Journal of Mathematical Modeling
Vol. 11, No. 2, 2023, pp. 301–322. Research Article

JMM

Symmetric-diagonal reductions as preprocessing for symmetric positive definite generalized eigenvalue solvers

Morad Ahmadnasab*

Department of Mathematics, Faculty of Science, University of Kurdistan, 66177-15175, Sanandaj, Iran
Email(s): M.Ahmadnasab@uok.ac.ir

Abstract. We discuss some potential advantages of the orthogonal symmetric-diagonal reduction in two main versions of the Schur-QR method for symmetric positive definite generalized eigenvalue problems. We also advise and use the appropriate reductions as preprocessing on the solvers, mainly the Cholesky-QR method, of the considered problems. We discuss numerical stability of the methods via providing upper bound for backward error of the computed eigenpairs and via investigating two kinds of scaled residual errors. We also propose and apply two kinds of symmetrizing which improve the stability and the performance of the methods. Numerical experiments show that the implemented versions of the Schur-QR method and the preprocessed versions of the Cholesky-QR method are usually more stable than the Cholesky-QR method.

Keywords: Symmetric definite generalized eigenvalue problem, Cholesky-QR method, Schur-QR method, QZ method, Rounding error analysis.
AMS Subject Classification 2010: 65F15, 15A21, 15A22.

1 Introduction

The symmetric positive definite (SPD) generalized eigenvalue problem (SPDGEVP) $Ax = \lambda Bx$, where $A, B \in \mathbb{R}^{n\times n}$ are symmetric and B (or A) is positive definite, arises in many applications in science and engineering [5]. The QZ algorithm [14] can be used to solve this problem. This method is numerically stable but, as it destroys the symmetry of the problem, there is no guarantee that the QZ method produces real eigenpairs in floating point arithmetic.

When $A \in \mathbb{R}^{n\times n}$ is symmetric and $B \in \mathbb{R}^{n\times n}$ is SPD, then we can use the Wilkinson's approach,

*Corresponding author
Received: 1 February 2023 / Revised: 7 March 2023 / Accepted: 11 March 2023
DOI: 10.22124/JMM.2023.23734.2120

known as the Cholesky-QR method, which solve

$$Ax = \lambda Bx, \tag{1}$$

via computing the eigendecomposition of the symmetric matrix $L^{-1}AL^{-T}$, where L is the lower triangular matrix in the Cholesky factorization of B, i.e., $B = LL^T$ [23]. This approach is numerically unstable [23] and for the most popular versions of this method, the backward error bounds contain the condition number of B [6].

For a second known approach of Wilkinson [23, page 337], which is known as the Schur-QR method [5], let us consider the eigendecomposition

$$B = VDV^T, \tag{2}$$

for the matrix B in (1) where D is a diagonal matrix of eigenvalues and V is a unitary matrix whose columns are the corresponding eigenvectors. Wilkinson proposed to use the eigendecomposition (2) to reduce (1) to symmetric standard eigenvalue problem (SSEVP)

$$Cy = \lambda y, \tag{3}$$

where

$$C = D^{-1/2}V^TAVD^{-1/2}, \quad \text{and} \quad y = D^{1/2}V^Tx.$$

Wilkinson also discussed in [23, page 337] the possibility and stability benefit of computing the eigendecomposition (2) by the methods of Jacobi, Givens, or Householder, where we shall have V in the factored form, and the real symmetric matrix C may be computed from the factors.

The idea of finding the factorization $B = SS^T$ (via applying Jacobi method on B) and using it to transform (1) to a SSEVP have been used by Golub on Illiac [17].

It is stated in [23, page 344] that any possible ill-conditioning of B is concentrated in the small elements of D in (2) and the matrix C in (3) has certain number of rows and columns with large elements. Therefore, the eigenvalues of the pencil $A - \lambda B$ of normal size are more likely to be preserved. As a heuristic, if the diagonal entries of D are ordered from smallest to largest (upward sorting), then the large entries of C in (3) are concentrated in the upper-left corner. Then, the small eigenvalues of C can be computed without excessive roundoff error contamination [9, page 464]. Also, for preserving the large eigenvalues of C, we should order the diagonal entries of D from largest to smallest (downward sorting) [20, page 233 -235].

A technique for the stable deflation of eigenpairs (1) is proposed in [4]. They viewed the problem as the simultaneous LDL^T factorization of the matrices A and B, where L is not necessarily a triangular matrix.

Iterative refinement, in fixed or extended precision, has been applied successfully in [21] for improving the forward and backward errors of approximate solutions of problem (1).

Davies et al. in [6] applied the Cholesky-Jacobi method on (1). Using their implementation strategy, some rigorous backward error bounds have been derived which are significantly smaller than bounds involving a factor of the condition number of B when B is ill-conditioned [6]. This has another advantage which, thanks to the Jacobi method, there is no need to think of a heuristic downward (or upward) sorting suggested in [9] and [20] for the Schur-QR method.

The methods such as the ones in [1, 6, 7] are designed to improve eigenpairs individually, therefore they need $O(n^4)$ arithmetic operations when refinement of all eigenpairs are requiered. In [15], an efficient refinement algorithm (with $O(n^3)$ arithmetic operations) for improving the accuracy of all eigenvectors of real symmetric matrix A has been proposed. To overcome the necessary need to the initial eigenvalue gap in [15], a new algorithm in [16] has been developed that can refine approximate eigenvectors corresponding to clustered eigenvalues. As they explained, the results of both [15, 16] are applicable for SPDGEVPs too.

In this paper, depend on the SPD property of either matrix A or matrix B, we study two main versions of the Schur-QR method for solving the symmetric generalized eigenvalue problems $Ax = \lambda Bx$. The first version that we study here is related to the problems whose matrix B is SPD. The second version, designed likewise but for the symmetric problems $Ax = \lambda Bx$ with A being SPD. It is true that this second group of the problems could be treated via the relations between the eigenpairs of the matrix pencils $P(\lambda) = A - \lambda B$ and the reverse matrix pencil $revP(\lambda) = -\lambda P(\lambda^{-1}) = (B - \lambda A)$ [12]. Nevertheless, because we consider and solve some examples whose both matrices A and B are SPD with different condition numbers, we shall classify the above versions as two different versions of the Schur-QR method.

For implementing each of the above versions, we proposes either an appropriate version of orthogonal symmetric-diagonal reduction as preprocessing for SPDGEVP solvers (mainly the Cholesky-QR method) or a reduction to symmetric standard eigenvalue problem. The former (resp., the latter) is the preprocessed Cholesky-QR method (resp., the Schur-QR method). We shall see that the preprocessed Cholesky-QR method is theoretically equivalent to the Schur-QR method. One aim of this paper is to report performance and stability potentials of the Schur-QR method, which to the best of our knowledge remain uncovered. We will suggest and apply two kinds of symmetrizing in Section 5, which in practice prevent the preprocessed Cholesky-QR method and the Schur-QR method from being unstable for the same examples.

Our symmetric-diagonal reductions are different from the one used in [22] and some of its references. In [22], as first phase, a non-orthogonal symmetric-diagonal reduction has been used to reduce a symmetric indefinite matrix pair (A, B) to tridiagonal-diagonal form by congruence transformations.

This research was motivated by the need for improving the stability and at the same time preserving or improving the computational efficiency of Schur-QR method for solving a SPDGEVP that enjoys the structure of the problem. Besides, it is well known that the Cholesky-QR method is advisable whenever the matrix B is reasonably well conditioned with respect to inversion or when B has a simpler structure than A, as when B is diagonal, see Section 5.2 in [2].

The rest of the paper is organized as follows. In Section 2, we introduce two versions of an orthogonal symmetric-diagonal reduction. We also discuss two versions of the Schur-QR method. At the last part of Section 2, we discuss the equivalency of the presented methods. In Section 3, we analyze the performance of the algorithms obtained in Section 2. Section 4 is devoted to the rounding error analysis of the methods where we provide upper bounds for backward errors of the eigenpairs computed by the versions which do not use any kinds of symmetrizing. Numerical experiments, including a study of scaled residual errors together with backward errors of computed eigenpairs for some representative examples from literature, are given in Section 5. In this section, we suggest and apply two kinds of symmetrizing for improving the stability and the performance of the methods. Conclusions are given in Section 6.

