Adv Comput Math (2016) 42:1103–1128
DOI 10.1007/s10444-016-9455-2

# A block Chebyshev-Davidson method for linear response eigenvalue problems

**Zhongming Teng[1] · Yunkai Zhou[2] · Ren-Cang Li[3]**

Received: 5 February 2015 / Accepted: 26 January 2016 /
Published online: 25 February 2016


**Abstract** We present a Chebyshev-Davidson method to compute a few smallest positive eigenvalues and corresponding eigenvectors of linear response eigenvalue problems. The method is applicable to more general linear response eigenvalue problems where some purely imaginary eigenvalues may exist. For the Chebyshev filter, a tight upper bound is obtained by a computable bound estimator that is provably

Communicated by: Carlos Garcia - Cervera

Zhongming Teng was supported in part by China Scholarship Council and Natural Science Foundation of Fujian province No. 2015J01580. Part of this work was done while this author was a visiting student at Department of Mathematics, University of Texas at Arlington.

Yunkai Zhou was supported in part by the National Science Foundation grants DMS-1228271 and DMS-1522587.

Ren-Cang Li was supported in part by the National Science Foundation grants DMS-1317330 and CCF-1527104, NSFC grant 11428104.

✉ Yunkai Zhou
yzhou@smu.edu

Zhongming Teng
peter979@163.com

Ren-Cang Li
rcli@uta.edu

[1] College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou, 350002, People's Republic of China

[2] Department of Mathematics, Southern Methodist University, Dallas, TX 75275, USA

[3] Department of Mathematics, University of Texas at Arlington, P. O. Box 19408, Arlington, TX 76019, USA

correct under a reasonable condition. When the condition fails, the estimated upper bound may not be a true one. To overcome that, we develop an adaptive strategy for updating the estimated upper bound to guarantee the effectiveness of our new Chebyshev-Davidson method. We also obtain an estimate of the rate of convergence for the Ritz values by our algorithm. Finally, we present numerical results to demonstrate the performance of the proposed Chebyshev-Davidson method.

**Keywords** Eigenvalue/eigenvector · Chebyshev polynomial · Davidson type method · Convergence rate · Linear response · Upper bound estimator

**Mathematics Subject Classifications (2010)** 65F15 · 15A18

## 1 Introduction

In computational quantum chemistry and physics, the so-called *random phase approximation* (RPA) describes the excitation states (energies) of physical systems in the study of collective motion of many-particle systems [2, 25, 26]. It has important applications in silicon and other nanoscale materials, analysis of interstellar clouds [2, 3], polarizabilities [20], and finding the electronic excitation spectrum of a quantum many-fermion system [27]. One important question in RPA is to compute a few eigenpairs associated with the smallest *positive* eigenvalues of the following eigenvalue problem:

$$\mathscr{H}w := \begin{bmatrix} A & B \\ -B & -A \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \end{bmatrix}, \tag{1.1}$$

where $A, B \in \mathbb{R}^{n\times n}$ are both symmetric matrices and $\begin{bmatrix} A & B \\ B & A \end{bmatrix}$ is positive definite. Through a similarity transformation, this eigenvalue problem can be equivalently transformed into [2, 3, 19]

$$Hz := \begin{bmatrix} 0 & K \\ M & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \lambda \begin{bmatrix} y \\ x \end{bmatrix}, \tag{1.2}$$

where $K = A - B$ and $M = A + B$. The eigenvalue problem Eq. (1.2) was still referred to as the *linear response eigenvalue problem* (LREP) [2, 19, 27] and will be so in this paper, too.

The condition imposed upon $A$ and $B$ in Eq. (1.1) implies that both $K$ and $M$ are real symmetric and positive definite [2]. But there are cases where one of them may be indefinite [17]. Throughout this paper, we consider the more general case:

$$\boxed{K \text{ and } M \text{ are Hermitian, one of them is positive definite, and the other may be indefinite.}} \tag{1.3}$$

For this general case, the interested quantities are the first few eigenvalues whose squares are the smallest few, together with their corresponding eigenvectors.

For simplicity, in the rest of this paper, we assume that $M$ is positive definite and $K$ is Hermitian but may be indefinite. Doing so loses no generality because otherwise we simply interchange the roles of $K$ and $M$.

From Eq. (1.2), we have $Kx = \lambda y$ and $My = \lambda x$ and they together produce

$$KMy = \lambda^2 y, \tag{1.4a}$$
$$MKx = \lambda^2 x. \tag{1.4b}$$

The three eigenvalue problems: (1.2) of $H$, (1.4a) of $KM$, and (1.4b) of $MK$, are theoretically equivalent. That is, if any one of them is solved, the solutions to theother two can be constructed from the solved one with little effort [2, Theorem 2.1].

Researches on solving Eq. (1.2), or equivalently Eqs. (1.4a) and (1.4b), have been very active, see [3, section 1] and references therein. The goal of this paper is to extend the Chebyshev-Davidson method and its block version proposed in [30, 33] for symmetric/Hermitian eigenvalue problems. The new method is capable of tackling the more general case (1.3) that may have purely imaginary eigenvalues whereas previous methods as surveyed in [3, section 1] cannot.

The rest of this paper is organized as follows. In Section 2, we present our algorithm and the convergence rate estimate based on an upper bound estimator. The computable upper bound estimator for $\lambda_{\max}(KM)$ and the adaptive strategy are detailed in Section 3. Numerical results of our algorithm are presented in Section 4. Finally, some conclusions are drawn in Section 5.

Throughout this paper, $\mathbb{K}^{n\times m}$ is the set of all $n \times m$ matrices with entries in $\mathbb{K}$, where $\mathbb{K}$ is $\mathbb{C}$ (the set of complex numbers) or $\mathbb{R}$ (the set of real numbers), $\mathbb{K}^n = \mathbb{K}^{n\times 1}$, and $\mathbb{K} = \mathbb{K}^1$. $I_n$ (or simply $I$ if its dimension is clear from the context) is the $n \times n$ identity matrix, and $e_j$ is its $j$th column. MATLAB-like convention is adopted to access the entries of vectors and matrices. For a matrix $X$, $X_{(:,i:j)}$ denotes the submatrix consisting the $i$-th to the $j$-th columns of $X$, $\mathrm{diag}(X)$ denotes the column vector of the diagonal entries of $X$, and $\mathrm{eig}(X)$ denotes the spectrum of $X$. If $X$ is nonsingular, $\kappa_2(X)$ denotes the spectral condition number of $X$. The superscript "$\cdot^{\mathrm{H}}$" takes conjugate transpose while "$\cdot^{\mathrm{T}}$" takes transpose only. The number $\bar{\alpha}$ is the conjugate of the scalar $\alpha$.

## 2 Chebyshev filter for LREP

### 2.1 Chebyshev filter

Chebyshev polynomials play important roles in theoretical analysis of numerical algorithms as well as in algorithmic design [21]. They are also useful in practice, as a means of accelerating single vector iterations or projection processes. In [30, 33], the polynomials are used to magnify the components of the desired eigenvectors while suppress those of the undesired ones.

The $m$-th Chebyshev polynomial of the 1st kind is

$$\mathscr{T}_m(t) = \begin{cases} \cos(m \arccos(t)), & \text{for } -1 \le t \le 1, \\ \cosh(m \operatorname{arccosh}(t)), & \text{for } |t| \ge 1. \end{cases} \tag{2.1}$$

It frequently shows up in numerical analysis and computations because of its numerous nice properties; for example, $|\mathscr{T}_m(t)| \leq 1$ for $|t| \leq 1$ and $|\mathscr{T}_m(t)|$ grows extremely fast[1] for $|t| > 1$. Also important is the associated three-term recurrence relation: $\mathscr{T}_0(t) = 1$, $\mathscr{T}_1(t) = t$, and for $m \geq 1$

$$\mathscr{T}_{m+1}(t) = 2t\,\mathscr{T}_m(t) - \mathscr{T}_{m-1}(t), \tag{2.2}$$

which is responsible for the numerical efficiency in its numerous applications.

