta Q)^T A^T A (Q+\delta Q) = Q^T A^T A Q + Q^T A^T A \delta Q + \delta Q^T A^T A Q + \delta Q^T A^T A \delta Q.$$

By hypothesis, $C^{(0)} = AQ$. From Lemma 2.4 and (5.6) we obtain, for any $1 \le k \le n$,

$$\begin{aligned} & |\sigma_k^2(A) - \sigma_k^2(C^{(0)})| = |\lambda_k(A^T A) - \lambda_k(Q^T A^T A Q)| \le 2\|A\|^2\|Q\|\|\delta Q\| + \|A\|^2\|\delta Q\|^2 \\ \le\ & \|A\|^2(2\|Q\| + \|\delta Q\|)\|\delta Q\| \le \|A\|^2(2\eta_5 + \eta_4\omega)\eta_4\omega \equiv \zeta_2\|A\|^2\omega. \end{aligned}$$

Let $\sigma_i(A)$ and $\sigma_j(A)$ be arbitrary two distinct singular values of A. Then we have

$$\begin{aligned} & |\sigma_i^2(A) - \sigma_j^2(A)| = |\lambda_i(A^T A) - \lambda_j(A^T A)| \\ \le\ & |\lambda_i(A^T A) - \lambda_i(Q^T A^T A Q)| + |\lambda_i(Q^T A^T A Q) - \lambda_j(Q^T A^T A Q)| + |\lambda_j(Q^T A^T A Q) - \lambda_j(A^T A)| \\ \le\ & |\lambda_i(Q^T A^T A Q) - \lambda_j(Q^T A^T A Q)| + 2\zeta_2\|A\|^2\omega = |\sigma_i^2(C^{(0)}) - \sigma_j^2(C^{(0)})| + 2\zeta_2\|A\|^2\omega \end{aligned}$$

and

$$\begin{aligned} & |\sigma_i^2(A) - \sigma_j^2(A)| = |\lambda_i(A^T A) - \lambda_j(A^T A)| \\ \ge\ & ||\lambda_i(Q^T A^T A Q) - \lambda_j(Q^T A^T A Q)| - |\lambda_i(A^T A) - \lambda_i(Q^T A^T A Q)| - |\lambda_j(Q^T A^T A Q) - \lambda_j(A^T A)| \\ \ge\ & |\lambda_i(Q^T A^T A Q) - \lambda_j(Q^T A^T A Q)| - 2\zeta_2\|A\|^2\omega = |\sigma_i^2(C^{(0)}) - \sigma_j^2(C^{(0)})| - 2\zeta_2\|A\|^2\omega. \end{aligned}$$

If $2\zeta_2\|A\|^2\omega \leq \rho_2 d$ for some $0 < \rho_2 < 1$, then we have

$$\frac{1}{1+\rho_2}|\sigma_i^2(C^{(0)}) - \sigma_j^2(C^{(0)})| \leq |\sigma_i^2(A) - \sigma_j^2(A)| \leq \frac{1}{1-\rho_2}|\sigma_i^2(C^{(0)}) - \sigma_j^2(C^{(0)})|.$$

By hypothesis, $\min_{\sigma_i(A))\neq\sigma_j(A)} |\sigma_i^2(A) - \sigma_j^2(A)| \geq d > 0$. Hence,

$$\min_{\sigma_i(C^{(0)})\neq\sigma_j(C^{(0)})} |\sigma_i^2(C^{(0)}) - \sigma_j^2(C^{(0)})| \geq (1-\rho_2) \min_{\sigma_i(A))\neq\sigma_j(A)} |\sigma_i^2(A) - \sigma_j^2(A)| \geq (1-\rho_2)d > 0.$$

By Theorem 5.5 we know that (5.5) holds if $\zeta_1\|A\|^2\upsilon < (1-\rho_2)d/(4\sqrt{2})$. □

6 Numerical Experiments

In this section, we present some numerical experiments to illustrate the effectiveness of Algorithms 3.2 and 4.1 for computing the symmetric eigenvalue decomposition and Algorithms 5.1 and 5.2 for computing the SVD. All numerical experiments were implemented in `MATLAB` 2021a running on a workstation of an Intel Xeon Gold CPU 6134 at 3.2 GHz, 32GB of RAM and NVIDIA GeForce GTX 1080 Ti. In our numerical tests, we set $\upsilon = 2^{-24}$ (single precision) and $\omega = 2^{-53}$ (double precision) for Algorithms 4.1 and 5.2.

In Algorithm 4.1, we use the built-in function `eig` in `MATLAB R2021a` as the eigensolver in precision υ, or employ LAPACK routine 'SSYEV' to compute the eigenvalues and associated eigenvectors of a given matrix in precision υ. In Algorithm 5.2, we use the built-in function `svd` in `MATLAB R2021a` as the SVD solver in precision υ, or employ LAPACK routine 'SGESVD' to compute the singular values and the associated left and right singular vectors of a given matrix in precision υ.

In our numerical tests, "`Res.`", "`OR-P.`", "`CT.`", "`JU.`", and "`SP.`" denote the computed relative residual $\|AP - PT\|_F/\|A\|_F$ (or $\|AV - U\Sigma\|_F/\|A\|_F$), the measure of orthonormality $\|P^TP - I_n\|_F$ of P, the running time in seconds, the Jacobi updates, and the number of sweeps at the final iterates of the corresponding algorithms, respectively. In addition, for Algorithms 3.2 and 4.1, we set $\epsilon = 0.1 \times \omega$ and $\nu = 20.0 \times \omega$. For Algorithms 5.1 and 5.2, we set $\epsilon = 0.1 \times \omega$ and $\nu = \omega$ and the algorithms are stopped if the ratio of Jacobi updates to $N := n(n-1)/2$ is less than 2×10^{-4}.

6.1 The spectral decomposition

Example 6.1 *Let A be an $n \times n$ random symmetric and positive definite matrix with pre-assigned singular value generated by* `MATLAB 2021a`*'s* `gallery ('randsvd', n, -kappa, mode)` *with $\kappa(A) =$* `kappa`*. We report our numerical results for* (a) `mode` $= 1$*: the large eigenvalue is* 1 *and the rest of the eigenvalues are* 1/`kappa`*,* (b) `mode` $= 2$*: the small eigenvalue is* 1/`kappa` *and the rest of the eigenvalues are* 1*,* (c) `mode` $= 3$*: geometrically distributed eigenvalues,* (d) `mode` $= 4$*: arithmetically distributed eigenvalues, and* (e) `mode` $= 5$*: random eigenvalues with uniformly distributed logarithm.*

In Table 6.1, we report the numerical results for Example 6.1. We observe from Table 6.1 that Algorithm 4.1 preserves the high accuracy of Algorithm 3.2 for all test matrices and even produces a sightly better orthonormality of P than Algorithm 3.2 (especially for the cases that $\texttt{mode} = 3, 4, 5$). Moreover, Algorithm 4.1 works much better than Algorithm 3.2 for the cases that $\texttt{mode} = 3, 4, 5$ in terms of the computing time, the number of rotations, and sweeps.

Figure 6.1 depicts the quantities $d(A)/(4\sqrt{2})$ and $\|\mathrm{off}(Q^TAQ)\|_F$ generated by Algorithm 4.1 versus the dimension n for Example 6.1 with $\texttt{kappa} = 10^8$, Here, $d(A) = \min_{\lambda_i(A))\neq\lambda_j(A)} |\lambda_i(A) - \lambda_j(A)|$ and $\texttt{bd} = n\|A\|\upsilon$ is an approximation estimate of the theoretical bound $\gamma_1\|A\|\upsilon$, which is obtained in Theorem 4.2. We see from Figure 6.1 that $\|\mathrm{off}(Q^TAQ)\|_F$ can be controlled by the theoretical bound $\gamma_1\|A\|\upsilon$. This confirms our theoretical result. However, the quantity $\gamma_1\|A\|\upsilon$ is not necessarily less than $d(A)/(4\sqrt{2})$, which depends on the eigenvalue distribution of the original matrix A. As noted in Theorem 4.4, when $\gamma_1\|A\|\upsilon < d(A)/(4\sqrt{2})$, the quadratic convergence of Algorithm 4.1 can be obtained (see $\texttt{mode} = 1, 2$).