2 Methods outline

Suppose both A and B are symmetric and B (or A) is positive definite. We shall describe two main versions of an orthogonal symmetric-diagonal reduction, based on the eigendecomposition of A or B, the one that is SPD. Each version can be used either as a preprocessing for SPDGEVP solvers or as the first step to reduce SPDGEVP problem to a SSEVP.

In this section, we merely introduce and explain the theory of the methods. Further technical notes about performance and stability of these methods will be discussed in Section 3 and Section 4, respectively. Some more modifications for improving the performance and the stability of the methods will be proposed and used in Section 5.

2.1 Preprocessed Cholesky-QR method

Assume $A \in \mathbb{R}^{n\times n}$ is symmetric and $B \in \mathbb{R}^{n\times n}$ is SPD. Here, we shall introduce a preprocessed Cholesky-QR method for solving the problem

$$Ax = \lambda Bx. \tag{4}$$

Let

$$B = U\Sigma U^T, \tag{5}$$

be the eigendecomposition of the SPD matrix B, where U is an orthogonal (unitary) $n \times n$ matrix and Σ is a diagonal $n \times n$ matrix whose diagonal entries are eigenvalues of B. Using (5), one can rewrite (4) as follows

$$Ax = \lambda U\Sigma U^T x.$$

This is equivalent to

$$AUU^T x = \lambda U\Sigma U^T x,$$

which is in turn equivalent to

$$U^T AUU^T x = \lambda \Sigma U^T x.$$

Denoting $\hat{x} := U^T x$, we see that the eigenvalues of problem (4) are the eigenvalues of

$$U^T AU\hat{x} = \lambda \Sigma \hat{x}. \tag{6}$$

This suggests the preprocessing part of the first method for solving problem (4). Problem (6) can be solved by either of the Cholesky-QR method or the QZ method. But as theoretically $U^T AU$ is symmetric and Σ is diagonal with positive diagonal entries, we mostly opt to use the Cholesky-QR method. In this case, we shall recover the eigenvectors of problem (4) from those of (6).

To ensure a better stability behaviour of the QR algorithm, we add some upward (resp. downward) sorting on the entries of matrix Σ when the backward stability of the small (resp. large) eigenvalues of (4) is the first concern. At this case, we apply appropriate column reorder on the matrix U.

The entire algorithm for computing the eigenpairs of the problem (4), using eigendecomposition of B and the Cholesky-QR method, is summed up as shown in Algorithm 1. Obviously, calling and using either of the Cholesky-QR method or the QZ method for solving problem (6) imposes additional and unnecessary costs and is not recommended in practice. Nevertheless, we have one main aim from such calls (such as the call in Step 3 of Algorithm 1). The aim is revisiting and investigating the improvement in the properties (i.e., stability and possibly performance) of the Cholesky-QR algorithm when applying on problems with simpler (i.e., diagonal) structure matrix B.

Algorithm 1 *Computing the eigenpairs of the problem (4) using preprocessed Cholesky-QR method, when B is SPD*

Inputs: Symmetric matrix A and SPD matrix B.
Outputs: Eigenpairs of problem (4).

1. Compute the eigendecomposition (5) for B. A predicted sorting for the diagonal entries of Σ followed by the appropriate reordering on the columns of U are to be included,
2. Construct $\mathscr{A} = U^T A U$, and assign $\mathscr{B} = \Sigma$, for U and Σ obtained in the first step.
3. Solve $\mathscr{A}\hat{x} = \lambda \mathscr{B}\hat{x}$ using the Cholesky-QR method.
4. The eigenvalues of problem (4) are the computed eigenvalues in Step 3 and their associated eigenvectors are $x = U\hat{x}$.

2.2 Reduction to symmetric standard eigenvalue problem when B is SPD

In this section, we explain the Schur-QR method for problems with SPD matrix B. Assume $A \in \mathbb{R}^{n\times n}$ is symmetric and $B \in \mathbb{R}^{n\times n}$ is SPD. If we enrich the Step 2 of Algorithm 1 by the process of transforming problem (6) to a SSEVP (i.e., by using the same idea as (3)), then instead of applying Cholesky-QR in Step 3, we need to solve a SSEVP.

Similar to what we have in Section 2.1, and depends on our need, we use upward (resp. downward) sorting on the entries of matrix Σ and then make appropriate reordering on the columns of matrix U.

In the following, we detail the process of reducing the SPDGEVP problem (6) to the desired SSEVP. Let

$$\Sigma_s = \operatorname{diag}(\sqrt{\sigma_1}, \ldots, \sqrt{\sigma_n}),$$

and

$$\Sigma_{is} = \Sigma_s^{-1} = \operatorname{diag}(1/\sqrt{\sigma_1}, \ldots, 1/\sqrt{\sigma_n}), \tag{7}$$

where σ_i for $i = 1, \ldots, n$ are the eigenvalues (and at the same time singular values) of the matrix B which are the diagonal entries of matrix Σ in (5). Considering the facts that $\hat{x} = U^T x$ and $\Sigma_s^2 = \Sigma$, (6) is equivalent to

$$\Sigma_{is} U^T A U \Sigma_{is} \Sigma_s U^T x = \lambda \Sigma_s U^T x.$$

By denoting $y = \Sigma_s U^T x$ and $\mathscr{S} = \Sigma_{is} U^T A U \Sigma_{is}$, we get the following SSEVP:

$$\mathscr{S} y = \lambda y. \tag{8}$$

Eigenvalues of problem (4) are the eigenvalues of (8) and their associated eigenvectors should be computed by $x = U\Sigma_{is} y$. Algorithm 2 shows the necessary steps for solving problem (4) via the reductions (6) and (8). To get to the desired SSEVP, we merely exploit the diagonal entries of Σ_s and Σ_{is} to conclude better performance.

Algorithm 2 *Computing the eigenpairs of the problem (4) via transforming (6) to a SSEVP, when B is SPD*

Inputs: Symmetric matrix A and SPD matrix B.
Outputs: Eigenpairs of problem (4).

1. Compute the eigendecomposition (5) for B. A predicted sorting for the diagonal entries of Σ followed by the appropriate reordering on the columns of U are to be included.
2. Construct $\mathscr{S} = \Sigma_{is} U^T A U \Sigma_{is}$ for U obtained in Step 1 and Σ_{is} defined in (7).
3. Solve $\mathscr{S} y = \lambda y$ using the symmetric QR method,
4. The eigenvalues of problem (4) are the computed eigenvalues in Step 3 and their associated eigenvectors are to be computed by $x = U \Sigma_{is} y$.

2.3 Orthogonal symmetric-diagonal reduction

Assume that $B \in \mathbb{R}^{n\times n}$ is symmetric and $A \in \mathbb{R}^{n\times n}$ is SPD. Instead of introducing a direct reduction on

$$Ax = \lambda Bx, \tag{9}$$

we can first change the roles of A and B by using the following displacement

$$T := B,\ B := A,\ \textit{and}\ A := T. \tag{10}$$

Then, we shall call Algorithm 1 to solve the new problem $Ax = \lambda Bx$. The eigenvalues of problem (9) are the reciprocals of the computed eigenvalues by Algorithm 1, and their associated eigenvectors are to be computed by $x = U\hat{x}$. Algorithm 3 shows the overall steps explained above.

Algorithm 3 *Computing the eigenpairs of the problem (9) using preprocessed Cholesky-QR method, when A is SPD*

Inputs: SPD matrix A and symmetric matrix B.
Outputs: Eigenpairs of problem (9).

1. Do the displacement (10),
2. Solve the new problem $Ax = \lambda Bx$ by Algorithm 1,
3. The eigenvalues of problem (9) are the reciprocals of the computed eigenvalues in Step 2, and their associated eigenvectors are the same as what computed in Step 2.

2.4 Reduction to symmetric standard eigenvalue problem when A is SPD

Assume that $B \in \mathbb{R}^{n\times n}$ is symmetric and $A \in \mathbb{R}^{n\times n}$ is SPD. This section for problem (9) is similar to Section 2.2 for problem (4). Similar to Section 2.3, we first use displacment (10). Then we call Algorithm 2 to solve the new problem $Ax = \lambda Bx$. The eigenvalues of problem (9) are the reciprocals of the

computed eigenvalues by Algorithm 2 and their associated eigenvectors are the same as those computed by Algorithm 2. Algorithm 4 illustrates the necessary steps for solving the problem (9).