For an $n \times n$ Hermitian matrix $A$, suppose $\mathrm{eig}(A) \subset [\alpha_0, \beta]$ and we are interested in the eigenvalues lying in $[\alpha_0, \alpha]$, where $\alpha_0 < \alpha < \beta$. The basic idea of the Chebyshev filtering approach is to use Chebyshev polynomials to magnify the components of an approximate eigenvector $u$ in the directions of the wanted eigenvectors and at the same time suppress those in the directions of the unwanted eigenvectors. To this end, we first transform $[\alpha, \beta]$ onto $[-1, 1]$ by the following affine mapping:

$$t \in [\alpha, \beta] \quad \rightarrow \quad \phi(t) = \frac{2t - (\alpha + \beta)}{\beta - \alpha} \in [-1, 1]. \tag{2.3}$$

Then $|\mathscr{T}_m(\phi(t))| \leq 1$ for $t \in [\alpha, \beta]$ and $|\mathscr{T}_m(\phi(t))|$ grows extremely fast for $t \notin [\alpha, \beta]$. For this reason, $\mathscr{T}_m(\phi(A))u$ magnify the components of $u$ in the directions of the eigenvectors associated with eigenvalues in $[\alpha_0, \alpha)$ and at the same time suppress those in the directions of the eigenvectors associated with eigenvalues in $[\alpha, \beta]$. The effect is more dramatic as $m$ increases. The three-term recurrence relation (2.2) yields

$$\mathscr{T}_{m+1}(\phi(A))u = 2\phi(A) \cdot \mathscr{T}_m(\phi(A))u - \mathscr{T}_{m-1}(\phi(A))u, \tag{2.4}$$

making it very efficient to compute $\mathscr{T}_m(\phi(A))u$.

What we just described is the basis of the Chebyshev filter in [30] and [33] for the large scale symmetric/Hermitian eigenvalue problem. But it is not difficult to notice that the idea can be applied to diagonalize matrices with real spectra. The following lemma [2] establishes some theoretical results, which show that Eq. (1.2) may be transformed into eigenvalue problems with real spectra. Therefore we can achieve acceleration by constructing appropriate Chebyshev filters.

**Lemma 2.1** *The following statements hold true for any matrices $M$ and $K$ in $\mathbb{C}^{n\times n}$, where $M$ is positive definite and $K$ is Hermitian.*

(a) *There exists a nonsingular $Y = [y_1, y_2, \ldots, y_n] \in \mathbb{C}^{n\times n}$ such that*

$$K = Y\Lambda^2 Y^{\mathrm{H}}, \quad M = XX^{\mathrm{H}}, \tag{2.5}$$

*where $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, $\lambda_1^2 \leq \lambda_2^2 \leq \cdots \leq \lambda_n^2$ and $X = Y^{-\mathrm{H}} = [x_1, x_2, \ldots, x_n]$.*

(b) $\left[\lambda_i y_i^{\mathrm{H}}, x_i^{\mathrm{H}}\right]^{\mathrm{H}}$ *is an eigenvector corresponding to $\lambda_i$ of the matrix $H$,*

$$H \begin{bmatrix} \lambda_i y_i \\ x_i \end{bmatrix} = \lambda_i \begin{bmatrix} \lambda_i y_i \\ x_i \end{bmatrix}. \tag{2.6}$$

[1] In fact, a result due to Chebyshev himself says that if $p(t)$ is a polynomial of degree no greater than $m$ and $|p(t)| \leq 1$ for $-1 \leq t \leq 1$, then $|p(t)| \leq |\mathscr{T}_m(t)|$ for any $t$ outside $[-1, 1]$ [6, p.65].

(c) *Let* $(\lambda_i, z_i)$ $(i = 1, 2)$ *be two eigenpairs of* $H$, *and partition* $z_i = \left[y_i^{\mathrm{H}}, x_i^{\mathrm{H}}\right]^{\mathrm{H}}$. *Then,*

(i) *if* $\lambda_1 \neq \bar{\lambda}_2$, *then* $y_1^{\mathrm{H}} x_2 + y_2^{\mathrm{H}} x_1 = 0$.
(ii) *if* $\lambda_1 \neq \pm\lambda_2$, *then* $y_1^{\mathrm{H}} x_2 = y_2^{\mathrm{H}} x_1 = 0$.

(d) *Both* $KM$ *and* $MK$ *are diagonalizable. In fact,*

$$KM = Y\Lambda^2 Y^{-1}, \quad MK = X\Lambda^2 X^{-1}.$$

Here $M$ is assumed to be positive definite. If in addition $K$ is positive semi-definite or definite, then $\lambda_1^2 \geq 0$ and thus we can make all $\lambda_i \geq 0$. In this case the eigenvalues of $H$ are all real $\pm\lambda_i$,

$$-\lambda_n \leq \cdots \leq -\lambda_2 \leq -\lambda_1 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n. \tag{2.7}$$

When $K$ is indefinite, the first few $\lambda_i^2$ may be negative. To define $\lambda_i$ unique, we take $\lambda_i$ to be the unique square root of $\lambda_i^2$ with nonnegative imaginary part. Lemma 2.1 was proved for $K$ being positive definite [2], but the proof carries over for indefinite $K$ with only minor changes.

Often in LREP (1.2), both $K$ and $M$ are definite and the first $k$ smallest *positive* eigenvalues $\lambda_i$ for $i = 1, 2, \ldots, k$ are of interest. They lie in the middle of the spectrum of $H$, a region to which a straightforward application of the scaled $\mathscr{T}_m(\phi(t))$ often does not work well. This causes problems for applying Chebyshev filters. Fortunately, we can apply the squaring technique, noticing that the square of these eigenvalues are all real and lie in the left end of the spectrum of $KM$ or $MK$. Moreover, $KM$ and $MK$ are diagonalizable by Lemma 2.1(d). In what follows, we restrict discussions on constructing Chebyshev filters for $KM$. Any development after minor modifications will work for $MK$ as well.

Let $v \in \mathbb{C}^n$. It can be expressed in the basis $\{y_i\}_{i=1}^n$ as

$$v = \eta_1 y_1 + \eta_2 y_2 + \cdots + \eta_n y_n.$$

Suppose that we have three scalars $\alpha_0 < \alpha < \beta$ such that

$$\lambda_i^2 \in [\alpha_0, \beta] \text{ for } i = 1, 2, \ldots, n, \text{ with } \lambda_k^2 < \alpha \text{ and } \lambda_{k+1}^2 \geq \alpha. \tag{2.8}$$

In practice, initially $\alpha_0$, $\alpha$, and $\beta$ are likely only rough estimates and may violate (2.8), but they can be refined adaptively to satisfy (2.8) in the end. To amplify the components of $v$ in $y_i$ for $i = 1, 2, \ldots, k$ and to suppress those in $y_i$ for $i \geq k+1$, we compute

$$\mathscr{T}_m(\phi(KM))v = \eta_1 \mathscr{T}_m(\phi(\lambda_1^2))y_1 + \eta_2 \mathscr{T}_m(\phi(\lambda_2^2))y_2 + \cdots + \eta_n \mathscr{T}_m(\phi(\lambda_n^2))y_n.$$

Making $|\mathscr{T}_m(\phi(\lambda_i^2))| \leq 1$ for $i > k+1$ while $|\mathscr{T}_m(\phi(\lambda_i^2))|$ for $i = 1, 2, \ldots, k$ are (potentially much) larger than 1, $\mathscr{T}_m(\phi(KM))$ serves the purpose of filtering out the unwanted eigen-directions in $v$.