To further illustrate the effectiveness of the preconditioner Q generated by Algorithm 4.1, in Figure 6.2, we plot the quantities $\|\mathrm{off}(A)\|_F$, $\|\mathrm{off}(Q^TAQ)\|_F$, $\texttt{bd}$, and $d(A)/(4\sqrt{2})$ versus the dimension n for Example 6.1 with $\texttt{mode} = 4$ and $\texttt{mode} = 5$, respectively. It can be seen that the preprocessed matrix Q is such that the Frobenius norm of the off-diagnal elements of Q^TAQ is much less than that of the original matrix A. This shows that the preconditioner Q is very effective.

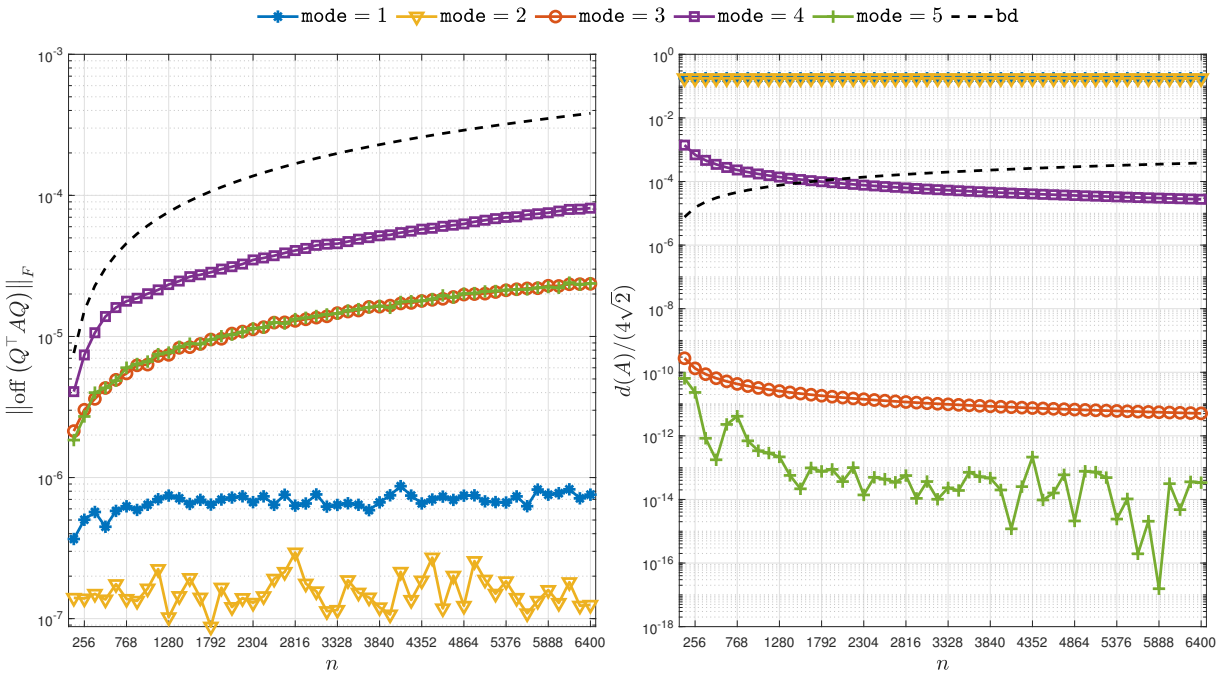

Figure 6.1: $\|\mathrm{off}(Q^TAQ)\|_F$ and $d(A)/(4\sqrt{2})$ versus n for Example 6.1.

To further illustrate the effectiveness of Algorithm 4.1, we also implement Algorithm 3.2 and

Table 6.1: Numerical results for Example 6.1 with $n = 512$.

	eig			Alg. 3.2					Alg. 4.1				
(mode, kappa)	Res.	OR-P.	CT.	Res.	OR-P.	JU.	CT.	SP.	Res.	OR-P.	JU.	CT.	SP.
$(1, 10^3)$	8.79e-16	3.86e-15	0.02	1.21e-15	4.23e-15	0.97N	0.63	1	1.35e-15	4.05e-15	0.98N	0.65	1
$(2, 10^3)$	3.29e-15	3.31e-15	0.02	7.64e-15	5.62e-15	0.73N	0.45	1	3.76e-15	5.61e-15	0.82N	0.54	1
$(3, 10^3)$	1.14e-15	5.46e-15	0.02	3.46e-15	9.33e-15	9.63N	5.70	12	1.98e-15	4.58e-15	2.00N	1.22	2
$(4, 10^3)$	3.13e-15	6.00e-15	0.02	4.14e-15	9.00e-15	8.83N	5.21	10	1.98e-15	4.61e-15	2.00N	1.22	2
$(5, 10^3)$	1.25e-15	5.87e-15	0.02	3.39e-15	9.29e-15	9.66N	5.68	12	1.83e-15	4.67e-15	2.02N	1.23	3
$(1, 10^4)$	6.23e-16	3.40e-15	0.01	1.18e-15	3.69e-15	1.00N	0.60	1	1.63e-15	3.68e-15	1.00N	0.64	1
$(2, 10^4)$	3.20e-15	3.17e-15	0.02	3.45e-15	5.68e-15	0.72N	0.44	1	3.67e-15	5.62e-15	0.82N	0.53	1
$(3, 10^4)$	1.13e-15	5.41e-15	0.01	3.46e-15	9.34e-15	9.89N	5.75	13	1.80e-15	4.62e-15	2.07N	1.27	3
$(4, 10^4)$	2.78e-15	5.85e-15	0.02	4.04e-15	8.91e-15	8.80N	5.16	10	2.00e-15	4.57e-15	2.00N	1.22	2
$(5, 10^4)$	1.37e-15	5.87e-15	0.02	3.49e-15	9.56e-15	10.04N	5.97	13	1.89e-15	4.71e-15	2.10N	1.30	3
$(1, 10^5)$	6.18e-16	4.04e-15	0.01	1.43e-15	3.64e-15	1.00N	0.59	1	1.63e-15	3.63e-15	1.00N	0.63	1
$(2, 10^5)$	3.22e-15	3.43e-15	0.01	4.54e-15	5.75e-15	0.73N	0.45	1	3.75e-15	5.48e-15	0.82N	0.54	1
$(3, 10^5)$	1.38e-15	5.71e-15	0.01	3.43e-15	9.62e-15	10.25N	6.04	14	1.94e-15	4.60e-15	2.18N	1.36	4
$(4, 10^5)$	2.82e-15	5.86e-15	0.02	4.06e-15	9.08e-15	8.86N	5.24	10	1.99e-15	4.57e-15	2.00N	1.23	2
$(5, 10^5)$	1.24e-15	5.88e-15	0.02	3.56e-15	9.69e-15	10.39N	6.18	15	1.90e-15	4.69e-15	2.23N	1.39	4
$(1, 10^6)$	7.51e-16	3.51e-15	0.01	1.07e-15	3.91e-15	1.00N	0.60	1	1.44e-15	3.78e-15	1.00N	0.64	1
$(2, 10^6)$	3.32e-15	3.41e-15	0.01	1.14e-14	5.67e-15	0.74N	0.45	1	3.88e-15	5.35e-15	0.83N	0.54	1
$(3, 10^6)$	1.05e-15	5.13e-15	0.02	4.16e-15	9.74e-15	10.81N	6.41	15	3.60e-15	4.98e-15	2.53N	1.59	6
$(4, 10^6)$	2.92e-15	6.37e-15	0.02	4.15e-15	9.16e-15	8.90N	5.28	10	1.91e-15	4.53e-15	2.00N	1.22	2
$(5, 10^6)$	1.19e-15	6.69e-15	0.02	3.74e-15	9.74e-15	10.88N	6.41	16	2.02e-15	4.93e-15	2.50N	1.57	6