Algorithm 4 *Computing the eigenpairs of problem* (9) *via transforming to a SSEVP when A is SPD*

Inputs: Symmetric matrix B and SPD matrix A.
Outputs: Eigenpairs of the problem (9).

1. Do the displacments (10).
2. Solve the new problem $Ax = \lambda Bx$ by Algorithm 2.
3. The eigenvalues of problem (9) are the reciprocals of the computed eigenvalues in Step 2 and their associated eigenvectors are the same as those computed in Step 2.

2.5 The equivalency of the presented methods

The idea of the Algorithm 1 (resp. 3) theoretically is the same as those of the Algorithm 2 (resp. 4). These uncover the following interesting facts.

Theorem 1. *a) The preprocessed Cholesky-QR method (Algorithm 1) is theoretically equivalent to the Schur-QR method (Algorithm 2). b) Algorithm 3 is theoretically equivalent to Algorithm 4.*

Proof. To prove case a), we can see that Step 3 of the Algorithm 1 includes two sub-steps: First is computing Cholesky decomposition of $\mathscr{B} = \Sigma_s \Sigma_s^T$. Second is computing the matrix $\mathscr{S}$ as what it is in Step 2 of Algorithm 2. This means that both Algorithm 1 and Algorithm 2 reduce the main problem to an identical standard eignevalue problem with the same way of eigenvector recovery.

For the case b), we should follow the same way as what we do for the case a), so we do not explain it.

□

3 Performance of the algorithms

Here we shall analyze the complexity of Algorithms 1. Algorithm 3 has the same complexity, except that in Step 3 it needs n more divisions. Algorithm 2 and Algorithm 4 have almost the same complexity as those of Algorithm 1 and Algorithm 3 respectively, but in both Algorithm 2 and Algorithm 4, the unnecessary operations in computing Cholesky factorization are avoided, so we expect better performance for them.

The process of computing eigenpairs of problem (4) by Algorithm 1 has four main steps. Below, the necessary operations for each step of Algorithm 1 are shown:

Step 1 involves

1-1. $\sim \frac{20}{3}n^3 + n^2$ flops for eigendecompostion,

1-2. $\sim nlog(n)$ flops for sorting eigenvalues,

1-3. and $2n^2+n$ memory locations for A, U and Σ.

Step 2 including matrix multiplications, requires

2-1. $4n^3-2n^2$ (or $\sim n^{2.807}$ in the case of using Strassen's algorithm) flops, for computing U^TAU.

Step 3 Cholesky-QR needs

3-1. $\sim 12n^3$ flops for computing eigenpairs of (6),

3-2. and n memory locations for eigenvalues (the associated eigenvectors can be stored in place of the columns of matrix Σ).

Step 4 needs

4-1. $2n^3-n^2$ flops for $x_i = U\hat{x}_i$, $i = 1,\ldots,n$ (x_i is the eigenvector associated with the eigenvalue λ_i).

According to this analysis and considering the similar steps in Algorithm 1 and Algorithm 3, the total number of flops required by Algorithm 1 (resp., by Algorithm 3), when we consider $4n^3-2n^2$ flops for the involved matrix multiplications, is asymptotic to $\frac{74}{3}n^3-2n^2$ (resp., to $\frac{74}{3}n^3-2n^2+n$). The number of memory locations in each one of Algorithm 1 and Algorithm 3 is $2n^2+2n$.

On the other side, the QZ method (which does not use the symmetry and symmetric definiteness of the involved matrices) needs about $46n^3$ flops [2] for solving either of problem (4) or problem (9). For problem (4), the Cholesky-QR method takes approximately $12n^3$ flops [9]. These facts are supported by the numerical experiments in Section 5.

4 Numerical stability

We discuss numerical stability of Algorithm 1. Almost the same (with reverse roles for A and B) is true for Algorithm 3. We expect the stability behaviour of Algorithm 2 (resp., Algorithm 4) to be very close to those of Algorithm 1 (resp., Algorithm 3), and hence we do not analyze them separately.

To discuss the numerical stability of Algorithm 1, we first analyze the rounding errors of the decomposition and the reduction introduced during the Steps 1 and 2 of Algorithm 1. Then, by analyzing the rounding errors of Steps 3 and 4 of the algorithm, we give upper bound on the backward error of the computed eigenpairs. We use the standard model of floating point arithmetic [11]:

$$fl(x \text{ op } y) = (x \text{ op } y)(1+\delta), \quad |\delta| \le u, \quad \text{op} = +,-,*,/,$$

where u is the unit roundoff. In what follows, by $\|\cdot\|_\infty$ and $\|\cdot\|_2$ we mean ∞-norm, and 2-norm vector norm (or their corresponding subordinate matrix norm), respectively. By the condition number of matrix X, we mean $k(X) = \|X\|_\infty \|X^{-1}\|_\infty$.

Let $\tilde{U}\tilde{\Sigma}\tilde{U}^T$ be the computed eigendecomposition in (5). Then, following the backward error analysis introduced in [3] for eigenvalue decomposition of real symmetric matrices based on orthogonal decompositions, we denote the residual associated with the computed decomposition $\tilde{U}\tilde{\Sigma}\tilde{U}^T$ by

$$\mu_R = \|B\tilde{U} - \tilde{U}\tilde{\Sigma}\|_2,$$

and the departure from orthogonality by

$$\mu_O = \|I - \tilde{U}^T \tilde{U}\|_2.$$

We define a backward error ΔB by

$$B + \Delta B = \tilde{U}\tilde{\Sigma}\tilde{U}^T. \tag{11}$$

As we have [3]

$$\|\Delta B\|_\infty \leq \mu_R \|\tilde{U}\|_\infty + \mu_O \|B\|_\infty,$$

therefore $\|\Delta B\|_\infty$ might be small if both μ_R and μ_O are small. Let us factor $\tilde{U} = WZ$ for orthogonal W. Using this, we can define a symmetric backward error ΔB^W via

$$B + \Delta B^W = W\tilde{\Sigma}W^T.$$

Let us denote

$$\mu_R^W = \|BW - W\tilde{\Sigma}\|_\infty, \quad \mu_O^W = \|\tilde{U} - W\|_\infty.$$

Then the decomposition $\tilde{U}\tilde{\Sigma}\tilde{U}^T$ is close to an eigenvalue decomposition of some symmetric matrix if μ_O^W is small, because $\|\Delta B - \Delta B^W\|_\infty \leq \mu_O^W \|\tilde{\Sigma}\|_\infty (1 + \|\tilde{U}\|_\infty)$ [3]. To ensure small μ_O^W for almost orthogonal matrices, we should use either of the polar factorization of $\tilde{U}$ or QR decompositon of appropriately scaled $\tilde{U}$ [3]. When μ_R^W is also small, then $\tilde{U}$ and $\tilde{\Sigma}$ are close to an eigenvalue decomposition of a nearby symmetric matrix [3].

When we compute $\mathscr{A}$ in the second step of Algorithm 1, we get

$$\tilde{\mathscr{A}} = \tilde{U}^T (A + \Delta A_1)\tilde{U} \tag{12}$$

instead. Considering (11) and applying standard results [9] on the product of matrices, after some algebraic simplifications, we get

$$\|\Delta A_1\|_\infty \leq a_n u k(\tilde{U}) \{\|A\|_\infty + (\mu_R \|\tilde{U}\|_2 + \mu_O \|B\|_2)\} + O(u^2). \tag{13}$$

When μ_R and μ_O are small enough, the most effective role for the upper bound in (13) goes to its first term, i.e., to $a_n u k(\tilde{U})\|A\|_\infty$.