The same filtering idea works when $v$ is replaced by a "block of vectors" $V \in \mathbb{C}^{n\times\ell}$ with a full column rank. Let

$$p_m(t) = \frac{\mathscr{T}_m(\phi(t))}{\mathscr{T}_m(\phi(\alpha_0))}, \quad \sigma_i = \frac{\mathscr{T}_{i-1}(\phi(\alpha_0))}{\mathscr{T}_i(\phi(\alpha_0))}.$$

It can be verified that

$$\begin{aligned}\sigma_1 &= \omega/(\alpha_0-\gamma),\\ \sigma_i &= \left(\frac{2}{\sigma_1}-\sigma_{i-1}\right)^{-1} \quad \text{for } i=2,3,\ldots,\end{aligned}$$

where $\gamma = \frac{\alpha+\beta}{2}$ and $\omega = \frac{\beta-\alpha}{2}$. The scaled Chebyshev polynomials are defined by

$$\begin{aligned}p_0(t) = 1, \quad p_1(t) = (t-\gamma)\sigma_1/\omega,\\ p_i(t) = 2\sigma_i\tfrac{t-\gamma}{\omega}\,p_{i-1}(t) - \sigma_{i-1}\sigma_i\,p_{i-2}(t) \quad \text{for } i=2,3,\ldots.\end{aligned}$$

**Algorithm 2.1** $V_{\text{new}} = \texttt{Chebyshev\_filter}(V, m, \alpha, \beta, \alpha_0)$

1 $\omega = (\beta-\alpha)/2$, $\gamma = (\alpha+\beta)/2$, $\sigma_1 = \omega/(\alpha_0-\gamma)$, $\tau = 2/\sigma_1$.
2 $V = (K(MV_{\text{old}}) - \gamma V_{\text{old}})\sigma_1/\omega$.
3 **for** $i = 2 : m$ **do**
  $\sigma_i = 1/(\tau-\sigma_{i-1})$, $V_{\text{new}} = 2(K(MV) - \gamma V)\sigma_i/\omega - \sigma_{i-1}\sigma_i V_{\text{old}}$.
  $V_{\text{old}} = V$, $V = V_{\text{new}}$.
**end**

Algorithm 2.1 displays the pseudo-code that implements the block version of the scaled Chebyshev filter for $KM$. It computes $V_{\text{new}} = p_m(KM)V$. The parameter $\alpha_0$ in Algorithm 2.1 is mainly used for the purpose of scaling, and hence a crude estimate $\alpha_0$ is usually sufficient. The lower bound $\alpha$ of unwanted eigenvalue interval can be estimated and refined during an iterative process of a subspace method with minimal or no extra computational cost. The upper bound $\beta$ of $\text{eig}(KM)$ needs to be supplied at the beginning. Though $\beta$ can be bounded by $\|K\|\|M\|$ for any consistent matrix norm, it is often too large to make the filtering effective. In Section 3, we present a (much) tighter bound by constructing an estimator using the Lanczos biorthogonalization procedure.

### 2.2 The main algorithm

Let $V_{\text{new}} = p_m(KM)V$ be the output of the Chebyshev filter in Algorithm 2.1. Naturally the filtered columns $V_{\text{new}}$ span an approximate invariant subspace of $KM$. Due to the equivalence of the eigenvalue problems (1.2) and (1.4a), the column space of $V_{\text{new}}$ approximates the subspace spanned by the $y$-components of the eigenvectors of $H$ associated with the designed eigenvalues $\lambda_i$, $i = 1, 2, \ldots, k$. To obtain the subspace spanned by the $y$-components, we can apply a Chebyshev filter on $MK$ to get $U_{\text{new}} = p_m(MK)U$, the filtered columns in $U_{\text{new}}$ span an approximate invariant subspace of $MK$, which also correspond to $\lambda_i$, $i = 1, 2, \ldots, k$. This works, but it introduces unnecessary cost. We can utilize the following lemma for a more cost effective method.

**Lemma 2.2** *Let $\mathcal{Z}$ be an invariant subspace of $H$ and let $Z = \begin{bmatrix} V \\ U \end{bmatrix}$ be the basis matrix of $\mathcal{Z}$ with both $V$ and $U$ having $n$ rows, then* $\text{span}(MV) = \text{span}(U)$.

Lemma 2.2 follows from the structure of $H$. It suggests that if the columns of $V$ span the $y$-component of an approximate invariant subspace of $H$, then the columns of $U := MV$ should span the $x$-component of the same approximate invariant subspace. There are at least two advantages for computing $U$ this way. First, it is economical because $MV$ has to be computed in the later projection phase, and second, $W := V^{\mathrm{H}}U = V^{\mathrm{H}}MV$ is guaranteed to be symmetric positive definite (SPD). This SPD property is important in the later projection phase.

Now that we have $\{\mathrm{span}(V), \mathrm{span}(U)\}$ as a pair of approximate deflating subspaces of $\{K, M\}$, by [2], the best approximations to the first $\lambda_i$ for $i = 1, 2, \ldots, k$ within the pair of approximate deflating subspaces are the eigenvalues of [2]

$$H_{\mathrm{SR}} = \begin{bmatrix} 0 & W_1^{-\mathrm{H}}U^{\mathrm{H}}KUW_1^{-1} \\ W_2^{-\mathrm{H}}V^{\mathrm{H}}MVW_2^{-1} & 0 \end{bmatrix}, \tag{2.9}$$

where $W_1$ and $W_2$ are two nonsingular factors of $W$ as in $W = W_1^{\mathrm{H}}W_2$. This reduced $H_{\mathrm{SR}}$ is closely related to the so-called *Rayleigh quotient pair* introduced in [29]. By [2, Theorem 2.9], we know that how $W$ is factorized does not affect eventual approximations to the eigenvalues and the corresponding eigenvectors of $H$. Therefore we may take $W_1 = W_2 = R$, where $W = R^{\mathrm{H}}R$ is its Cholesky decomposition. As a consequence.

$$H_{\mathrm{SR}} = \begin{bmatrix} 0 & G \\ I & 0 \end{bmatrix}, \qquad \text{where } G = W_1^{-\mathrm{H}}U^{\mathrm{H}}KUW_1^{-1}. \tag{2.10}$$

Furthermore, if $(\mu^2, q)$ is an eigenpair of $G$, then $(\mu, \begin{bmatrix} \mu q \\ q \end{bmatrix})$ is an eigenpair of $H_{\mathrm{SR}}$ by Lemma 2.1(b). Therefore a corresponding approximate eigenpair of $H$ can be taken as [2]

$$\left(\mu, \begin{bmatrix} \mu V R^{-1}q \\ UR^{-1}q \end{bmatrix}\right). \tag{2.11}$$

Algorithm 2.2 presents a piece of pseudo-code for a block Chebyshev-Davidson type method to compute the first $n_{\mathrm{want}}$ eigenpairs of LREP. In the algorithm, we apply the Chebyshev filter in Algorithm 2.1 to expand subspaces and seek the best approximate eigenpairs of $H$ by computing the eigenpairs of the small size-$2k$ matrix $H_{\mathrm{SR}}$ as in Eq. (2.10). For convenience, we identify this algorithm by `BChebyDLR`($\ell$), where $\ell$ is the block size. We prefer this block version because usually block type methods with relatively small block sizes are more competitive than non-block versions, especially when the desired eigenvalues have clusters or even multiples.

[2] This is only true if $K$ is at least positive semi-definite. But later we show numerically that the eigenvalues of $H_{\mathrm{SR}}$ still provide very good approximations to $\lambda_i$ for $i = 1, 2, \ldots, k$ even when $K$ is indefinite.