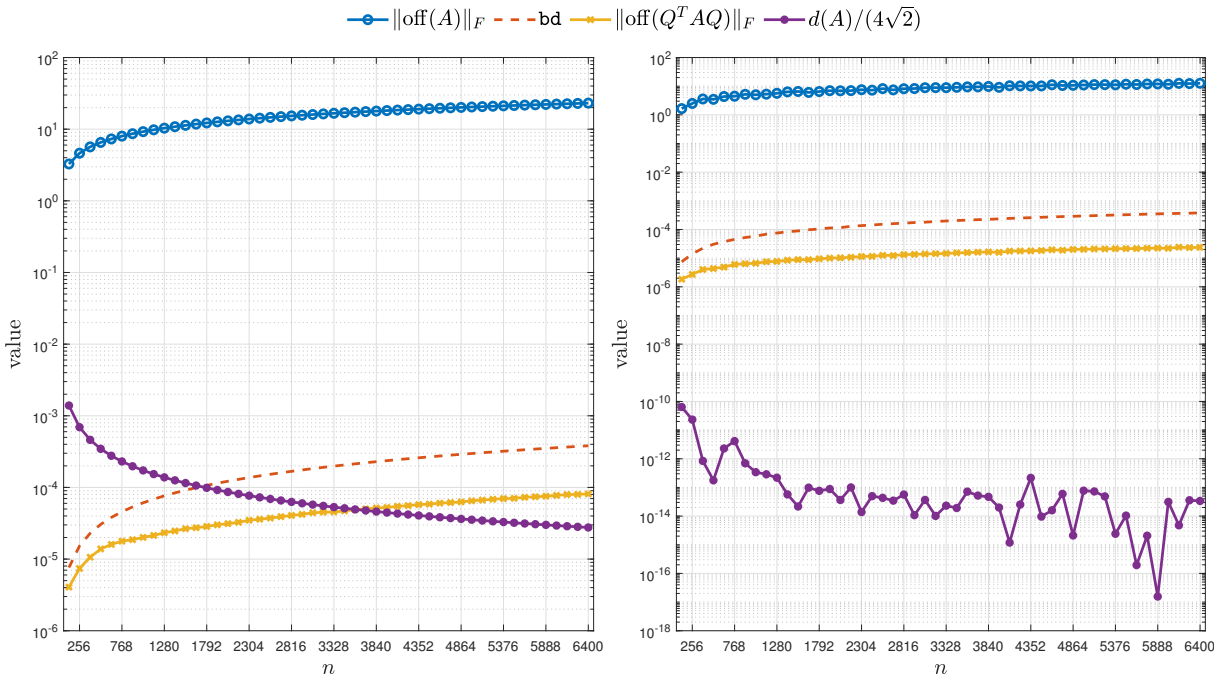

Figure 6.2: Different quantities versus n for Example 6.1 (mode-4 (left) and mode-5 (right)).

Algorithm 4.1 in parallel. Here, we utilize NVIDIA cuSOLVER library [1] and NVIDIA cuBLAS library[2] and the parallel ordering of the rotation set $\{(i,j) \mid 1 \le i < j \le n\}$ can be taken as some non-overlapping order, e.g., the merry-go-round ordering $(i,j) = (1,2),(3,4),\ldots,(n-1,n),(1,4),\ldots,(n-3,n-1),(1,6),\ldots,(n,n-2)$. For a fair comparison, all algorithms were rewritten as mex-files and compiled by the corresponding MATLAB command 'mex' or 'mexcuda'. The pre-processing stage in Steps 1–3 of Algorithm 4.1 was implemented on the NVIDIA CUDA routine, where the function '`cusolverDnSsyevd`' was employed as the eigensolver in precision v and the MGS method was replaced by the Householder QR factorization, which can theoretically guarantee higher orthogonality [22, §5.2].

The numerical results for Example 6.1 with different n are displayed in Tables 6.2–6.5. Here, "`CT-Pre.`" means the running time for the pre-processing stage in Steps 1–3 of (parallel) Algorithm 4.1 and "`CT-J.`" means the running time for the stage of the Jacobi procedure of (parallel) Algorithm 4.1 or (parallel) Algorithm 3.2. We see from Tables 6.2–6.5 that, as parallel Algorithm 3.2, parallel Algorithm 4.1 can significantly improve the efficiency of Algorithm 4.1. As expected, Algorithm 4.1 (parallel version, respectively) works much better than Algorithm 3.2 (parallel version, respectively) in terms of the total computing time.

In the following numerical example, we consider the case of multiple eigenvalues.

Example 6.2 *We consider the case of multiple eigenvalues. Let* $A = P_*\mathrm{diag}(\mathtt{lam}\otimes\mathbf{1}_4)P_*^T$ *be an* $n \times n$ *random symmetric and positive definite matrix with* $n = 4s$*, where the orthogonal matrix*

[1] https://docs.nvidia.com/cuda/cusolver
[2] https://docs.nvidia.com/cuda/cublas

Table 6.2: Numerical results for Example 6.1 with $n = 1024$.

	parallel Alg. 3.2				parallel Alg. 4.1				
(mode, kappa)	Res.	OR-P.	CT-J.	SP.	Res.	OR-P.	CT-Pre.	CT-J.	SP.
$(3, 10^3)$	8.08e-13	9.37e-13	1.19	14	1.01e-13	1.34e-13	0.53	0.15	2
$(4, 10^3)$	6.65e-13	6.78e-13	1.12	11	1.10e-13	1.22e-13	0.36	0.14	2
$(5, 10^3)$	8.09e-13	9.60e-13	1.33	14	1.03e-13	1.31e-13	0.41	0.15	2
$(3, 10^4)$	8.74e-13	1.02e-12	1.31	16	1.01e-13	1.53e-13	0.52	0.15	2
$(4, 10^4)$	6.62e-13	6.95e-13	1.00	11	1.15e-13	1.23e-13	0.37	0.14	2
$(5, 10^4)$	9.51e-13	1.16e-12	1.33	16	1.02e-13	1.53e-13	0.39	0.15	3
$(3, 10^5)$	9.37e-13	1.34e-12	1.34	18	1.00e-13	2.01e-13	0.51	0.16	3
$(4, 10^5)$	6.75e-13	7.14e-13	1.13	11	1.14e-13	1.25e-13	0.39	0.15	2
$(5, 10^5)$	9.89e-13	1.43e-12	1.23	18	9.55e-14	2.05e-13	0.38	0.16	3
$(3, 10^6)$	9.97e-13	1.42e-12	1.61	19	9.84e-14	2.46e-13	0.56	0.17	3
$(4, 10^6)$	6.89e-13	6.94e-13	1.00	12	1.16e-13	1.23e-13	0.38	0.15	2
$(5, 10^6)$	1.03e-12	1.47e-12	1.52	19	9.41e-14	2.54e-13	0.38	0.20	4
	Alg. 3.2				Alg. 4.1				
(mode, kappa)	Res.	OR-P.	CT-J.	SP.	Res.	OR-P.	CT-Pre.	CT-J.	SP.
$(3, 10^3)$	5.04e-15	1.36e-14	34.28	13	2.76e-15	6.33e-15	0.52	6.64	2
$(4, 10^3)$	6.40e-15	1.32e-14	30.90	11	2.83e-15	6.45e-15	0.37	6.63	2
$(5, 10^3)$	5.09e-15	1.37e-14	34.34	14	2.66e-15	6.40e-15	0.42	6.64	2
$(3, 10^4)$	5.07e-15	1.38e-14	34.63	14	2.57e-15	6.53e-15	0.53	6.99	3
$(4, 10^4)$	6.43e-15	1.30e-14	30.78	11	2.80e-15	6.42e-15	0.37	6.60	2
$(5, 10^4)$	5.19e-15	1.39e-14	35.67	15	2.68e-15	6.50e-15	0.45	7.21	3
$(3, 10^5)$	5.15e-15	1.41e-14	37.72	15	2.67e-15	6.58e-15	0.51	7.61	3
$(4, 10^5)$	6.18e-15	1.30e-14	32.33	11	2.83e-15	6.41e-15	0.41	6.93	2
$(5, 10^5)$	5.24e-15	1.40e-14	38.17	16	2.68e-15	6.62e-15	0.39	7.74	3
$(3, 10^6)$	5.14e-15	1.43e-14	38.66	17	3.53e-15	6.71e-15	0.55	8.27	4
$(4, 10^6)$	6.33e-15	1.31e-14	32.06	11	2.82e-15	6.41e-15	0.41	6.93	2
$(5, 10^6)$	5.32e-15	1.44e-14	39.15	17	2.72e-15	6.65e-15	0.42	8.35	4

$P_* \in \mathbb{R}^{n\times n}$ is randomly generated by the built-in functions `randn` *and* `orth` *in* `MATLAB R2021a` *and the vector* `lam` $\in \mathbb{R}^s$ *of exact eigenvalues is generated as follows:*

(a) `mode` $= 1$*:* `lam=[1; ones(s-1,1)*1/kappa];`

(b) `mode` $= 2$*:* `lam=[ones(s-1,1); 1/kappa];`

(c) `mode` $= 3$*:* `lam=kappa.*linspace(-1,0,s);`

(d) `mode` $= 4$*:* `lam=1-(1-1/kappa).*linspace(0,1,s);`

(e) `mode` $= 5$*:* `lam=kappa.*(-rand(s,1)).`

We apply Algorithm 4.1 and its parallel version to Example 6.2. The numerical results for Example 6.2 with different n are reported in Tables 6.6–6.9, where the values of $\|\mathrm{off}(Q^TAQ)\|_F$ and $d(A)/(4\sqrt{2})$ were calculated under CPU environment. We observe from Tables 6.6–6.9 that