Now, the same analysis as those in [6] could be considered when the Cholesky-QR method is used for solving the problem

$$\tilde{\mathscr{A}}\hat{x} = \lambda \tilde{\mathscr{B}}\hat{x}, \tag{14}$$

with $\tilde{\mathscr{A}}$ in (12) and $\tilde{\mathscr{B}} = \tilde{\Sigma}$ in (11). To this end, let

$$\tilde{\mathscr{B}} = L_s L_s^T, \tag{15}$$

be computed Cholesky factorization of matrix $\tilde{\mathscr{B}}$. As it is shown in [6], when we use a normwise backward stable eigensolver within the Cholesky-QR method for (14), we get a computed eigensystem which is the exact eigensystem of

$$L_s^{-1}(\tilde{\mathscr{A}} + \Delta\tilde{\mathscr{A}})L_s^{-T},$$

which for $g_1(n)$, a polynomial in n,

$$\|\Delta\tilde{\mathscr{A}}\|_\infty \leq g_1(n) u k(\tilde{\mathscr{B}}) \|\tilde{\mathscr{A}}\|_\infty. \tag{16}$$

Considering (12), (13) and (16), and denoting $\Delta A = \Delta A_1 + \tilde{U}^{-T}\Delta\tilde{\mathscr{A}}\,\tilde{U}^{-1}$, we get

$$L_s^{-1}(\tilde{\mathscr{A}} + \Delta\tilde{\mathscr{A}})L_s^{-T} = L_s^{-1}\tilde{U}^T(A + \Delta A)\tilde{U}L_s^{-T},$$

where

$$\|\Delta A\|_\infty \leq a_n u k(\tilde{U})\big(\|A\|_\infty + \mu_R\|\tilde{U}\|_2 + \mu_O\|B\|_2\big) + g_1(n)u k(\tilde{U})^2 k(\tilde{\mathscr{B}})\|A\|_\infty + O(u^2). \tag{17}$$

This shows that upper bound on the backward error of the computed eigenpairs by the preprocessed Cholesky-QR method given in Algorithm 1 includes $u k(\tilde{\mathscr{B}})\|A\|_\infty$, almost the same as those for the Cholesky-QR method [6].

Nevertheless, experimental results together with a study of scaled residual errors in section 5 show that, in finite precision arithmetic, the implementation of the studied algorithms which consists of some symmetrizing suggested in Section 5 are more stable than the Cholesky-QR method.

5 Numerical properties of the methods and some possible improvements

In the first part of this section, the scaled residual errors produced by the studied methods are used to explain more on their better stabilities (compared with the Cholesky-QR method). In the second part, the formula of normwise backward errors and the formula of mean of backward errors for computed eigenpairs will be presented. In the third part, we suggest two kinds of symmetrizing to be used in the appropriate implemented versions of the studied methods. The fourth part of this section consists of 5 different examples to assess and support the results and the suggestions of the previous sections. The experiments were performed in MATLAB 9, where the roundoff is $u = 2^{-53} \approx 1.1 \times 10^{-16}$.

From now on, by the Cholesky-QR method (CQR) we mean the `eig` function in MATLAB with the option `chol` and by the QZ method we mean the `eig` function in MATLAB with the option `qz`. We denote by OSDR1, RSSEP1, OSDR2, and RSSEP2 the implementations of Algorithm 1, Algorithm 2, Algorithm 3, and Algorithm 4, respectively.

In what follows, all the values on X-axis and Y-axis of the figures are in basis-10 logarithmic scale.

5.1 Scaled residual errors

In this section, considering the matrix pair of the problem (4), we are interested in the scaled residual errors related to the transformation matrix of the OSDR1 method when we compare them with those from the CQR method. The explanation and property of the scaled residual errors produced by the OSDR1 method and the RSSEP1 method are almost the same, so we do not discuss here those from the RSSEP1 method.

As it is seen in Algorithm 1, the OSDR1 method consists of two reductions. The first reduction is done by using Step 1 and Step 2 of the algorithm which provides the symmetric-diagonal matrix pair

$$(\mathscr{A},\mathscr{B}) \text{ from } (A,B). \tag{18}$$

The second reduction is done in the first part of the Step 3 of the algorithm. This second reduction is what actually the CQR method does on every SPD generalized eigenvalue problem $\mathscr{A}x = \lambda\mathscr{B}x$ with symmetric matrix $\mathscr{A}$ and SPD matrix $\mathscr{B}$, i. e., transferring the matrix pair

$$(\mathscr{A},\mathscr{B}) \text{ to the matrix pair } (\mathscr{C}_s,\mathscr{I}_n), \tag{19}$$

which corresponds to a symmetric standard eigenvalue problem $\mathscr{C}_s\hat{x} = \hat{\lambda}\mathscr{I}_n\hat{x}$ for n-by-n identity matrix $\mathscr{I}_n$. This analysis gives us more insight into the behavior of the OSDR1 methods and the CQR method.

It is simple to see that the overall transformation matrix in the OSDR1 method, including both reductions in (18) and (19), is

$$M_s = UL_s^{-T}, \tag{20}$$

where U is the unitary matrix computed in (5) and L_s is in general the lower triangular (but here is a diagonal) matrix obtained in (15). This means that the symmetric-diagonal matrix pair $(\mathscr{C}_s, \mathscr{I}_n)$ is congruent to the pair (A,B), where $\mathscr{C}_s = M_s^T AM_s$ and $\mathscr{I}_n = M_s^T BM_s$.

On the other hand, when we use the CQR method for solving the problem (4), the symmetric-diagonal matrix pair $(\mathscr{C}, \mathscr{I}_n) = (L^{-1}AL^{-T}, \mathscr{I}_n)$, for the lower triangular matrix L in the Cholesky factorization $B = LL^T$, becomes congruent to the pair (A,B), where for the transformation matrix

$$M = L^{-T}, \tag{21}$$

we have $\mathscr{C} = M^T AM$ and $\mathscr{I}_n = M^T BM$.

Suppose M_i for $i = 1,2$, be the transformation matrices in (20), and (21) respectively. Also, suppose (S_i, D_i) for $i = 1,2$ be the symmetric-diagonal matrix pairs produced using M_i for $i = 1,2$ respectively. We compute the following quantities:

- the scaled residual error related to A produced by each method,

$$\text{SREA}_i = \frac{\|M_i^T AM_i - S_i\|}{\|A\|\|M_i\|^2}, \quad i = 1,2. \tag{22}$$

- the scaled residual error related to B produced by each method,

$$\text{SREB}_i = \frac{\|M_i^T BM_i - D_i\|}{\|B\|\|M_i\|^2}, \quad i = 1,2. \tag{23}$$

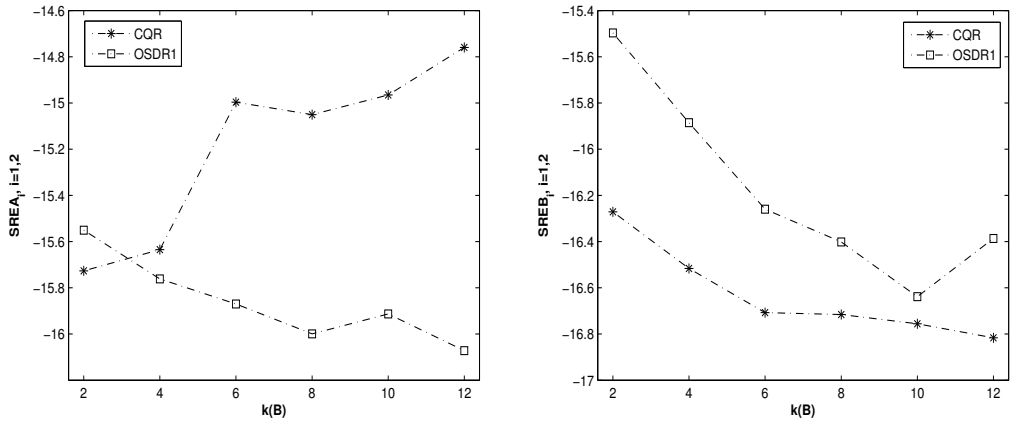
- the condition number of the transformation matrices, $k(M_i)$, $i = 1,2$.

We have computed the scaled residual errors (22) and (23) for all the problems in Examples 1-5 of Section 5.4. The explanation associated with the results are as follows:

- The global results for SREA_i, $i = 1,2$, for all considered problems, are almost the same. In these problems, the values of SREA_1 are smaller than the values of SREA_2 when $k(B) > 1\text{e}3$.
- The global results for SREB_i, $i = 1,2$, for all considered problems, are almost the same. The values of SREB_2 are smaller than the values of SREB_1 at almost all the considered problems.
- In all the problems, the condition numbers $k(M_1)$ and $k(M_2)$ are almost equal and they increase as $k(B)$ varies between the interval used in each example.