**Algorithm 2.2** BChebyDLR($\ell$)

(i) $V_0 = \texttt{randn}(n, \ell)$, $\widetilde{V} = V_0$, $V = U = [\,]$, $j = 1$,
$k = 0$ ($k$ tracks the number of columns in $V$ and $U$),
$s = 0$ ($s$ is the dimension of $G$),
$n_c = 0$ ($n_c$ is the number of converged eigenpairs).
(ii) Initial estimates: $\beta, \alpha, \alpha_0$, e.g., by Algorithm 3.2.
(iii) **while** ($j \le j_{\max}$) **do**
1 $\widetilde{V} = \texttt{Chebyshev\_filter}(\widetilde{V}, m, \beta, \alpha, \alpha_0)$.
2 Orthogonalize $\widetilde{V}$ against $U$ to obtain new $\widetilde{V}$.
3 Compute $\widetilde{U} = M\widetilde{V}$, $\widetilde{V}^{\mathrm{H}}\widetilde{U} = R^{\mathrm{H}}R$, and $\widehat{U} = \widetilde{U}R^{-1}$, $\widehat{V} = \widetilde{V}R^{-1}$.
4 Let $\widehat{E} = K\widehat{U}$, $s = s + \mathrm{size}(\widehat{U}, 2)$, $k = s + n_c$, $V = [V, \widehat{V}]$ and $U = [U, \widehat{U}]$.
5 $\widehat{G} = U(:, n_c + 1 : k)^{\mathrm{H}}\widehat{E}$, $E = [E, \widehat{E}]$, and $G = \begin{bmatrix} G & \widehat{G} \\ \widehat{G}^{\mathrm{H}} & \widehat{U}^{\mathrm{H}}\widehat{E} \end{bmatrix}$.
6 Compute the spectral decomposition: $G = Q\Omega Q^{\mathrm{H}}$ with the diagonal elements of $\Omega$ in the ascending order, and let $k_{\mathrm{old}} = k$.
7 **if** ($s > s_{\max}$) **then** restart: $s = s_{\mathrm{keep}}$, $k = n_c + s_{\mathrm{keep}}$.
8 Update $U_{(:,n_c+1:k)} = U_{(:,n_c+1:k_{\mathrm{old}})}Q_{(:,1:s)}$, $V_{(:,n_c+1:k)} = V_{(:,n_c+1:k_{\mathrm{old}})}Q_{(:,1:s)}$ and $E = EQ_{(:,1:s)}$.
9 Compute $\ell$ Ritz pairs $(\rho_i, \hat{z}_i)$ of $G$, with $\hat{z}_i = \begin{bmatrix} \hat{y}_i \\ \hat{x}_i \end{bmatrix}$, where $\rho_i = \Omega_{(i,i)}^{\frac{1}{2}}$, $\hat{x}_i = U_{(:,n_c+i)}$ and $\hat{y}_i = \rho_i V_{(:,n_c+i)}$.
10 Test for convergence; $t_c$ is the number of newly converged Ritz pairs; store the converged Ritz pairs in the ascending order, and let $n_c = n_c + t_c$.
11 **if** ($n_c \ge n_{\mathrm{want}}$) **then** return.
12 **if** ($t_c > 0$), $s_{\mathrm{old}} = s$ **then** $s = s - t_c$, $E_{(:,1:s)} = E_{(:,t_c+1:s_{\mathrm{old}})}$ and $G_{(1:s,1:s)} = \Omega_{(t_c+1:s_{\mathrm{old}},t_c+1:s_{\mathrm{old}})}$.
13 $\alpha = \texttt{median}(\mathrm{diag}(\Omega))$, $\alpha_0 = \min(\mathrm{diag}(\Omega))$, and update $\beta$ if necessary.
14 $\widetilde{V} = V_{(:,n_c+1:n_c+\ell)}$.
15 $j = j + 1$.
**end**

A few remarks regarding Algorithm 2.2 are in order:

1. In our MATLAB implementation, initially we set $V_0$ to a random $n \times \ell$ matrix, e.g., as generated by MATLAB's `randn`, for simplicity. But if a better $V_0$ is known, we should use it.
2. Initially $k = s$, and then $k = s + n_c$ if some of the wanted Ritz pairs have converged, where $n_c$ is the number of converged Ritz pairs (2.11). Converged $Uq$ and $Vq$ are stored in the first $n_c$ columns of $U$ and $V$, respectively.
3. According to Lemma 2.1(c), deflations are done by orthogonalizing the newly generated block $\widetilde{V}$ against $U_{(:,1:n_c)}$ and $\widetilde{U}$ against $V_{(:,1:n_c)}$. However, in Algorithm 2.2 the orthogonality between the columns of $\widetilde{U}$ and those of $V_{(:,1:n_c)}$ is implied due to orthogonalization at step 2, according to Theorem 2.1 below.
4. Algorithm 3.2 in the next section provides an initial crude estimate of $\alpha$, i.e., low bound of unwanted eigenvalues of $KM$, and the parameter $\alpha_0$ for scaling. We will discuss this estimator in detail later. These estimates $\alpha_0$ and $\alpha$ are updated at step 14, where `median`(diag($\Omega$)) is the MATLAB built-in function to return the

median value of the elements in the vector diag($\Omega$). Another choice for updating $\alpha$ was discussed in [30].

5. The estimated $\beta$ by Algorithm 3.2 is a strict upper bound under a condition as listed in (3.2). But this condition is not easy to verify without expensive computations. When it is not satisfied, it is possible that the bound $\beta$ returned from Algorithm 3.2 may be an underestimate, which can make the filter ineffective if the upper bound is not updated. In the numerical examples presented in Section 4 and many others we tried, $\beta$ by Algorithm 3.2 is indeed a strict upper bound. However, to make the code robust, we need a way to address the case that $\beta$ happens to be an underestimate. Fortunately this can be easily done by an adaptive strategy, as discussed at the end of Section 3. This adaptive strategy is applied in the numerical tests in Section 4.
6. At step 10, the convergence of a Ritz pair $(\rho_i, \hat{z}_i)$ is tested by checking if its relative residual norm

$$\frac{\|H\hat{z}_i - \rho_i \hat{z}_i\|_1}{(\|H\|_1 + |\rho_i|)\|\hat{z}_i\|_1} \leq \texttt{tol}, \tag{2.12}$$

where `tol` is a pre-set tolerance. At the same time, the swap procedure is to keep the converged Ritz values in the ascending order to make sure no wanted eigenvalues are missed. If $K$ is indefinite, then pure imaginary Ritz values will appear. In this case, we sort them by the ascending order of their squares. Once we have new converged Ritz values, a purging procedure is used to remove the associated columns of $E$ as well as the diagonal elements of $G$. This is simply done at step 12.

Computationally, the matrix spectral norm $\|\cdot\|_2$ is considerably more difficult to compute that the $\ell_1$ matrix-operator norm $\|\cdot\|_1$. It is this consideration that leads us to use the $\ell_1$ vector and matrix norm $\|\cdot\|_1$ in Eq. (2.12). Theoretically, the $\ell_2$ vector and matrix-operator norm $\|\cdot\|_2$ are probably more preferable.

**Theorem 2.1** *Let $U$, $V$ be two $n \times k$ matrices generated by Algorithm 2.2. Then $V^{\mathrm{H}}U = V^{\mathrm{H}}MV = I_k$.*

*Proof* To distinguish the two $U$'s and $V$'s, we rewrite the assignments at step 4 as $U = [U_{\mathrm{old}}, \widehat{U}]$ and $V = [V_{\mathrm{old}}, \widehat{V}]$. During the first sweep of the **while** loop, $V^{\mathrm{H}}U = I_\ell$ at step 4 because of how $\widehat{U}$ and $\widehat{V}$ are defined at step 3. Therefore $V_{\mathrm{old}}^{\mathrm{H}}U_{\mathrm{old}} = I_{\tilde{k}}$ is ensured from the previous sweep, where $\tilde{k} = k - \ell$. Since $\widehat{V}^{\mathrm{H}}\widehat{U} = I_\ell$ (because of step 3), and $\widehat{V}^{\mathrm{H}}U_{\mathrm{old}} = 0$ (because of step 2 and step 3), we have