Table 6.3: Numerical results for Example 6.1 with $n = 2048$.

	parallel Alg. 3.2				parallel Alg. 4.1				
(mode, kappa)	Res.	OR-P.	CT-J.	SP.	Res.	OR-P.	CT-Pre.	CT-J.	SP.
$(3, 10^3)$	1.80e-12	1.99e-12	5.91	15	2.27e-13	3.55e-13	0.51	0.79	2
$(4, 10^3)$	1.52e-12	1.60e-12	4.57	12	2.34e-13	2.46e-13	0.50	0.78	2
$(5, 10^3)$	1.69e-12	2.05e-12	5.81	15	2.38e-13	3.91e-13	0.50	0.81	3
$(3, 10^4)$	1.93e-12	2.51e-12	6.18	16	2.37e-13	4.88e-13	0.51	0.86	3
$(4, 10^4)$	1.59e-12	1.62e-12	4.70	12	2.36e-13	2.46e-13	0.49	0.78	2
$(5, 10^4)$	1.91e-12	2.38e-12	6.51	17	2.34e-13	4.96e-13	0.50	0.89	3
$(3, 10^5)$	1.97e-12	2.95e-12	6.95	18	2.44e-13	6.60e-13	0.50	1.00	4
$(4, 10^5)$	1.58e-12	1.65e-12	4.64	12	2.38e-13	2.47e-13	0.50	0.78	2
$(5, 10^5)$	2.07e-12	2.89e-12	7.24	19	2.52e-13	7.05e-13	0.51	1.03	4
$(3, 10^6)$	2.19e-12	3.14e-12	7.51	20	2.58e-13	8.61e-13	0.53	1.25	5
$(4, 10^6)$	1.54e-12	1.59e-12	4.65	12	2.39e-13	2.46e-13	0.52	0.78	2
$(5, 10^6)$	2.14e-12	2.79e-12	7.86	21	2.66e-13	9.45e-13	0.51	1.38	6
	Alg. 3.2				Alg. 4.1				
(mode, kappa)	Res.	OR-P.	CT-J.	SP.	Res.	OR-P.	CT-Pre.	CT-J.	SP.
$(3, 10^3)$	7.31e-15	1.95e-14	395.99	14	3.58e-15	9.01e-15	3.80	67.77	3
$(4, 10^3)$	9.40e-15	1.91e-14	321.23	11	3.71e-15	8.99e-15	2.86	65.00	2
$(5, 10^3)$	7.48e-15	1.97e-14	348.94	14	3.64e-15	9.01e-15	3.14	66.43	3
$(3, 10^4)$	7.42e-15	1.98e-14	353.86	15	3.66e-15	9.15e-15	3.52	69.58	3
$(4, 10^4)$	9.70e-15	1.87e-14	319.15	11	3.71e-15	8.97e-15	2.88	65.02	2
$(5, 10^4)$	7.60e-15	1.98e-14	358.31	15	3.71e-15	9.13e-15	3.02	69.32	4
$(3, 10^5)$	7.37e-15	2.00e-14	360.13	16	3.57e-15	9.22e-15	3.47	73.13	3
$(4, 10^5)$	9.54e-15	1.88e-14	316.63	11	3.71e-15	9.04e-15	2.87	64.64	2
$(5, 10^5)$	1.23e-14	2.02e-14	365.95	16	3.69e-15	9.29e-15	2.87	72.70	3
$(3, 10^6)$	7.55e-15	2.03e-14	372.45	17	3.63e-15	9.43e-15	3.50	77.54	4
$(4, 10^6)$	9.27e-15	1.89e-14	317.12	11	3.71e-15	8.97e-15	2.88	64.57	2
$(5, 10^6)$	7.71e-15	2.04e-14	376.45	18	3.60e-15	9.31e-15	2.82	76.62	4

parallel Algorithm 4.1 is much more efficient than Algorithm 4.1 in terms of the total computing time.

We point out that, from our other numerical tests, we see that the stopping criterion $|a_{ij}| \leq \nu \min\{|a_{ii}|, |a_{jj}|\}$ in Algorithm 3.2 ($|t_{ij}| \leq \nu \min\{|t_{ii}|, |t_{jj}|\}$ in Algorithm 4.1, respectively) is as effective as the stopping criterion $|a_{ij}| \leq \nu\sqrt{a_{ii}a_{jj}}$ ($|t_{ij}| \leq \nu\sqrt{t_{ii}t_{jj}}$, respectively) if A is positive definite (as noted in Remark 3.2).

6.2 The singular value decomposition

In this section, we compare Algorithm 5.1 and Algorithm 5.2 for computing the SVD of a real matrix. We consider the following example.

Example 6.3 *Let A be an $m \times n$ random matrix with pre-assigned singular value generated by* `MATLAB 2021a`*'s* `gallery ('randsvd', [m,n], kappa, mode)` *with $\kappa(A) =$* `kappa`*. We report our numerical results for* (a) `mode` $= 1$*: the large singular value is equal to* 1 *and the rest of the*

Table 6.4: Numerical results for Example 6.1 with $n = 3072$.

	parallel Alg. 3.2				parallel Alg. 4.1				
(mode, kappa)	Res.	OR-P.	CT-J.	SP.	Res.	OR-P.	CT-Pre.	CT-J.	SP.
$(3, 10^3)$	2.74e-12	3.51e-12	15.84	15	2.92e-13	4.45e-13	14.70	2.59	2
$(4, 10^3)$	2.64e-12	2.69e-12	12.85	12	3.24e-13	3.54e-13	13.36	2.38	2
$(5, 10^3)$	3.08e-12	3.25e-12	16.60	16	2.96e-13	4.83e-13	13.93	2.59	3
$(3, 10^4)$	2.96e-12	3.89e-12	17.47	17	2.99e-13	5.27e-13	14.04	2.66	3
$(4, 10^4)$	2.69e-12	2.73e-12	12.42	12	3.39e-13	3.54e-13	11.75	2.59	2
$(5, 10^4)$	3.13e-12	3.91e-12	17.88	17	2.97e-13	6.03e-13	13.24	2.70	3
$(3, 10^5)$	3.17e-12	4.42e-12	19.66	19	2.98e-13	7.28e-13	14.66	2.99	4
$(4, 10^5)$	2.59e-12	2.67e-12	12.39	13	3.40e-13	3.54e-13	12.68	2.63	2
$(5, 10^5)$	2.86e-12	4.23e-12	20.03	19	2.79e-13	8.30e-13	13.83	3.13	5
$(3, 10^6)$	3.18e-12	4.87e-12	21.01	21	2.83e-13	9.20e-13	14.76	3.12	4
$(4, 10^6)$	2.70e-12	2.73e-12	12.47	12	3.37e-13	3.56e-13	11.93	2.52	2
$(5, 10^6)$	3.01e-12	4.61e-12	21.29	21	2.74e-13	8.31e-13	12.55	2.92	3
	Alg. 3.2				Alg. 4.1				
(mode, kappa)	Res.	OR-P.	CT-J.	SP.	Res.	OR-P.	CT-Pre.	CT-J.	SP.
$(3, 10^3)$	9.14e-15	2.43e-14	1474.36	14	4.41e-15	1.11e-14	16.01	284.97	3
$(4, 10^3)$	1.16e-14	2.37e-14	1439.29	12	4.87e-15	1.10e-14	13.61	282.76	2
$(5, 10^3)$	9.14e-15	2.46e-14	1526.52	15	4.37e-15	1.11e-14	13.21	273.05	3
$(3, 10^4)$	9.12e-15	2.45e-14	1503.95	16	4.35e-15	1.12e-14	13.55	278.95	3
$(4, 10^4)$	1.13e-14	2.35e-14	1329.41	12	4.47e-15	1.10e-14	11.95	263.22	2
$(5, 10^4)$	9.23e-15	2.47e-14	1521.26	16	4.45e-15	1.12e-14	13.24	291.74	4
$(3, 10^5)$	9.22e-15	2.48e-14	1517.20	17	4.41e-15	1.14e-14	14.96	322.14	5
$(4, 10^5)$	1.19e-14	2.35e-14	1392.39	12	4.49e-15	1.10e-14	12.32	272.59	2
$(5, 10^5)$	9.37e-15	2.49e-14	1519.39	18	4.33e-15	1.13e-14	12.77	303.30	6
$(3, 10^6)$	9.44e-15	2.53e-14	1535.48	18	4.41e-15	1.16e-14	14.32	324.59	5
$(4, 10^6)$	1.16e-14	2.36e-14	1316.85	12	4.49e-15	1.10e-14	11.99	261.22	2
$(5, 10^6)$	9.50e-15	2.54e-14	1559.92	19	4.39e-15	1.17e-14	12.59	323.06	5

singular values are equal to 1/kappa, (b) mode $= 2$*: the small singular value is equal to* 1/kappa *and the rest of the singular values are equal to* 1, (c) mode $= 3$*: geometrically distributed singular values,* (d) mode $= 4$*: arithmetically distributed singular values, and* (e) mode $= 5$*: random singular values with uniformly distributed logarithm.*

The numerical results for Example 6.3 are reported in Table 6.10. We see from Table 6.10 that Algorithm 5.2 works more efficient than Algorithm 5.1 for the cases mode $= 3, 4, 5$.