We know that, in practice, D_i for both $i = 1$ and $i = 2$, would be considered as the identity matrix. And this means that SREB_i, $i = 1,2$ are not as important as SREA_i, $i = 1,2$.

We detail, as representative, SREA_i, $i = 1,2$ and SREB_i, $i = 1,2$ for the problems in Example 4 of Section 5.2 with $n = 100$ and MODE $= 3$. Figure 1 (a) (resp. Figure 1 (b)) illustrates SREA_i, $i = 1,2$ (resp. SREB_i, $i = 1,2$) versus $k(B)$ for $n = 100$, respectively.



(a) SREA$_i$, $i = 1,2$ (b) SREB$_i$, $i = 1,2$

Figure 1: *Scaled residual errors versus* $k(B)$.

5.2 Backward errors of the computed eigenpairs

To evaluate relative normwise backward error of the computed eigenpair $(\tilde{x},\tilde{\lambda})$, we use the following explicit expression ([8], [10])

$$\eta(\tilde{x},\tilde{\lambda}) = \frac{\|\tilde{\mathbf{r}}\|}{(|\tilde{\lambda}|\ \|B\| + \|A\|)\|\tilde{x}\|},$$

where $\tilde{\mathbf{r}} = \tilde{\lambda}B\tilde{x} - A\tilde{x}$ is the residual of the pair $(\tilde{x},\tilde{\lambda})$. We denote the mean of backward errors of the computed finite eigenvalues as follows

$$\eta_{Mean} = Mean\{\eta(\tilde{x}_i,\tilde{\lambda}_i) :\ i = 1,\ldots,n\}.$$

We use the MATLAB function `isspd` from [18] to check the positive definiteness of the matrices. It is known that for accurate computation of small eigenvalues of (4), descending sorting (downward grading) of eigenvalues is effective [20]. This is because, the resulting matrix becomes graded, and the QR method is known to have good performance on graded matrices. The same is true for accurate computation of small eigenvalues of (9). See Examples 1-4 below. Also for accurate computation of large eigenvalues, ascending sorting (upward grading) of eigenvalues is effective [20]. Example 4 provides a case with one large eigenvalue that needs ascending sorting (not descending sorting) for eigenvalues.

To apply descending (resp. ascending) sorting of eigenvalues on the matrix $U\Sigma U^T$ in (5), we simply use the appropriate descending (resp. ascending) sorting on the diagonal entries of the matrix Σ. Then we do the same reordering (permutation) on the columns of the associated matrix U.

5.3 Symmetrizing

In finite precision computation, the symmetry of the matrix U^TAU in Algorithm 1 is not guaranteed. For this reason and as we prefer to solve the reduced problems by the CQR method, we suggest to use the

perturbation (i.e., symmetrizing)

$$\mathscr{A}(j,i) := \mathscr{A}(i,j), \quad \text{for } i = 2,\ldots,n, \quad \text{and } j = 1,\ldots,i-1, \tag{24}$$

on $\mathscr{A}$ (before starting the Step 3 of Algorithm 1) to make it exactly symmetric. We also suggest to use the following perturbation (i.e., symmetrizing) on the matrices $\mathscr{S}$,

$$\mathscr{S}(j,i) := \mathscr{S}(i,j), \quad \text{for } i = 2,\ldots,n, \quad \text{and } j = 1,\ldots,i-1, \tag{25}$$

before Step 3 of Algorithm 2.

The experiments, reported in Section 5.4, are usually the ones that involve the perturbations (24) and (25). But we have also checked with the ones that do not involve the above perturbations. The results are listed as follows.

1. When we apply the OSDR1 method (resp., the OSDR2 method) without the perturbation (24), its algorithm in Step 3 (resp., sub-Step 2-3) automatically switch to use the QZ method instead of the CQR method. This (at the cost of some performance) results in a preprocessed version of the QZ method with similar or better stability compared with those obtained directly from the QZ method.

2. When we use an implementation of the OSDR1 method (resp., the OSDR2 method) which does not use the perturbation (24) and does not switch to use the QZ method in Step 3 (resp., sub-Step 2-3), then we see almost the same stability property as those in the OSDR1 method (resp., the OSDR2 method) which uses (24).

   Anyway, for exploiting the structure of problem and getting a reasonable performance, one should use the perturbation (24) in the implementation of both the OSDR1 method and the OSDR2 method.

3. We also found that, in all considered cases, the perturbation (25) not only improves the performance of the RSSEP1 method and the RSSEP2 method but also is essential in the stability of these methods. See Example 3 for a representative of the cases that the perturbation (25) is not used.

5.4 Numerical Examples

We introduce five groups of examples to support the discussions of the prevoius sections and to show the improvements achieved using the reductions and the symmetrizings used in this work.

Example 1. We consider a group of examples from Matrix Market collection [19]. A brief description of the selected matrix pairs are given in Table 1. The structures of the matrices A and B are noted in the third and fourth columns of the Table 1, respectively. We use the abbreviation 'rspd' for the structure of a matrix to indicate that is a real symmetric positive definite matrix. Likewise, the abbreviation 'rspsd' for a matrix means that is real symmetric positive semi-definite. For the matrix pairs (A,B) in BCSST07 and BCSST19, both A's and B's are rspd. But, as in both of these problems $k(B) < k(A)$, we report on the solutions obtained by applying the considered methods for solving both $Ax = \lambda Bx$ and $Bx = \lambda Ax$. The CQR method cannot be used directly for BCSST01, BCSST04, and BCSST13, because, for these problems, A's are rspd and B's are rspsd. For the above reasons and because we want to provide

some direct test problems, we use the matrix pairs in BCSST01, BCSST04, BCSST07, BCSST13, and BCSST19 to make the following matrix pairs

$$(\hat{A},\hat{B}) := (B,A).$$

Now, our aim is to solve the problems BCSST07, and BCSST19, together with the problems

$$\hat{A}^{(i)}x = \lambda \hat{B}^{(i)}x, \tag{26}$$

for $i = 01,04,07,13$ and 19 to remind the original of $\hat{A}$ and $\hat{B}$. Here by BCSST07 and BCSST19, we mean these problems with their exact matrices A and B.

Table 2 displays the mean of backward errors of the eigenpairs resulted from solving the problems in (26) and the problems BCSST07 and BCSST19 by the OSDR1 method, the RSSEP1 method, the QZ method and the CQR method. Second column of Table 2 shows the condition number of B's or $\hat{B}$'s depend on the type and the name of the problems. Table 3 shows CPU-times for solving the problems introduced in Table 2.

Table 1: Matrix pairs from the Matrix Market collection.

Name	n	A	B	Brief description
BCSST01	48	rspd	rspsd	Small test problem
BCSST04	132	rspd	rspsd	Oil rig – not condensed
BCSST07	420	rspd	rspd	Medium test problem – consistent mass
BCSST13	2003	rspd	rspsd	Fluid flow generalized eigenvalues
BCSST19	817	rspd	rspd	Part of a suspension bridge

Example 2. We consider an example from [6]. Let

$$A = \begin{bmatrix} 1 & \alpha & 0 & \delta \\ \alpha & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ \delta & 0 & 0 & \varepsilon \end{bmatrix}, \text{ and } B = \text{diag}(\varepsilon,1,\varepsilon,1), \quad \alpha,\ \delta > 0, \quad 0 < \varepsilon < 1.$$

For $\alpha = 1$, $\delta =$ 1e-3 and a range of ε from 1e-10 to 1e-18, we have 9 problems. We solve the problems by the OSDR1 method, the RSSEP1 method, the QZ method, and the CQR method. The CQR method is unstable for almost all cases, while the QZ method displays almost the best stability and both the OSDR1 method and the RSSEP1 method obtain backward errors smaller than unit roundoff for each value of ε. Figure 2 (a) and Figure 2 (b) illustrate mean of backward errors versus $k(B)$ for the eigenvalues of minimal absolute ($\lambda_{Min} = min\{|\lambda_i|, i = 1,4\}$) and for the finite eigenvalues ($\lambda_{\text{finite}} = \{\lambda, |\lambda| < \infty\}$) of the problems, respectively. For the problems solved in this example CPU-times of the OSDR1 method and the RSSEP1 method are more than those of the CQR method and the QZ method.

Table 2: Means of backward errors of the computed eigenpairs $(\hat{x},\hat{\lambda})$ for the problems in (26), BCSST07 and BCSST19.