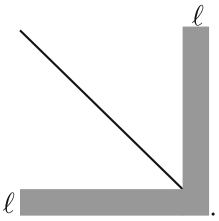
$$V_{\mathrm{old}}^{\mathrm{H}}\widehat{U} = V_{\mathrm{old}}^{\mathrm{H}}M\widehat{V} = V_{\mathrm{old}}^{\mathrm{H}}M^{\mathrm{H}}\widehat{V} = U_{\mathrm{old}}^{\mathrm{H}}\widehat{V} = 0,$$

and therefore

$$V^{\mathrm{H}}U = \begin{bmatrix} V_{\mathrm{old}}^{\mathrm{H}}U_{\mathrm{old}} & V_{\mathrm{old}}^{\mathrm{H}}\widehat{U} \\ \widehat{V}^{\mathrm{H}}U_{\mathrm{old}} & \widehat{V}^{\mathrm{H}}\widehat{U} \end{bmatrix} = I_k,$$

as expected. □

The eigenpairs of $H_{\text{SR}}$ in Eq. (2.10) are obtained by computing the eigenpairs of $G$, where $G$ at every step takes the following form:

$G =$ 


This form is preserved in the algorithm when refining the basis matrices $U$ and $V$ at step 8. The new basis matrices $UQ$ and $VQ$ are reassigned to $U$ and $V$. Refinement of basis matrices usually is applied in the thick restart method [12, 28] when the dimension of subspace exceeds the pre-set maximum dimension. Here, we refine the basis matrices $U$ and $V$ at every step. Although extra costs are incurred, eigenvectors often converge in fewer iterations [18, 22, 33]. Furthermore, the restart is easily executed by resetting the integers $s$ and $k$ when the size of $G$ exceeds $s_{\max}$. The restart at step 7 is to keep the size of $U$ and $G$ under control. There are many ways to specify $s_{\max}$ and $s_{\text{keep}}$. In our numerical examples, we simply choose $s_{\max} = 2n_{\text{want}}$ and $s_{\text{keep}} = n_{\text{want}}$.

### 2.3 Convergence analysis for $\lambda_1$

A convergence analysis of the Chebyshev-Davidson method for the Hermitian eigenvalue problem was given in [30, 33]. Here, we analyze the convergence rate of Algorithm 2.2 per iteration step. Using the technique in [14] for estimating the accuracy of the smallest Ritz eigenvalue, we obtain an estimate in Theorem 2.2 below. In fact, Theorem 2.2 after minor modifications also holds for the Chebyshev-Davidson method for the Hermitian eigenvalue problem.

Suppose $\begin{bmatrix} v \\ u \end{bmatrix}$ is an approximate eigenvector corresponding to the first eigenvalue $\lambda_1$ of $H$, where $u = Mv$. We define

$$\rho(u, v) = \frac{u^{\text{H}} K u}{u^{\text{H}} v}.$$

Basing on the claim we made about the eigenvalues of Eq. (2.9) there, we can show that $\sqrt{\rho(u, v)}$ is the best approximation of $\lambda_1$ within the pair $\{u, v\}$. For proving Theorem 2.2, we need the following lemma that says $\rho(u, v)$ is an upper bound for the smallest Ritz value from $\{\text{span}(U), \text{span}(V)\}$.

**Lemma 2.3** *Let $U \in \mathbb{C}^{n\times k}$, $V \in \mathbb{C}^{n\times k}$ and $\rho_i$ for $1 \le i \le k$ be computed in Algorithm 2.2, and let $u \in \text{span}(U)$, $v \in \text{span}(V)$, $u = Mv$ and $\mu^2 = \min_{1\le i\le k} \rho_i^2$. Then $\rho(u, v) \ge \mu^2$.*

*Proof* As we know in Algorithm 2.2, $\rho_i^2 = \Omega_{(i,i)}$ for $1 \le i \le k$, where the diagonal elements of $\Omega$ are the eigenvalues of $G$. Let $v = V\tilde{v}$, where $\tilde{v} \in \mathbb{C}^k$. We have,

$$\mu^2 = \min_{1\le i\le k} \rho_i^2 = \min_{g\in\mathbb{C}^k} \frac{g^{\mathrm{H}} G g}{g^{\mathrm{H}} g} = \min_{g\in\mathbb{C}^k} \frac{g^{\mathrm{H}} V^{\mathrm{H}} M K M V g}{g^{\mathrm{H}} g},$$

$$\begin{aligned}\rho(u, v) &= \frac{u^{\mathrm{H}} K u}{u^{\mathrm{H}} v} = \frac{v^{\mathrm{H}} M K M v}{v^{\mathrm{H}} M v} \\ &= \frac{\tilde{v}^{\mathrm{H}} V^{\mathrm{H}} M K M V \tilde{v}}{\tilde{v}^{\mathrm{H}} V^{\mathrm{H}} M V \tilde{v}} \\ &= \frac{\tilde{v}^{\mathrm{H}} V^{\mathrm{H}} M K M V \tilde{v}}{\tilde{v}^{\mathrm{H}} \tilde{v}}.\end{aligned}$$

The last equality holds because of $V^{\mathrm{H}} M V = I_k$ by Theorem 2.1. Hence, $\rho(u, v) \ge \mu^2$. □

**Theorem 2.2** *Let* $(\mu_j, \begin{bmatrix} v_j \\ u_j \end{bmatrix})$ *be from the* $j$*-th iteration of Algorithm 2.2, which is the* $j$*-th approximation of the exact eigenvalue* $\lambda_1$ *and eigenvector* $z_1 = \begin{bmatrix} \lambda_1 y_1 \\ x_1 \end{bmatrix}$*, where* $y_1$ *and* $x_1$ *are defined by Eq. (2.6). Let* $\alpha_{j+1}$ *and* $\beta_{j+1}$ *be the parameters* $\alpha$ *and* $\beta$ *used in the* $(j+1)$*-st Chebyshev filter step, and* $m$ *the degree of the Chebyshev polynomial. Define* $\phi(t)$*, similarly to Eq. (2.3), as*

$$\phi(t) = \frac{2t - (\alpha_{j+1} + \beta_{j+1})}{\beta_{j+1} - \alpha_{j+1}}.$$

*Then*

$$\frac{\mu_{j+1}^2 - \lambda_1^2}{\mu_j^2 - \lambda_1^2} \le \left(\frac{1-\xi}{1+\xi}\right)^2 + O(\delta_j^2), \tag{2.13}$$

*where*

$$\xi = \frac{\mathscr{T}_m(\phi(\lambda_1^2)) - \mathscr{T}_m(\phi(\lambda_\diamond^2))}{\mathscr{T}_m(\phi(\lambda_1^2)) - \mathscr{T}_m(\phi(\lambda_*^2))} \quad \textit{with}$$

$$\mathscr{T}_m(\phi(\lambda_*^2)) = \min_{i\ne 1} \mathscr{T}_m(\phi(\lambda_i^2)), \quad \mathscr{T}_m(\phi(\lambda_\diamond^2)) = \max_{i\ne 1} \mathscr{T}_m(\phi(\lambda_i^2)),$$

*and* $\delta_j \to 0$*, where* $\delta_j = \sin\angle_M(v_j, y_1)$ *is the sine of the angle between* $v_j$ *to* $y_1$ *in the* $M$*-inner product.*

*Proof* Using the same notations as before, we know $KM = Y\Lambda^2 X^{\mathrm{H}}$ with $Y = [y_1, y_2, \ldots, y_n]$, $X = [x_1, x_2, \ldots, x_n]$, and

$$v_j = \mu_j V q_1, \quad u_j = U q_1, \tag{2.14}$$

where $q_1$ is the eigenvector of $G$ corresponding to $\mu_j$ and $\|q_1\|_2 = 1$.