7 Conclusions

We have proposed a mixed precision Jacobi method for computing the eigenvalue decomposition of a real symmetric matrix. We first use an eigensolver to compute the eigenvalue decomposition of the original real symmetric matrix at low precision and then a low-precision matrix of eigenvectors is obtained. Then by using the modified Gram-Schmidt method to the low-precision eigenvector matrix, a high-precision orthogonal matrix is computed, which is employed as an

Table 6.5: Numerical results for Example 6.1 with $n = 4096$.

	parallel Alg. 3.2				parallel Alg. 4.1				
(mode, kappa)	Res.	OR-P.	CT-J.	SP.	Res.	OR-P.	CT-Pre.	CT-J.	SP.
$(3, 10^3)$	4.81e-12	6.26e-12	38.98	7	4.64e-13	7.94e-13	1.02	5.81	1
$(4, 10^3)$	4.96e-12	5.08e-12	31.51	6	4.76e-13	5.03e-13	1.01	5.71	1
$(5, 10^3)$	6.09e-12	6.83e-12	39.62	7	4.65e-13	8.67e-13	1.01	6.25	2
$(3, 10^4)$	7.19e-12	7.52e-12	40.86	7	4.79e-13	1.12e-12	1.03	6.29	2
$(4, 10^4)$	5.13e-12	5.27e-12	32.24	6	4.80e-13	5.05e-13	1.06	5.68	1
$(5, 10^4)$	6.56e-12	7.84e-12	44.39	8	5.13e-13	1.10e-12	1.03	7.41	2
$(3, 10^5)$	6.44e-12	8.26e-12	44.42	8	5.26e-13	1.44e-12	1.07	6.77	2
$(4, 10^5)$	4.87e-12	5.05e-12	31.50	6	4.86e-13	5.04e-13	1.18	5.73	1
$(5, 10^5)$	7.58e-12	8.04e-12	47.17	8	5.12e-13	1.43e-12	1.06	7.56	2
$(3, 10^6)$	7.89e-12	9.31e-12	50.17	9	5.38e-13	1.71e-12	1.06	8.12	2
$(4, 10^6)$	4.97e-12	5.10e-12	31.19	6	4.80e-13	5.00e-13	1.08	5.70	1
$(5, 10^6)$	7.73e-12	9.58e-12	50.30	9	5.64e-13	2.10e-12	1.07	8.81	2
	Alg. 3.2				Alg. 4.1				
(mode, kappa)	Res.	OR-P.	CT-J.	SP.	Res.	OR-P.	CT-Pre.	CT-J.	SP.
$(3, 10^3)$	1.07e-14	2.83e-14	3995.41	15	5.04e-15	1.28e-14	41.73	705.95	3
$(4, 10^3)$	1.43e-14	2.75e-14	3598.65	12	5.08e-15	1.27e-14	36.85	683.58	2
$(5, 10^3)$	1.08e-14	2.85e-14	3974.42	15	5.06e-15	1.28e-14	38.70	744.56	3
$(3, 10^4)$	1.07e-14	2.86e-14	4112.14	16	5.06e-15	1.29e-14	45.39	790.42	4
$(4, 10^4)$	1.46e-14	2.74e-14	3620.17	12	5.13e-15	1.26e-14	42.78	731.64	2
$(5, 10^4)$	1.07e-14	2.87e-14	4065.31	17	5.01e-15	1.29e-14	37.88	753.82	4
$(3, 10^5)$	1.08e-14	2.89e-14	3999.31	17	5.06e-15	1.31e-14	47.01	839.97	4
$(4, 10^5)$	1.38e-14	2.74e-14	3648.51	12	5.10e-15	1.27e-14	37.36	687.37	3
$(5, 10^5)$	1.52e-14	2.91e-14	4239.42	17	4.98e-15	1.31e-14	38.69	806.17	4
$(3, 10^6)$	1.25e-14	2.94e-14	4194.82	18	5.02e-15	1.36e-14	45.05	941.32	5
$(4, 10^6)$	1.38e-14	2.74e-14	3639.05	12	5.08e-15	1.27e-14	39.10	707.96	2
$(5, 10^6)$	1.11e-14	2.96e-14	2244.13	19	8.57e-15	1.36e-14	39.88	942.16	6

initial guess of the Jacobi method. The rounding error analysis and quadratic convergence of the proposed method are discussed. We also present a mixed precision one-side Jacobi method for computing the singular values and singular vectors of a real matrix. The numerical results show the efficiency of the proposed mixed precision Jacobi method over the classical Jacobi method. Moreover, the parallel mixed precision Jacobi method can achieve higher speedup on GPUs. An interesting question is how to develop a mixed precision method for the generalized eigenvalue problem. This needs further study.

References

[1] *IEEE Standard for Floating-Point Arithmetic*, IEEE Std 754-2008 (Revision of IEEE 754-1985), Institute of Electrical and Electronics Engineers, 2008.

[2] *Multiprecision Computing Toolbox for MATLAB*, Advanpix, Tokyo, http://www.advanpix.com.

Table 6.6: Numerical results for Example 6.2 with multiple eigenvalues and $n = 1024$.

			parallel Alg. 4.1					Alg. 4.1				
(mode, kappa)	$\|\mathrm{off}(Q^TAQ)\|_F$	$d(A)/(4\sqrt{2})$	Res.	OR-P.	CT-Pre.	CT-J.	SP.	Res.	OR-P.	CT-Pre.	CT-J.	SP.
$(1, 10^3)$	1.78e-06	1.77e-01	1.02e-14	1.89e-14	0.36	0.30	4	2.18e-15	5.02e-15	0.36	3.43	1
$(2, 10^3)$	4.77e-07	1.77e-01	2.22e-14	2.03e-14	0.33	0.31	9	8.69e-15	9.56e-15	0.31	6.24	2
$(3, 10^3)$	1.42e-05	4.85e-06	1.11e-13	1.91e-13	0.46	0.48	6	3.22e-15	6.35e-15	0.47	6.87	2
$(4, 10^3)$	3.74e-05	6.93e-04	1.42e-13	1.72e-13	0.43	0.36	6	3.03e-15	6.38e-15	0.44	6.83	2
$(5, 10^3)$	1.08e-05	1.26e-09	1.23e-13	2.17e-13	0.45	0.33	6	2.94e-15	6.47e-15	0.46	6.92	5
$(1, 10^4)$	2.39e-06	1.77e-01	7.40e-15	3.86e-14	0.39	0.11	4	2.21e-15	5.00e-15	0.39	3.44	1
$(2, 10^4)$	6.65e-07	1.77e-01	2.23e-14	2.19e-14	0.29	0.26	7	9.17e-15	9.96e-15	0.29	6.15	2
$(3, 10^4)$	1.16e-05	6.50e-07	1.12e-13	2.14e-13	0.47	0.30	5	2.66e-15	6.55e-15	0.48	7.31	3
$(4, 10^4)$	3.69e-05	6.93e-04	1.55e-13	1.77e-13	0.43	0.56	6	3.15e-15	6.40e-15	0.46	7.04	2
$(5, 10^4)$	8.33e-06	3.70e-08	1.04e-13	2.23e-13	0.43	0.35	6	3.22e-15	6.52e-15	0.43	7.28	3
$(1, 10^5)$	2.60e-06	1.77e-01	1.20e-14	4.49e-14	0.40	0.11	4	2.20e-15	4.97e-15	0.40	3.50	1
$(2, 10^5)$	4.27e-07	1.77e-01	2.20e-14	2.37e-14	0.31	0.26	7	9.05e-15	9.78e-15	0.31	6.25	2
$(3, 10^5)$	9.92e-06	8.16e-08	1.05e-13	2.52e-13	0.46	0.33	6	2.87e-15	6.59e-15	0.46	7.55	3
$(4, 10^5)$	3.72e-05	6.93e-04	1.53e-13	1.77e-13	0.44	0.36	6	3.36e-15	6.42e-15	0.44	7.03	2
$(5, 10^5)$	9.79e-06	4.65e-09	1.10e-13	2.83e-13	0.39	0.64	6	2.89e-15	6.63e-15	0.41	7.85	3
$(1, 10^6)$	2.43e-06	1.77e-01	1.20e-14	3.73e-14	0.39	0.12	4	2.17e-15	4.94e-15	0.40	3.38	1
$(2, 10^6)$	7.08e-07	1.77e-01	2.45e-14	2.19e-14	0.30	0.27	7	9.40e-15	9.54e-15	0.31	6.37	2
$(3, 10^6)$	8.52e-06	9.84e-09	1.08e-13	3.10e-13	0.42	0.37	7	2.92e-15	6.67e-15	0.45	8.10	3
$(4, 10^6)$	3.84e-05	6.93e-04	1.49e-13	1.80e-13	0.42	0.37	6	3.18e-15	6.41e-15	0.43	6.86	2
$(5, 10^6)$	8.81e-06	5.65e-11	1.04e-13	3.21e-13	0.35	0.37	7	3.05e-15	6.65e-15	0.38	8.17	5