Problem	$k(B)$ or $k(\hat{B})$	$\eta_{Mean}^{\text{OSDR1}}$	$\eta_{Mean}^{\text{RSSEP1}}$	η_{Mean}^{QZ}	η_{Mean}^{CQR}
$\hat{A}^{(01)}x=\lambda\hat{B}^{(01)}x$	8.82e+5	4.54e-15	1.98e-15	9.45e-17	2.08e-12
$\hat{A}^{(04)}x=\lambda\hat{B}^{(04)}x$	2.29e+6	1.81e-14	2.20e-14	1.28e-16	2.32e-12
BCSST07	7.61e+3	2.49e-16	2.44e-16	3.27e-15	1.49e-16
$\hat{A}^{(07)}x=\lambda\hat{B}^{(07)}x$	7.57e+6	2.98e-16	2.96e-16	1.95e-15	1.38e-15
$\hat{A}^{(13)}x=\lambda\hat{B}^{(13)}x$	1.09e+10	1.96e-16	1.98e-16	1.62e-16	2.26e-14
BCSST19	2.34e+5	2.00e-17	2.01e-17	7.00e-16	1.77e-16
$\hat{A}^{(19)}x=\lambda\hat{B}^{(19)}x$	1.34e+11	1.22e-16	1.21e-16	2.31e-16	1.15e-13

Table 3: CPU- times for solving the problems introduced in Table 2.

Problem	t_{OSDR1}	t_{RSSEP1}	t_{QZ}	t_{CQR}
$\hat{A}^{(01)}x=\lambda\hat{B}^{(01)}x$	3.10e-3	2.70e-3	3.30e-3	1.50e-3
$\hat{A}^{(04)}x=\lambda\hat{B}^{(04)}x$	2.98e-2	2.48e-2	1.79e-2	4.90e-3
BCSST07	2.15e-1	2.46e-1	7.18e-1	1.30e-1
$\hat{A}^{(07)}x=\lambda\hat{B}^{(07)}x$	2.68e-1	2.43e-1	7.43e-1	1.48e-1
$\hat{A}^{(13)}x=\lambda\hat{B}^{(13)}x$	1.20e+1	1.28e+1	8.38e+1	4.60e+0
BCSST19	1.07e+0	9.61e-1	7.08e+0	6.67e-1
$\hat{A}^{(19)}x=\lambda\hat{B}^{(19)}x$	1.38e+0	1.24e+0	7.23e+0	6.38e-1

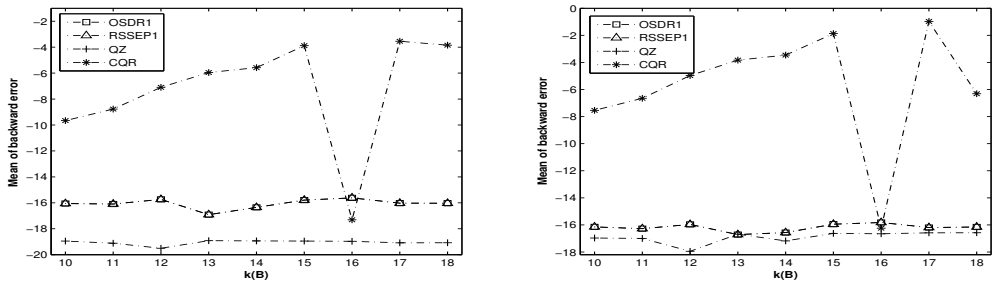
Example 3. This example was discussed in [13]. Let $A,B\in\mathbb{R}^{n\times n}$ and

$$A=\begin{bmatrix} 5 & -4 & 1 & & & & \\ -4 & 6 & -4 & 1 & & & \\ 1 & -4 & 6 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 6 & -4 & 1 \\ & & & 1 & -4 & 6 & -4 \\ & & & & 1 & -4 & 5 \end{bmatrix}, \quad \text{and} \quad B(i,j)=\frac{232792560}{i+j-1},$$

for $i,j=1,\ldots,n$. We consider 9 problems with 9 different sizes, $n=2,\ldots,10$. Condition number, $k(B)$, is an increasing function of n. Both set of matrices A and matrices B in these examples are SPD.

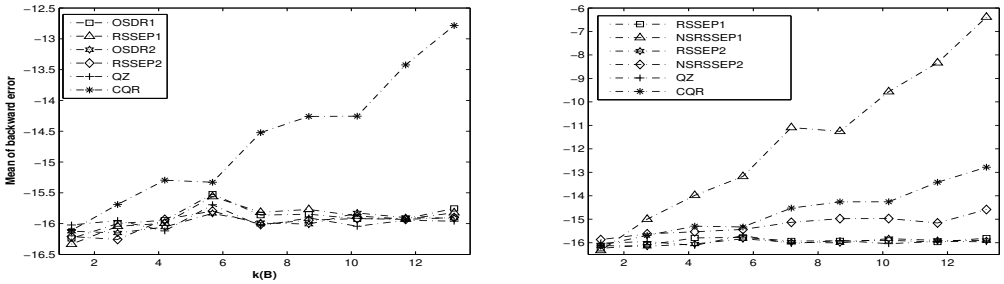
Figure 3 (a) displays mean of backward errors of the computed eigenpairs versus $k(B)$ when the problems are solved by the OSDR1 method, the RSSEP1 method, the OSDR2 method, the RSSEP2 method, the QZ method, and the CQR method. When $k(B)$ changes from its minimum to its maximum, the mean of backward errors of the solutions produced by the CQR method is increasing but the mean of backward errors of the solutions produced by the other methods stay near to unit roundoff.

Figure 3 (b) displays mean of backward errors of the computed eigenpairs versus $k(B)$ when the problems are solved by the RSSEP1 method, the NSRSSEP1 method, the RSSEP2 method, the NSRSSEP2



(a) $\eta_{Mean}^{\lambda_{Min}}$ (b) $\eta_{Mean}^{\lambda_{finite}}$

Figure 2: η_{Mean} versus $k(B)$.



(a) *Symmetrized using (24)* (b) *Not symmetrized*

Figure 3: η_{Mean} versus $k(B)$.

method, the QZ method, and the CQR method. Here NSRSSEP1 (resp. NSRSSEP2) stands for not symmetrized RSSEP1 (resp. RSSEP2) respectively, that is, a version of RSSEP1 (resp. RSSEP2) without the perturbation (25). As we can see, mean of backward errors of the eigenpairs computed by the NSRSSEP2 method are about one order larger than those computed by the RSSEP2 method. Also, the NSRSSEP1 method behaves unstably. CPU-times of the NSRSSEP1 method (resp. the NSRSSEP2 method) are more than CPU-times of the RSSEP1 method (resp. the RSSEP2 method).

This example (as representative of the other examples) shows the important roles of the perturbation (25) in the stability and in the performance of the RSSEP1 method and the RSSEP2 method. Also, CPU-times of the CQR method and the QZ method for this example are less than those of the other methods.

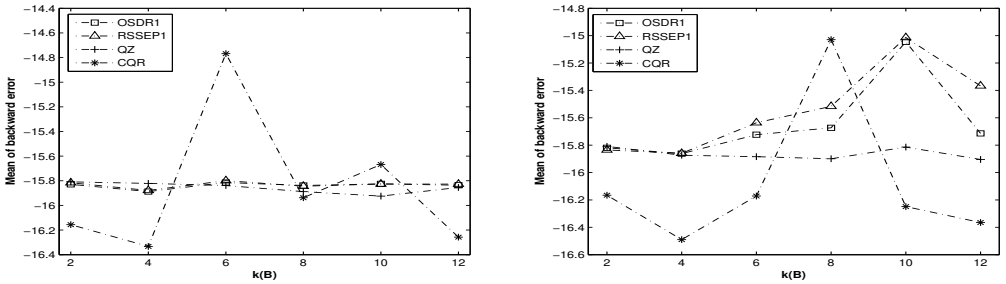
Example 4. Here we use some families of random matrices for A and B in (4). Specifically, we use MATLAB codes,

$$A = \text{randn}(n); \quad A = (A + A^T)/2; \tag{27}$$

for generating A, and

$$B = \text{gallery}('\text{randsvd}', n, \text{-1e}k, \text{MODE}); \tag{28}$$

for generating B where k belongs to a selection of $\{2,4,6,8,10,12\}$. Here the function 'randn' generates



(a) *Ascending sorting* (b) *Descending sorting*

Figure 4: η_{Mean} versus $k(B)$ for matrices B with MODE $= 1$ and $n = 100$.

a random matrix with entries normally distributed in $[-1,\ 1]$. Higham's test matrix,

$$\text{gallery}('\text{randsvd}', n, \text{-1e}k, \text{MODE}),$$

is mostly SPD with condition $\approx$ 1e+k [11]. MODE may be one of the following values:

1: one large singular value,
2: one small singular value,
3: geometrically distributed singular values,
4: arithmetically distributed singular values,
5: random singular values with uniformly distributed logarithm.
If omitted, MODE defaults to 3.