Let $\|\cdot\|_M$ denote the norm induced by the $M$-inner product. Since $My_1 = x_1$ and $y_1^{\mathrm{H}}x_1 = 1$, we have $\|y_1\|_M = 1$. Set

$$\tilde{v}_j = \frac{v_j}{\|v_j\|_M} = \frac{\mu_j V q_1}{\mu_j} = V q_1.$$

Then $\tilde{v}_j$ has the decomposition $\tilde{v}_j = \gamma_j y_1 + \delta_j s_j$, where $s_j \in \mathrm{span}\{y_2, \ldots, y_n\}$, $\|s_j\|_M = 1$ and $\gamma_j^2 + \delta_j^2 = 1$. As stated in [30, 33], $\delta_j \to 0$ at least as fast as by the power method.

By Eq. (2.14), we have $u_j = M\tilde{v}_j = \gamma_j My_1 + \delta_j Ms_j$, and it is not hard to see that $y_1^{\mathrm{H}}MKMs_j = 0$ and $y_1^{\mathrm{H}}Ms_j = 0$. Therefore,

$$\begin{aligned}
\mu_j^2 &= q_1^{\mathrm{H}}Gq_1 = u_j^{\mathrm{H}}Ku_j \\
&= (\gamma_j My_1 + \delta_j Ms_j)^{\mathrm{H}}K(\gamma_j My_1 + \delta_j Ms_j) \\
&= (\gamma_j My_1 + \delta_j Ms_j)^{\mathrm{H}}(\lambda_1^2\gamma_j y_1 + \delta_j KMs_j) \\
&= \gamma_j^2\lambda_1^2 + \delta_j^2 s_j^{\mathrm{H}}MKMs_j \\
&= \lambda_1^2 + \delta_j^2(s_j^{\mathrm{H}}MKMs_j - \lambda_1^2). \qquad (2.15)
\end{aligned}$$

Let $\hat{v}_{j+1}$ be a linear combination of $\tilde{v}_j$ and $\mathscr{T}_m(\phi(KM))\tilde{v}_j$, i.e., $\hat{v}_{j+1} = \tilde{v}_j - \tau\mathscr{T}_m(\phi(KM))\tilde{v}_j$ for some scalar $\tau$, and let $\hat{u}_{j+1} = M\hat{v}_{j+1}$. Then

$$\begin{aligned}
\hat{v}_{j+1} &= \gamma_j[1 - \tau\mathscr{T}_m(\phi(\lambda_1^2))]y_1 + \delta_j[I - \tau\mathscr{T}_m(\phi(KM))]s_j, \\
\hat{u}_{j+1} &= \gamma_j[1 - \tau\mathscr{T}_m(\phi(\lambda_1^2))]My_1 + \delta_j M[I - \tau\mathscr{T}_m(\phi(KM))]s_j.
\end{aligned}$$

The best approximation of $\lambda_1$ within the pair $\{\hat{u}_{j+1}, \hat{v}_{j+1}\}$ is $\sqrt{\rho(\hat{u}_{j+1}, \hat{v}_{j+1})}$. Substituting $t_1 = 1 - \tau\mathscr{T}_m(\phi(\lambda_1^2))$ and $\widetilde{T} = I - \tau\mathscr{T}_m(\phi(KM))$ in $\rho(\hat{u}_{j+1}, \hat{v}_{j+1})$, we get

$$\begin{aligned}
\rho(\hat{u}_{j+1}, \hat{v}_{j+1}) &= \frac{\hat{u}_{j+1}^{\mathrm{H}}K\hat{u}_{j+1}}{\hat{u}_{j+1}^{\mathrm{H}}\hat{v}_{j+1}} \\
&= \frac{\gamma_j^2 t_1^2\lambda_1^2 + \delta_j^2 s_j^{\mathrm{H}}\widetilde{T}^{\mathrm{H}}MKM\widetilde{T}s_j}{\gamma_j^2 t_1^2 + \delta_j^2 s_j^{\mathrm{H}}\widetilde{T}^{\mathrm{H}}M\widetilde{T}s_j} \\
&= \lambda_1^2 + \frac{\delta_j^2 s_j^{\mathrm{H}}\widetilde{T}^{\mathrm{H}}(MKM - \lambda_1^2 M)\widetilde{T}s_j}{\gamma_j^2 t_1^2 + \delta_j^2 s_j^{\mathrm{H}}\widetilde{T}^{\mathrm{H}}M\widetilde{T}s_j} \\
&= \lambda_1^2 + \frac{(\delta_j^2/t_1^2)s_j^{\mathrm{H}}\widetilde{T}^{\mathrm{H}}(MKM - \lambda_1^2 M)\widetilde{T}s_j}{1 + (\delta_j^2/t_1^2)(s_j^{\mathrm{H}}\widetilde{T}^{\mathrm{H}}M\widetilde{T}s_j - t_1^2)} \\
&= \lambda_1^2 + (\delta_j^2/t_1^2)s_j^{\mathrm{H}}\widetilde{T}^{\mathrm{H}}(MKM - \lambda_1^2 M)\widetilde{T}s_j + O(\delta_j^4).
\end{aligned}$$

Let $\mu_{j+1}$ be the $(j+1)$-st approximation of $\lambda_1$ computed by Algorithm 2.2. Then by Lemma 2.3, we have $\mu_{j+1}^2 \le \rho(\hat{u}_{j+1}, \hat{v}_{j+1})$. Set

$$\widehat{T} = I - \tau\mathscr{T}_m(\phi(\Lambda^2)), \quad t_\tau^2 = \max_{i\neq 1}\left[1 - \tau\mathscr{T}_m(\phi(\lambda_i^2))\right]^2.$$

We have

$$\begin{aligned}
\mu_{j+1}^2 - \lambda_1^2 &\le \rho(\hat{u}_{j+1}, \hat{v}_{j+1}) - \lambda_1^2 \\
&= (\delta_j^2/t_1^2) s_j^{\mathrm{H}} \widetilde{T}^{\mathrm{H}} (MKM - \lambda_1^2 M) \widetilde{T} s_j + O(\delta_j^4) \\
&= (\delta_j^2/t_1^2) s_j^{\mathrm{H}} X \widehat{T} (\Lambda^2 - \lambda_1^2 I) \widehat{T} X^{\mathrm{H}} s_j + O(\delta_j^4) \\
&= (\delta_j^2/t_1^2) s_j^{\mathrm{H}} X (\Lambda^2 - \lambda_1^2 I)^{\frac{1}{2}} \widehat{T} \widehat{T} (\Lambda^2 - \lambda_1^2 I)^{\frac{1}{2}} X^{\mathrm{H}} s_j + O(\delta_j^4) \\
&\le (t_\tau^2/t_1^2) \delta_j^2 s_j^{\mathrm{H}} X (\Lambda^2 - \lambda_1^2 I)^{\frac{1}{2}} (\Lambda^2 - \lambda_1^2 I)^{\frac{1}{2}} X^{\mathrm{H}} s_j + O(\delta_j^4) \\
&= (t_\tau^2/t_1^2) \delta_j^2 s_j^{\mathrm{H}} (MKM - \lambda_1^2 M) s_j + O(\delta_j^4) \qquad \text{(by Eq. 2.15)} \\
&= (t_\tau^2/t_1^2)(\mu_j^2 - \lambda_1^2) + O(\delta_j^4).
\end{aligned}$$

Therefore, by Eq. (2.15), we have $\mu_j^2 - \lambda_1^2 = O(\delta_j^2)$ and

$$\frac{\mu_{j+1}^2 - \lambda_1^2}{\mu_j^2 - \lambda_1^2} \le \min_\tau \frac{t_\tau^2}{t_1^2} + O(\delta_j^2).$$

From the definition of $t_\tau$, we know that

$$t_\tau^2 = \max\left\{ \left[1 - \tau \mathscr{T}_m(\phi(\lambda_\diamond^2))\right]^2, \left[1 - \tau \mathscr{T}_m(\phi(\lambda_*^2))\right]^2 \right\},$$

where $\mathscr{T}_m(\phi(\lambda_*^2)) = \min_{i \ne 1} \mathscr{T}_m(\phi(\lambda_i^2))$ and $\mathscr{T}_m(\phi(\lambda_\diamond^2)) = \max_{i \ne 1} \mathscr{T}_m(\phi(\lambda_i^2))$.