Table 6.7: Numerical results for Example 6.2 with $n = 2048$.

			parallel Alg. 4.1					Alg. 4.1				
(mode, kappa)	$\|\mathrm{off}(Q^TAQ)\|_F$	$d(A)/(4\sqrt{2})$	Res.	OR-P.	CT-Pre.	CT-J.	SP.	Res.	OR-P.	CT-Pre.	CT-J.	SP.
$(1, 10^3)$	2.11e-06	1.77e-01	2.70e-14	4.07e-14	0.88	0.34	3	2.85e-15	9.41e-15	2.61	31.79	1
$(2, 10^3)$	5.59e-07	1.77e-01	2.06e-13	2.08e-13	0.84	0.51	5	1.32e-14	1.65e-14	2.44	82.06	3
$(3, 10^3)$	2.81e-05	2.41e-06	3.62e-13	4.46e-13	0.90	1.62	6	4.77e-15	8.94e-15	3.15	65.41	3
$(4, 10^3)$	7.34e-05	3.46e-04	3.43e-13	4.39e-13	0.90	1.48	6	4.96e-15	8.93e-15	2.95	64.77	2
$(5, 10^3)$	2.54e-05	2.11e-09	3.73e-13	4.70e-13	0.88	1.75	7	1.20e-14	9.01e-15	2.94	65.51	4
$(1, 10^4)$	2.96e-06	1.77e-01	2.59e-14	4.72e-14	0.84	0.34	3	2.82e-15	6.90e-15	2.74	32.27	1
$(2, 10^4)$	4.51e-07	1.77e-01	5.92e-14	5.83e-14	0.84	0.44	5	1.23e-14	1.64e-14	2.43	81.56	3
$(3, 10^4)$	2.28e-05	3.22e-07	3.70e-13	5.00e-13	0.89	1.67	6	3.78e-15	9.05e-15	3.07	67.53	3
$(4, 10^4)$	7.25e-05	3.46e-04	3.40e-13	4.23e-13	0.90	1.46	6	4.42e-15	9.00e-15	3.01	64.87	2
$(5, 10^4)$	2.25e-05	2.90e-10	3.70e-13	5.37e-13	0.89	1.68	6	4.20e-15	9.11e-15	2.80	67.79	4
$(1, 10^5)$	3.37e-06	1.77e-01	3.20e-14	5.32e-14	0.90	0.34	3	3.01e-15	6.87e-15	2.86	32.35	1
$(2, 10^5)$	4.59e-07	1.77e-01	2.12e-13	1.03e-13	0.85	0.52	5	1.25e-14	1.61e-14	2.42	82.63	3
$(3, 10^5)$	1.88e-05	4.03e-08	3.95e-13	6.87e-13	0.90	1.80	7	4.61e-15	9.23e-15	3.16	72.61	3
$(4, 10^5)$	7.29e-05	3.46e-04	3.40e-13	4.54e-13	0.90	1.46	6	4.70e-15	9.03e-15	2.98	64.77	2
$(5, 10^5)$	1.78e-05	2.24e-09	3.83e-13	7.27e-13	0.90	1.70	6	5.17e-15	9.14e-15	2.79	71.76	3
$(1, 10^6)$	3.64e-06	1.77e-01	2.93e-14	6.72e-14	0.82	0.27	2	2.87e-15	6.82e-15	2.94	32.53	1
$(2, 10^6)$	4.44e-07	1.77e-01	1.60e-13	1.58e-13	0.83	0.50	5	1.13e-14	1.63e-14	2.41	82.42	3
$(3, 10^6)$	1.63e-05	4.84e-09	3.82e-13	9.76e-13	0.89	1.91	8	3.81e-15	9.43e-15	3.04	78.58	5
$(4, 10^6)$	7.17e-05	3.46e-04	3.47e-13	4.20e-13	0.91	1.48	6	4.37e-15	8.90e-15	3.05	64.67	2
$(5, 10^6)$	1.21e-05	2.17e-10	3.84e-13	9.05e-13	0.89	1.90	8	1.06e-14	9.32e-15	2.81	76.41	5

Table 6.8: Numerical results for Example 6.2 with multiple eigenvalues and $n = 3072$.

			parallel Alg. 4.1					Alg. 4.1				
(mode, kappa)	$\|\mathrm{off}(Q^T AQ)\|_F$	$d(A)/(4\sqrt{2})$	Res.	OR-P.	CT-Pre.	CT-J.	SP.	Res.	OR-P.	CT-Pre.	CT-J.	SP.
$(1, 10^3)$	2.14e-06	1.77e-01	2.31e-14	1.65e-13	11.30	1.19	4	3.49e-15	2.02e-14	11.34	125.94	1
$(2, 10^3)$	5.50e-07	1.77e-01	5.89e-14	5.23e-14	11.04	4.78	8	1.52e-14	1.98e-14	11.11	328.61	3
$(3, 10^3)$	4.17e-05	1.60e-06	3.32e-13	5.24e-13	12.78	5.08	6	5.23e-15	1.10e-14	12.87	264.42	3
$(4, 10^3)$	1.10e-04	2.30e-04	4.07e-13	4.34e-13	12.27	7.11	7	5.33e-15	1.10e-14	12.45	258.16	2
$(5, 10^3)$	4.21e-05	1.78e-08	3.33e-13	5.58e-13	12.25	5.79	7	5.10e-15	1.10e-14	12.45	262.61	3
$(1, 10^4)$	3.03e-06	1.77e-01	1.47e-14	9.88e-14	11.96	1.33	4	3.55e-15	8.35e-15	11.73	129.88	1
$(2, 10^4)$	5.27e-07	1.77e-01	5.83e-14	4.99e-14	11.06	4.76	8	1.60e-14	2.01e-14	11.18	325.42	3
$(3, 10^4)$	3.39e-05	2.14e-07	3.15e-13	6.52e-13	12.90	5.10	6	4.74e-15	1.12e-14	12.98	275.07	3
$(4, 10^4)$	1.10e-04	2.30e-04	4.08e-13	4.37e-13	12.25	7.27	7	5.67e-15	1.10e-14	12.42	261.08	2
$(5, 10^4)$	3.11e-05	1.45e-09	3.03e-13	7.13e-13	12.12	5.61	7	4.83e-15	1.12e-14	12.23	275.09	4
$(1, 10^5)$	3.89e-06	1.77e-01	1.80e-14	2.10e-13	12.04	1.20	4	3.53e-15	8.35e-15	12.16	130.78	1
$(2, 10^5)$	6.90e-07	1.77e-01	5.53e-14	4.83e-14	11.01	4.78	8	1.64e-14	2.01e-14	11.10	328.93	3
$(3, 10^5)$	2.77e-05	2.67e-08	3.17e-13	7.55e-13	13.08	5.49	7	5.72e-15	1.13e-14	13.23	296.06	3
$(4, 10^5)$	1.08e-04	2.30e-04	4.21e-13	4.56e-13	12.45	7.25	7	6.65e-15	1.10e-14	12.40	261.20	2
$(5, 10^5)$	2.68e-05	7.81e-10	3.00e-13	8.20e-13	12.29	5.56	7	5.98e-15	1.13e-14	12.41	295.18	6
$(1, 10^6)$	4.40e-06	1.77e-01	1.75e-14	2.16e-13	12.52	1.26	5	3.52e-15	8.29e-15	12.33	130.52	1
$(2, 10^6)$	6.09e-07	1.77e-01	5.46e-14	5.43e-14	11.01	4.88	9	1.50e-14	2.00e-14	11.09	329.44	3
$(3, 10^6)$	2.38e-05	3.21e-09	3.03e-13	1.09e-12	13.25	6.10	8	5.12e-15	1.16e-14	13.51	332.16	6
$(4, 10^6)$	1.10e-04	2.30e-04	4.06e-13	4.29e-13	12.44	7.26	7	6.71e-15	1.10e-14	12.51	262.98	2
$(5, 10^6)$	2.23e-05	1.97e-11	2.92e-13	1.01e-12	12.36	5.91	8	7.13e-15	1.16e-14	12.35	323.93	6