In our experiments, a vast number of examples, with sizes ranging from $n = 10$ to $n = 2000$, have been considered. In almost all cases, backward errors of eigenpairs computed by the OSDR1 method and the RSSEP1 method are much less sensitive to the variations of $k(B)$ than what backward errors of eigenpairs computed by the CQR method are. Also the QZ method shows the best stability behavior in almost all the cases.

Figures 4-7 show the results of solving problems (4) with A and B defined in (27) and (28) for different values of MODE and when $n = 100$. For solving the problems with MODE $= 1$, we separately use both ascending and descending sorting in Step 1 of the Algorithm 1. Figure 4 shows that for these problems (which have some large eigenvalues) ascending sorting results in better stability.

As expected, the CQR method at almost all cases needs the minimum CPU-time between the three considered methods. When $n \leq 20$, the QZ method is cheaper than the OSDR1 method and the RSSEP1 method. When $20 < n \leq 50$, the OSDR1 method, the RSSEP1 method and the QZ method show almost the same performance. When $50 < n \leq 100$, the performance of the OSDR1 method and the RSSEP1 method are better than the QZ method at almost all cases. When $n \geq 100$, the CPU-times of the OSDR1 method and the RSSEP1 method are much closer to the CPU-times of the CQR method than those of the QZ method. Table 4 illustrate the overall of the relationship between CPU-times consumed by the methods and the size of the problems. Figure 8 (a) (resp., Figure 8 (b)) illustrates CPU-times of the methods for solving the problems with $n = 100$ (resp., with $n = 500$) where MODE $= 3$.

Example 5. In this example, for each value of n in $\{50, 100, 150, 200, 250, 300\}$ (as size of the problem), we consider 64 different problems. These problems are constituted by 8 different matrices A and 8

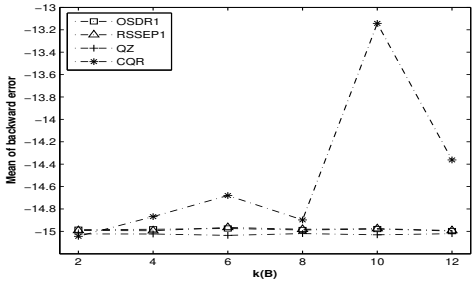
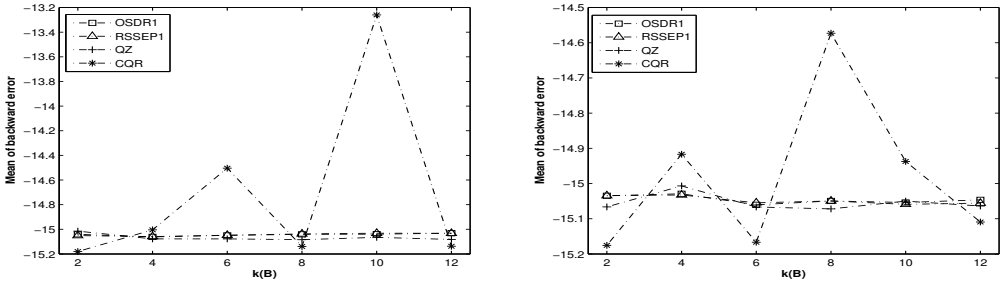
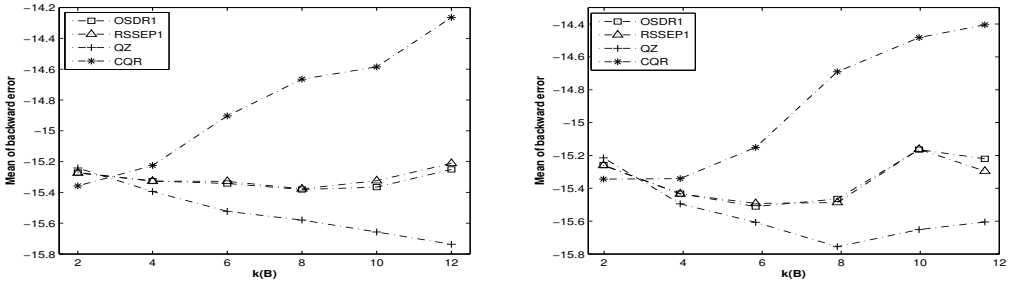


Figure 5: η_{Mean} *versus* $k(B)$ *for matrices B with MODE* $= 2$ *and* $n = 100$.



(a) *First representative* (b) *Second representative*

Figure 6: η_{Mean} versus $k(B)$ for matrices B with MODE $= 4$ and $n = 100$.

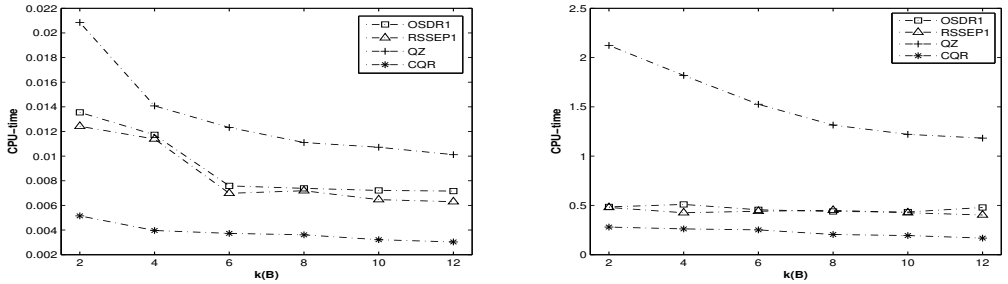


(a) *MODE* $= 3$ (b) *MODE* $= 5$

Figure 7: η_{Mean} versus $k(B)$ for matrices B with MODE $= 3$ and or MODE $= 5$ for $n = 100$.

Table 4: CPU-time of the methods versus n.

Problem size (n)	$n \leq 20$	$20 < n < 50$	$50 \leq n < 100$	$100 \leq n$
Smallest CPU-time	t_{CQR}	t_{CQR}	t_{CQR}	t_{CQR}
2nd smallest CPU-time	t_{QZ}	$t_{\text{QZ}} \approx t_{\text{OSDR1}}$ ($\approx t_{\text{RSSEP1}}$)	t_{OSDR1} or t_{RSSEP1} (at almost all cases)	t_{OSDR1} (or t_{RSSEP1})



(a) *Representative for* $n = 100$ (b) *Representative for* $n = 500$

Figure 8: CPU-time versus $k(B)$ for matrices B with sizes $n = 100$ and $n = 500$ when MODE $= 3$.

Table 5: N^{OSDR1}, N^{RSSEP1}, N^{OSDR2}, N^{RSSEP2}, N^{CQR}, and N^{QZ} versus n.

Problem size (n)	N^{OSDR1}	N^{RSSEP1}	N^{OSDR2}	N^{RSSEP2}	N^{CQR}	N^{QZ}
50	12	8	5	9	5	25
100	11	10	14	14	7	8
150	16	10	17	11	8	2
200	12	14	13	17	8	0
250	10	11	18	17	8	0
300	13	10	18	15	8	0

different matrices B both generated by (28) when MODE $= 3$. The condition numbers of the matrices ranges from about 1e+1 to about 1e+15.