We first consider the case where $\mathscr{T}_m(\phi(\lambda_\diamond^2)) + \mathscr{T}_m(\phi(\lambda_*^2)) \ge 0$. Under this additional condition,

$$t_\tau^2 = \begin{cases} \left[1 - \tau \mathscr{T}_m(\phi(\lambda_\diamond^2))\right]^2, & \text{for } \tau \ge 2/[\mathscr{T}_m(\phi(\lambda_\diamond^2)) + \mathscr{T}_m(\phi(\lambda_*^2))] \text{ or } \tau \le 0, \\ \left[1 - \tau \mathscr{T}_m(\phi(\lambda_*^2))\right]^2, & \text{for } 0 \le \tau \le 2/[\mathscr{T}_m(\phi(\lambda_\diamond^2)) + \mathscr{T}_m(\phi(\lambda_*^2))]. \end{cases}$$

For $0 \le \tau \le 2/[\mathscr{T}_m(\phi(\lambda_\diamond^2)) + \mathscr{T}_m(\phi(\lambda_*^2))]$,

$$\frac{\mathrm{d}(t_\tau/t_1)^2}{\mathrm{d}\tau} = \frac{2\left[\mathscr{T}_m(\phi(\lambda_1^2)) - \mathscr{T}_m(\phi(\lambda_*^2))\right]\left[\tau \mathscr{T}_m(\phi(\lambda_1^2)) - 1\right]\left[\tau \mathscr{T}_m(\phi(\lambda_*^2)) - 1\right]}{\left[1 - \tau \mathscr{T}_m(\phi(\lambda_1^2))\right]^4}.$$

Therefore, if $\mathscr{T}_m(\phi(\lambda_1^2)) > 0$, then

$$\begin{cases} \dfrac{\mathrm{d}(t_\tau/t_1)^2}{\mathrm{d}\tau} \ge 0, & \text{for } 0 \le \tau \le 1/[\mathscr{T}_m(\phi(\lambda_1^2))], \\ \dfrac{\mathrm{d}(t_\tau/t_1)^2}{\mathrm{d}\tau} \le 0, & \text{for } 1/[\mathscr{T}_m(\phi(\lambda_1^2))] \le \tau \le 2/[\mathscr{T}_m(\phi(\lambda_\diamond^2)) + \mathscr{T}_m(\phi(\lambda_*^2))]. \end{cases}$$

Meanwhile, for $\tau \ge 2/[\mathscr{T}_m(\phi(\lambda_\diamond^2)) + \mathscr{T}_m(\phi(\lambda_*^2))]$ or $\tau \le 0$, it is not hard to prove

$$\begin{cases} \dfrac{\mathrm{d}(t_\tau/t_1)^2}{\mathrm{d}\tau} \ge 0, & \text{for } \tau \le 0, \\ \dfrac{\mathrm{d}(t_\tau/t_1)^2}{\mathrm{d}\tau} \ge 0, & \text{for } \tau \ge 2/[\mathscr{T}_m(\phi(\lambda_\diamond^2)) + \mathscr{T}_m(\phi(\lambda_*^2))]. \end{cases}$$

When $\mathscr{T}_m(\phi(\lambda_1^2)) < 0$, we have

$$\begin{cases} \dfrac{\mathrm{d}(t_\tau/t_1)^2}{\mathrm{d}\tau} \le 0, \ \text{for } 0 \le \tau \le 2\big/[\mathscr{T}_m(\phi(\lambda_\diamond^2)) + \mathscr{T}_m(\phi(\lambda_*^2))], \\ \dfrac{\mathrm{d}(t_\tau/t_1)^2}{\mathrm{d}\tau} \le 0, \ \text{for } 1\big/\mathscr{T}_m(\phi(\lambda_1^2))] \le \tau \le 0, \\ \dfrac{\mathrm{d}(t_\tau/t_1)^2}{\mathrm{d}\tau} \ge 0, \ \text{for } \tau \le 1\big/\mathscr{T}_m(\phi(\lambda_1^2))], \\ \dfrac{\mathrm{d}(t_\tau/t_1)^2}{\mathrm{d}\tau} \ge 0, \ \text{for } \tau \ge 2\big/[\mathscr{T}_m(\phi(\lambda_\diamond^2)) + \mathscr{T}_m(\phi(\lambda_*^2))]. \end{cases}$$

Therefore

$$\begin{aligned} \min_\tau \frac{t_\tau^2}{t_1^2} &= \frac{\left(1 - \frac{2}{\mathscr{T}_m(\phi(\lambda_\diamond^2)) + \mathscr{T}_m(\phi(\lambda_*^2))} \mathscr{T}_m(\phi(\lambda_\diamond^2))\right)^2}{\left(1 - \frac{2}{\mathscr{T}_m(\phi(\lambda_\diamond^2)) + \mathscr{T}_m(\phi(\lambda_*^2))} \mathscr{T}_m(\phi(\lambda_1^2))\right)^2} \\ &= \frac{\left(\mathscr{T}_m(\phi(\lambda_\diamond^2)) - \mathscr{T}_m(\phi(\lambda_*^2))\right)^2}{\left(\mathscr{T}_m(\phi(\lambda_\diamond^2)) + \mathscr{T}_m(\phi(\lambda_*^2)) - 2\mathscr{T}_m(\phi(\lambda_1^2))\right)^2} \\ &= \left(\frac{1-\xi}{1+\xi}\right)^2, \end{aligned}$$

where $\xi = [\mathscr{T}_m(\phi(\lambda_1^2)) - \mathscr{T}_m(\phi(\lambda_\diamond^2))]\big/[\mathscr{T}_m(\phi(\lambda_1^2)) - \mathscr{T}_m(\phi(\lambda_*^2))]$.

The remaining case where $\mathscr{T}_m(\phi(\lambda_\diamond^2)) + \mathscr{T}_m(\phi(\lambda_*^2)) \le 0$ can be addressed in a similar way to derive the same result. □

Once the convergence rate for $\lambda_1$ is established, according to the remark 3 for Algorithm 2.2, the convergence analysis for other eigenvalues $\lambda_2, \cdots, \lambda_k$ can be addressed using the argument of deflation. For example, with a similar argument, the convergence rate formula for $\lambda_2$ can be gotten as

$$\frac{\mu_{j+1}^2 - \lambda_2^2}{\mu_j^2 - \lambda_2^2} \le \left(\frac{1-\xi}{1+\xi}\right)^2 + O(\delta_j^2), \tag{2.16}$$

where

$$\xi = \frac{\mathscr{T}_m(\phi(\lambda_2^2)) - \mathscr{T}_m(\phi(\lambda_\diamond^2))}{\mathscr{T}_m(\phi(\lambda_2^2)) - \mathscr{T}_m(\phi(\lambda_*^2))} \quad \text{with}$$

$$\mathscr{T}_m(\phi(\lambda_*^2)) = \min_{i \ne 1,2} \mathscr{T}_m(\phi(\lambda_i^2)), \quad \mathscr{T}_m(\phi(\lambda_\diamond^2)) = \max_{i \ne 1,2} \mathscr{T}_m(\phi(\lambda_i^2)),$$

and $\delta_j = \sin \angle_M(v_j, y_2)$.

Similar convergence estimate appears in the constant-shift stationary Richardson method for computing the largest eigenvalue of generalized eigenvalue problems [10]. More convergence estimates for the generalized Davidson method can be found in [15, 16].

## 3 Upper bound for eig($KM$)

Theoretically, $\|K\|_1\|M\|_1$ and $\|K\|_\infty\|M\|_\infty$ are upper bounds of $\lambda^2$ and they are easily computable, where $\lambda$ is an eigenvalue of $H$, but they are usually too large for the Chebyshev filter to be effective. A sharper upper bound is needed. The $k$-step Lanczos method for computing a sharp upper bound of the spectrum of a symmetric matrix is proposed in [32]. In this section, we propose a similar technique to compute an upper bound of the spectrum of $KM$.