Table 6.9: Numerical results for Example 6.2 with $n = 4096$.

			parallel Alg. 4.1					Alg. 4.1				
(mode, kappa)	$\|\mathrm{off}(Q^TAQ)\|_F$	$d(A)/(4\sqrt{2})$	Res.	OR-P.	CT-Pre.	CT-J.	SP.	Res.	OR-P.	CT-Pre.	CT-J.	SP.
$(1, 10^3)$	2.22e-06	1.77e-01	5.86e-14	6.15e-14	1.41	2.80	1	3.86e-15	3.36e-14	38.72	344.47	1
$(2, 10^3)$	1.13e-06	1.77e-01	2.67e-13	2.69e-13	1.36	4.21	2	1.64e-14	2.54e-14	37.70	1105.04	4
$(3, 10^3)$	5.67e-05	1.20e-06	7.53e-13	9.99e-13	1.49	14.49	4	7.51e-15	1.27e-14	39.08	706.47	3
$(4, 10^3)$	1.48e-04	1.73e-04	6.74e-13	8.08e-13	1.59	13.86	3	6.02e-15	1.27e-14	37.55	694.53	2
$(5, 10^3)$	5.28e-05	3.85e-09	7.65e-13	1.18e-12	1.73	14.46	4	5.82e-15	1.27e-14	37.69	714.97	3
$(1, 10^4)$	3.28e-06	1.77e-01	6.25e-14	6.50e-14	1.37	2.37	1	3.85e-15	9.98e-15	36.98	348.95	1
$(2, 10^4)$	6.16e-07	1.77e-01	3.03e-13	2.53e-13	1.44	4.46	2	1.66e-14	2.35e-14	35.99	887.64	3
$(3, 10^4)$	4.54e-05	1.60e-07	7.88e-13	1.13e-12	1.46	13.74	3	5.40e-15	1.28e-14	43.89	781.40	3
$(4, 10^4)$	1.47e-04	1.73e-04	6.73e-13	8.53e-13	1.57	14.07	3	8.07e-15	1.26e-14	43.43	711.09	2
$(5, 10^4)$	4.24e-05	2.19e-09	8.08e-13	1.22e-12	1.56	14.62	3	6.07e-15	1.28e-14	43.34	758.60	4
$(1, 10^5)$	4.35e-06	1.77e-01	5.44e-14	5.63e-14	1.49	2.37	1	3.88e-15	9.61e-15	38.26	349.81	1
$(2, 10^5)$	6.10e-07	1.77e-01	3.27e-13	3.25e-13	1.35	4.35	2	1.57e-14	2.52e-14	36.24	1152.83	4
$(3, 10^5)$	3.71e-05	2.00e-08	7.88e-13	1.61e-12	1.47	13.61	3	6.29e-15	1.31e-14	45.85	841.83	4
$(4, 10^5)$	1.43e-04	1.73e-04	6.60e-13	8.13e-13	1.53	14.57	3	6.10e-15	1.26e-14	43.82	725.35	2
$(5, 10^5)$	4.85e-05	7.00e-10	7.72e-13	1.55e-12	1.51	13.35	3	7.49e-15	1.31e-14	42.22	853.29	5
$(1, 10^6)$	5.06e-06	1.77e-01	5.68e-14	5.85e-14	1.54	2.36	1	3.95e-15	9.58e-15	37.15	342.74	1
$(2, 10^6)$	5.36e-07	1.77e-01	2.60e-13	2.62e-13	1.38	5.01	2	1.65e-14	2.30e-14	34.75	855.35	3
$(3, 10^6)$	3.21e-05	2.40e-09	8.11e-13	2.20e-12	1.52	13.90	3	6.95e-15	1.34e-14	39.27	873.81	7
$(4, 10^6)$	1.47e-04	1.73e-04	6.74e-13	8.56e-13	1.52	14.18	3	7.70e-15	1.26e-14	39.87	709.16	2
$(5, 10^6)$	3.00e-05	4.78e-11	7.96e-13	2.08e-12	1.52	14.11	3	1.46e-14	1.33e-14	39.45	896.90	6

Table 6.10: Numerical results for Example 6.3 with $m \times n = 2048 \times 1024$.

Trule	Alg. 5.1						Alg. 5.2					
(mode, kappa)	Res.	OR-U.	OR-V.	JU.	CT.	SP.	Res.	OR-U.	OR-V.	JU.	CT.	SP.
$(1, 10^3)$	1.43e-15	5.41e-15	1.51e-14	0.45N	97.20	19	2.10e-15	5.60e-15	1.76e-14	1.27N	82.17	16
$(2, 10^3)$	2.39e-15	5.45e-15	4.68e-15	0.01N	31.18	7	3.93e-15	4.82e-15	9.81e-15	0.02N	67.47	15
$(3, 10^3)$	1.02e-14	5.49e-15	4.07e-13	10.78N	171.35	18	8.84e-15	3.04e-15	1.26e-13	1.95N	39.01	5
$(4, 10^3)$	1.13e-14	4.35e-15	3.63e-13	9.45N	142.77	14	1.25e-14	2.90e-15	1.28e-13	1.98N	35.01	4
$(5, 10^3)$	9.94e-15	5.25e-15	4.43e-13	11.01N	177.33	19	9.41e-15	3.04e-15	1.26e-13	1.95N	34.76	4
$(1, 10^4)$	2.99e-15	6.18e-15	3.67e-14	2.34N	86.50	15	3.29e-15	6.07e-15	4.12e-14	3.36N	86.63	13
$(2, 10^4)$	2.53e-15	5.02e-15	3.42e-15	0.01N	35.54	8	3.22e-15	4.90e-15	7.28e-15	0.02N	62.86	14
$(3, 10^4)$	9.53e-15	6.39e-15	4.18e-13	11.58N	190.71	21	8.38e-15	4.44e-15	1.30e-13	1.97N	39.20	5
$(4, 10^4)$	1.13e-14	3.80e-14	3.74e-13	9.52N	142.80	14	1.23e-14	2.95e-15	1.28e-13	1.98N	34.92	4
$(5, 10^4)$	9.10e-15	5.18e-15	4.31e-13	11.67N	188.75	21	8.81e-15	4.36e-15	1.28e-13	2.00N	38.96	5
$(1, 10^5)$	3.64e-15	6.62e-15	1.24e-13	4.19N	91.54	13	4.11e-15	6.29e-15	9.98e-14	4.77N	97.02	13
$(2, 10^5)$	2.13e-15	4.84e-15	2.68e-15	0.01N	30.86	7	3.00e-15	4.84e-15	4.28e-15	0.02N	61.92	14
$(3, 10^5)$	1.09e-14	7.06e-15	4.16e-13	12.38N	207.45	24	8.64e-15	7.37e-15	1.61e-13	2.26N	49.73	7
$(4, 10^5)$	1.10e-14	3.93e-15	3.55e-13	9.44N	139.42	14	1.25e-14	3.15e-15	1.27e-13	1.98N	34.40	4
$(5, 10^5)$	8.98e-15	6.64e-15	4.64e-13	12.29N	207.07	24	7.90e-15	6.47e-15	1.47e-13	2.15N	44.43	6
$(1, 10^6)$	4.34e-15	6.09e-15	1.96e-13	5.44N	101.81	13	4.32e-15	5.69e-15	1.37e-13	5.68N	104.21	13
$(2, 10^6)$	1.87e-15	4.91e-15	6.27e-15	0.01N	35.14	8	2.85e-15	4.88e-15	4.29e-15	0.02N	57.50	13
$(3, 10^6)$	8.72e-15	8.47e-15	4.23e-13	13.41N	229.25	27	7.70e-15	9.11e-15	1.89e-13	2.51N	56.01	8
$(4, 10^6)$	1.11e-14	3.86e-15	3.79e-13	9.49N	144.61	15	1.23e-14	2.77e-15	1.28e-13	1.99N	34.22	4
$(5, 10^6)$	9.11e-15	7.38e-15	4.64e-13	13.83N	237.56	28	9.09e-15	7.97e-15	1.66e-13	2.36N	50.55	7

[3] A. Abdelfattah, H. Anzt, E. G. Boman, et al., *A survey of numerical linear algebra methods utilizing mixed-precision arithmetic*, Int. J. High Perform. Comput. Appl., 35 (2021), pp. 344–369.