We solve each problem with fixed A and B (each one of the 64 problems) by the OSDR1 method, the RSSEP1 method, the OSDR2 method, the RSSEP2 method, the CQR method, and the QZ method. Then, we compute

$$\eta_{Mean}^{Min} = \min\{\eta_{Mean}^{\text{OSDR1}}, \eta_{Mean}^{\text{RSSEP1}}, \eta_{Mean}^{\text{OSDR2}}, \eta_{Mean}^{\text{RSSEP2}}, \eta_{Mean}^{\text{CQR}}, \eta_{Mean}^{\text{QZ}}\}. \tag{29}$$

For each group of the problems with a specific size, n, and a range of condition numbers from about 1e+1 to about 1e+15 (for As and Bs), the number of times that the $\eta_{Mean}^{\text{methodname}}$ in (29) gives η_{Mean}^{Min} is denoted by $N^{\text{methodname}}$. Table 5 reports on N^{OSDR1}, N^{RSSEP1}, N^{OSDR2}, N^{RSSEP2}, N^{CQR}, and N^{QZ} versus n. When n changes from 50 to 300, N^{CQR} does not experience a meaningful change and N^{QZ} decreases from 25 to 0. At the same time, N^{OSDR1} and N^{RSSEP1} (resp. N^{OSDR2} and N^{RSSEP2}) show a competitive behaviors respectively.

For the problems considered in this example, we have the following results:

- when $k(B) \leq$ 1e3, usually η_{Mean}^{CQR} outperforms those of the other compared methods.
- when $k(B) >$ 1e3, and $k(A) > k(B)$, either $\eta_{Mean}^{\text{OSDR1}}$ or $\eta_{Mean}^{\text{RSSEP1}}$ provides η_{Mean}^{Min}.
- when $k(B) >$ 1e3, and $k(A) \leq k(B)$, either $\eta_{Mean}^{\text{OSDR2}}$ or $\eta_{Mean}^{\text{RSSEP2}}$ provides η_{Mean}^{Min}.

It should be noted that, the problems in Example 4 (with respect to the condition numbers of the matrices) are almost particular cases of the problems in Example 5. The problems of Example 4 are very close to the cases in Example 5 where $1e1 \leq k(A) \leq 1e3$.

6 Conclusions

We have discussed and analyzed two main versions of the Schur-QR method (i.e., the RSSEP1 method and the RSSEP2 method) for solving SPDGEVP. The first version (resp., the second version) of the Schur-QR method has been organized for the problems with symmetric matrix A (resp., B) and SPD matrix B (resp., A). Each one of these versions, at the first phase, enjoys an appropriate orthogonal symmetric-diagonal reduction. We used this phase of each version of the Schur-QR method to introduce an idea for making the preprocessed CQR method, i.e., the OSDR1 method (resp., the OSDR2 method), for the problems whose matrices B (resp., A) are SPD. The possibility of making the preprocessed QZ methods was examined in some of the experiments.

The orthogonal symmetric-diagonal reduction phase together with appropriate upward or downward eigenvalue sorting and two specified symmetrizings improve the numerical stability of the methods. Based on the experimental results, the backward errors of the eigenpairs computed by the considered four methods (i.e., shortly, OSDR1, RSSEP1, OSDR2, and RSSEP2) receive negligible effects from the changes in $k(B)$ (resp. $k(A)$) for the problems in (4) (resp. in (9)). For the problems which can be solved by either of these methods and the CQR method, usually OSDR1, RSSEP1, OSDR2, and RSSEP2 act more stable than the CQR method.

Studying of the scaled residual errors, and especially SREA_i, $i = 1,2$, uncovers one of the main reasons responsible for the superiority seen in the stability of the mentioned methods compared with those of the CQR method. The modifications suggested in (24) and (25) improve the performances of the discussed methods. The modification (25) have also essential role in the statbilty of the RSSEP1 method and the RSSEP2 method.

Our experiments support the performance analysis in Section 3 and indicate that the CQR method is the cheapest method between the compared methods in this study. For $n \leq 20$, usually CPU-times of four methods are larger than those of the QZ method and the CQR method. For $20 < n < 50$, the considered four methods become faster and need comparable CPU-times. For the problems with sizes $50 \leq n < 100$, (at almost all cases), CPU-times necessary by the considered four methods are less than those of the QZ method and larger than those of the CQR method. In fact, for the problems with sizes $n > 50$, the growth rate of the CPU-times of the QZ method is much more than those of the considered four methods as n increases. Therefore, for the problems with sizes $n > 100$, CPU-times of the considered four methods stay much closer to the CPU-times of the CQR method than those of the QZ method.

The main contributions of this paper includes

1. revisiting and discussing two versions of the Schur-QR method for SPDGEVPs,
2. suggesting and studying the preprocessed versions of the CQR method and the QZ method,
3. proving that the Schur-QR method is theoretically equivalent to the preprocessed Cholesky-QR method,
4. analyzing the performance of the studied methods,
5. analyzing the stability of the studied methods by bounding backward errors of the computed eigenpairs and also by the scaled residual errors,

6. suggesting and applying some symmetrizings which improve the performance and stability of the considered methods,

7. providing some detailed numerical experiments with considerable comparisons and visualizations,

8. introducing a family of problems (in Example 5) where at most of the cases one of the two versions of the Schur-QR method or the two preprocessed versions of the CQR method behave more stable than both of the CQR method and the QZ method.

Optimized and efficient implementation of the proposed Algorithms is a direction for future work.

Acknowledgments

The author would like to thank the editor and anonymous referees. Special thanks goes to Professor Siegfried M. Rump who reviewed the earlier version of this manuscript and provided valuable suggestions.

References

[1] M. Ahuesa, A. Largillier, F.D. D´Almeida, P.B. Vasconcelos, *Spectral refinement on quasi-diagonal matrices*, Linear Algebra Appl. **401** (2005) 109–117.

[2] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[3] S. Chandrasekara, I.C.F. Ipsen, *Backward errors for eigenvalue and singular value decompositions*, Numer. Math. **68** (1994) 215–223.

[4] S. Chandrasekara, *An efficient and stable algorithm for the symmetric-definite generalized eigenvalue problem*, SIAM J. Matrix Anal. Appl. **21** (2000) 1202–1228.

[5] B.N. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole, Pacific Grove, CA, 1995.

[6] 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** (2001) 472–493.

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

[8] V. Frayssé, V. Toumazou, *A note on the normwise perturbation theory for the regular generalized eigenproblem*, Numer. Linear Algebra Appl. **5** (1998) 1–10.

[9] G.H. Golub, C.F. Van Loan, *Matrix Computations* (3rd edition). The Johns Hopkins University Press, Baltimore, 1996.

[10] D.J. Higham, N.J. Higham, *Structured backward error and condition of generalized eigenvalue problems*, SIAM J. Matrix Anal. Appl. **20** (1998) 493–512.

[11] N.J. Higham, *Accuracy and Stability of Numerical Algorithms* (Second edition). SIAM, Philadelphia, 2002.

[12] P. Lancaster, *Linearization of regular matrix polynomials*, Electron. J. Linear Algebra **17** (2008) 21–27.

[13] S. Miyajima, T. Ogita, S.M. Rump, S. Oishi, *Fast verification for all eigenpairs in symmetric positive definite generalized eigenvalue problems*, Reliab. Comput. **14** (2010) 24–45.

[14] C.B. Moler, G.W. Stewart, *An algorithm for generalized matrix eigenvalue problems*, SIAM J. Numer. Anal. **10** (1973) 241–256.

[15] T. Ogita, K. Aishima, *Iterative refinement for symmetric eigenvalue decomposition*, Japan J. Indust. Appl. Math. **35** (2018) 1007–1035.

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

[17] G. Peters, J.H. Wilkinson, *$Ax = \lambda Bx$ and the generalized eigenproblem*, SIAM J. Numer. Anal. **7** (1970) 479–492.

[18] S.M. Rump, *Verification of positive definiteness*, BIT **46** (2006) 433–452.

[19] R. Boisvert, R. Pozo, K. Remington, R. Barrett, J. Dongarra, *Matrix Market*, NIST.

[20] G. W. Stewart, *Matrix Algorithms, Volume II: Eigensystems*, SIAM, Philadelphia, 2001.

[21] F. Tisseur, *Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems*, SIAM J. Matrix Anal. Appl. **22** (2001) 1038–1057.

[22] F. Tisseur, *Tridiagonal-Diagonal reduction of symmetric indefinite pairs*, SIAM J. Matrix Anal. Appl. **26** (2004) 215–232.

[23] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, UK, 1965.