[4] E. Anderson, Z. Bai, C. Bischof, et al., *LAPACK User's Guide,* 3rd ed., SIAM, Philadelphia, 1999.

[5] M. Baboulin, A. Buttari, J. Dongarra, J. Kurzak, J. Langou, J. Langou, P. Luszczek, S. Tomov, *Accelerating scientific computations with mixed precision algorithms*, Comput. Phys. Commun., 180 (2009), pp. 2526–2533.

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

[7] M. Bečka, G. Okša, *Preconditioned Jacobi SVD Algorithm Outperforms PDGESVD*, In: R. Wyrzykowski, E. Deelman, J. Dongarra, K. Karczewski, (eds.), Parallel Processing and Applied Mathematics. PPAM 2019, Lecture Notes in Computer Science, Vol. 12043, Springer, Cham, 2020.

[8] H. Bowdler, R. S. Martin, C. Reinsch, J. H. Wilkinson, *The QR and QL algorithms for symmetric matrices*, Numer. Math., 11 (1968), pp. 293–306.

[9] E. Carson, N. J. Higham, *Accelerating the solution of linear systems by iterative refinement in three precisions*, SIAM J. Sci. Comput., 40 (2018), pp. A817–A847.

[10] M. P. Connolly, N. J. Higham, T. Mary. Stochastic rounding and its probabilistic backward error analysis. *SIAM J. Sci. Comput.*, 43:A566–A585, 2021.

[11] B. N. Datta, *Numerical Linear Algebra and Applications*, 2nd ed., SIAM, Philadelphia, 2010.

[12] E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*, J. Comput. Phys., 17 (1975), pp. 87–94.

[13] P. I. Davies, N. J. Higham, F. Tisseur. Analysis of the cholesky method with iterative refinement for solving the symmetric definite generalized eigenproblem. *SIAM J. Matrix Anal. Appl.*, 23:472–493, 2001.

[14] P. P. M. de Rijk, *A one-sided Jacobi algorithm for computing the singular value decomposition on a vector computer*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 359–371.

[15] J. Demmel, K. Veselić, *Jacobi's method is more accurate than QR*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1204–1245.

[16] I. S. Dhillon, B. N. Parlett, *Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices*, Linear Algebra Appl., 387 (2004), pp. 1–28.

[17] J. J. Dongarra, *Algorithm 589 sicedr: A FORTRAN subroutine for improving the accuracy of computed matrix eigenvalues*, ACM Trans. Math. Software, 8 (1982), pp. 371–375.

[18] J. J. Dongarra, C. B. Moler, J. H. Wilkinson, *Improving the accuracy of computed eigenvalues and eigenvectors*, SIAM J. Numer. Anal., 20 (1983), pp. 23–45.

[19] Z. Drmač, *Implementation of Jacobi rotations for accurate singular value computation in floating point arithmetic*, SIAM J. Sci. Comput., 18 (1997), pp. 1200–1222.

[20] Z. Drmač, K. Veselić, *New fast and accurate Jacobi SVD algorithm. I*, SIAM J. Matrix Anal. Appl., 29 (2008), pp. 1322–1342.

[21] G. E. Forsythe, P. Henrici, *The cyclic Jacobi method for computing the principal values of a complex matrix*, Trans. Amer. Math. Soc., 94 (1960), pp. 1–23.

[22] G. H. Golub, C. F. Van Loan, *Matrix Computations*, 4th ed., The Johns Hopkins University Press, Baltimore, 2013.

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

[24] K. K. Gupta, *Solution of eigenvalue problems by Sturm sequence method*, Internat. J. Numer. Methods Engrg., 4 (1972), pp. 379–404.

[25] W. Gao, Y. Ma, M. Shao, *A mixed precision Jacobi SVD algorithm*, arXiv:2209.04626, 2022.

[26] P. Henrici, *On the speed of convergence of cyclic and quasicyclic Jacobi methods for computing eigenvalues of Hermitian matrices*, J. Soc. Indust. Appl. Math., 6 (1958), pp. 144–162.

[27] M. R. Hestenes, *Inversion of matrices by biorthogonalization and related results*, J. Soc. Indust. Appl. Math., 6 (1958), pp. 51–90.

[28] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.

[29] N. J. Higham, T. Mary. A new approach to probabilistic rounding error analysis. *SIAM J. Sci. Comput.*, 41:A2815–A2835, 2019.

[30] N. J. Higham, T. Mary, *Mixed precision algorithms in numerical linear algebra*, Acta Numer., 31 (2022), pp. 347–414.

[31] C. G. J. Jacobi, *Über ein leichtes Verfahren, die in der Theorie der Säcularstörungen vorkommenden Gleichungen numerisch aufzulösen*, Crelle's Journal für reine und angew. Math., 30 (1846), pp. 51–95.

[32] C. T. Kelley, *Newton's method in mixed precision*, SIAM Rev., 64 (2022), pp. 191–211.

[33] S. Larsson, V. Thomée, *Partial Differential Equations with Numerical Methods*, Springer, Berlin, 2003.

[34] T. Ogita, K. Aishima, *Iterative refinement for symmetric eigenvalue decomposition*, Jpn. J. Ind. Appl. Math., 35 (2018), pp. 1007–1035.

[35] T. Ogita, K. Aishima, *Iterative refinement for symmetric eigenvalue decomposition II: clustered eigenvalues*. Jpn. J. Ind. Appl. Math., 36 (2019), pp. 435–459.

[36] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998.

[37] M. Petschow, E. S. Quintana-Ortí, P. Bientinesi, *Improved accuracy and parallelism for MRRR-based eigensolvers—a mixed precision approach*, SIAM J. Sci. Comput., 36 (2014), pp. C240–C263.

[38] A. Ruhe, *On the quadratic convergence of a generalization of the Jacobi method to arbitrary matrices*, BIT, 8 (1968), pp. 210–231.

[39] A. Schönhage, *On the quadratic convergence of the Jacobi process*, Numer. Math., 6 (1964), pp. 410–412.

[40] I. Slapničar, *Symmetric matrix eigenvalue techniques*, In L. Hogben (ed.), Handbook of Linear Algebra, Vol. 55, pp. 1–26, CRC Press: Boca Raton, FL, USA, 2014.

[41] G. L. G. Sleijpen, H. A. Van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[42] Y. Tsai, *Mixed-Precision Numerical Linear Algebra Algorithms: Integer Arithmetic Based LU Factorization and Iterative Refinement for Hermitian Eigenvalue Problem*, PhD thesis, University of Tennessee, 2020.

[43] H. P. M. van Kempen, *On the quadratic convergence of the special cyclic Jacobi method*, Numer. Math., 9 (1966), pp. 19–22.

[44] J. H. Wilkinson, *Note on the quadratic convergence of the cyclic Jacobi process*, Numer. Math., 4 (1962), pp. 296–300.

[45] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon, Oxford, U.K., 1965.

[46] I. Yamazaki, S. Tomov, J. Dongarra, *Mixed-precision Cholesky QR factorization and its case studies on multicore CPU with multiple GPUs*, SIAM J. Sci. Comput., 37 (2015), pp. C307–C330.

[47] C. Yang, *Solving large-scale eigenvalue problems in SciDAC applications*, J. Phys. Conf. Ser., 16 (2005), pp. 425–434.

[48] L. M. Yang, A. Fox, G. Sanders, *Rounding error analysis of mixed precision block Householder QR algorithms*, SIAM J. Sci. Comput., 43 (2021), pp. A1723–A1753.