**Lemma 3.1** ([5, Theorem 1]) *Let $\Omega$ be an $n \times n$ real diagonal matrix, $D$ a $k \times k$ diagonal matrix with diagonal elements $d_1 \le d_2 \le \cdots \le d_k$, and let $Q$ be an $n \times k$ matrix having full column rank, where $1 \le k \le n$. Then there exist $k$ diagonal elements $\mu_{i_1} \le \mu_{i_2} \le \cdots \le \mu_{i_k}$ of $\Omega$ such that*

$$\|\Omega Q - QD\|_2 \ge \max_{1\le j\le k} |d_j - \mu_{i_j}|\sigma_{\min}(Q),$$

*where $\sigma_{\min}(Q)$ is the smallest singular value of $Q$.*

To state the next theorem, we introduce, in addition to the notations in Lemma 2.1, $\lambda_i(KM)$ for $1 \le i \le n$ for the eigenvalues of $KM$ in the ascending order, and

$$\lambda_{\max}(KM) := \max_i \lambda_i(KM) = \lambda_n(KM) = \lambda_n^2.$$

**Theorem 3.1** *Let $U$ and $V$ be two $n \times k$ matrices with full column rank, $H_K = U^{\mathrm{H}}KU$, and $H_M = V^{\mathrm{H}}MV$, where $1 \le k \le n$, and let $\theta_1 \le \theta_2 \le \cdots \le \theta_k$ be the eigenvalues of $H_K H_M$. Then there exist $k$ eigenvalues $\lambda_{i_1}^2 \le \lambda_{i_2}^2 \le \cdots \le \lambda_{i_k}^2$ of $KM$ such that*

$$\max_{1\le j\le k} |\theta_j - \lambda_{i_j}^2| \le \sqrt{\|M\|_2\|H_M^{-1}\|_2}\ \|KMV - VH_KH_M\|_2 \tag{3.1a}$$

$$\le \sqrt{\|M\|_1\|H_M^{-1}\|_1}\ \|KMV - VH_KH_M\|_2. \tag{3.1b}$$

*If, in addition,*

$$|\lambda_{\max}(KM) - \lambda_{\max}(H_KH_M)| = \min_{1\le i\le n} |\lambda_i(KM) - \lambda_{\max}(H_KH_M)|, \tag{3.2}$$

*where $\lambda_{\max}(H_KH_M) = \theta_k =: \theta_{\max}$, then*

$$\lambda_n^2 \le \theta_{\max} + \sqrt{\|M\|_2\|H_M^{-1}\|_2}\ \|KMV - VH_KH_M\|_2 \tag{3.3a}$$

$$\le \theta_{\max} + \sqrt{\|M\|_1\|H_M^{-1}\|_1}\ \|KMV - VH_KH_M\|_2. \tag{3.3b}$$

*Proof* The inequalities in Eq. (3.3) are consequences of Eq. (3.1) by Eq. (3.2). So we will prove (3.1) only. By Lemma 2.1(a), we have $K = Y\Lambda^2 Y^{\mathrm{H}}$,

$M = XX^{\mathrm{H}}$, $H_K = \widetilde{Y}\Theta\widetilde{Y}^{\mathrm{H}}$, and $H_M = \widetilde{X}\widetilde{X}^{\mathrm{H}}$, where $\Theta = \mathrm{diag}(\theta_1, \theta_2, \ldots, \theta_k)$. We have

$$\begin{aligned}\|KMV - VH_KH_M\|_2 &= \|Y\Lambda^2X^{\mathrm{H}}V - VH_KH_M\|_2\\ &= \|Y(\Lambda^2X^{\mathrm{H}}V - X^{\mathrm{H}}VH_KH_M)\|_2\\ &= \|Y(\Lambda^2X^{\mathrm{H}}V\widetilde{Y} - X^{\mathrm{H}}V\widetilde{Y}\Theta)\widetilde{X}^{\mathrm{H}}\|_2\\ &\geq \frac{\|\Lambda^2X^{\mathrm{H}}V\widetilde{Y} - X^{\mathrm{H}}V\widetilde{Y}\Theta\|_2}{\|X^{\mathrm{H}}\|_2\|\widetilde{Y}\|_2}.\end{aligned}$$

According to Lemma 3.1, there exist $\lambda_{i_1}^2 \leq \lambda_{i_2}^2 \leq \cdots \leq \lambda_{i_k}^2$ such that

$$\|\Lambda^2X^{\mathrm{H}}V\widetilde{Y} - X^{\mathrm{H}}V\widetilde{Y}\Theta\|_2 \geq \sigma_{\min}\big(X^{\mathrm{H}}V\widetilde{Y}\big)\max_{1\leq j\leq k}|\theta_j - \lambda_{i_j}^2| = \max_{1\leq j\leq k}|\theta_j - \lambda_{i_j}^2|.$$

The last equality holds because of $\sigma_{\min}\big(X^{\mathrm{H}}V\widetilde{Y}\big) = \sqrt{\lambda_{\min}\big(\widetilde{Y}^{\mathrm{H}}V^{\mathrm{H}}XX^{\mathrm{H}}V\widetilde{Y}\big)} = 1$. Therefore

$$\|KMV - VH_KH_M\|_2 \geq \frac{1}{\|X^{\mathrm{H}}\|_2\|\widetilde{Y}\|_2}\max_{1\leq j\leq k}|\theta_j - \lambda_{i_j}^2|.$$

Now use

$$\|X^{\mathrm{H}}\|_2 = \sqrt{\|M\|_2} \leq \sqrt{\|M\|_1}, \quad \|\widetilde{Y}\|_2 = \sqrt{\|H_M^{-1}\|_2} \leq \sqrt{\|H_M^{-1}\|_1}$$

to conclude the proof of Eq. (3.1). □

The inequality (3.3) provides upper bounds on $\lambda_n^2$. If the eigenvector associated with $\theta_{\max}$ is also available, we have the following theorem for an improved bound.

**Theorem 3.2** *Under the conditions of Theorem 3.1, including (3.2). Let $w$ be an eigenvector corresponding to the largest eigenvalue $\theta_{\max}$ of $H_KH_M$, normalized such that $w^{\mathrm{H}}H_Mw = 1$. Then*

$$\lambda_n^2 \leq \theta_{\max} + \sqrt{\|M\|_2}\,\|KMVw - VH_KH_Mw\|_2 \tag{3.4a}$$

$$\leq \theta_{\max} + \sqrt{\|M\|_1}\,\|KMVw - VH_KH_Mw\|_2. \tag{3.4b}$$

*Proof* We have

$$\begin{aligned}\|KMVw - VH_KH_Mw\|_2 &= \|KMVw - \theta_{\max}Vw\|_2\\ &= \|Y\Lambda^2X^{\mathrm{H}}Vw - \theta_{\max}Vw\|_2\\ &= \|Y(\Lambda^2 - \theta_{\max}I_n)X^{\mathrm{H}}Vw\|_2\\ &\geq \frac{\|(\Lambda^2 - \theta_{\max}I_n)X^{\mathrm{H}}Vw\|_2}{\|X^{\mathrm{H}}\|_2}\\ &\geq \frac{|w^{\mathrm{H}}V^{\mathrm{H}}XX^{\mathrm{H}}Vw|^{1/2}}{\|X^{\mathrm{H}}\|_2}|\lambda_n^2 - \theta_{\max}| \quad \text{(by Eq. 3.2)}\\ &\geq \frac{|w^{\mathrm{H}}V^{\mathrm{H}}MVw|^{1/2}}{\|X^{\mathrm{H}}\|_2}|\lambda_n^2 - \theta_{\max}|.\end{aligned}$$

Now use $\|X^{\mathrm{H}}\|_2 = \sqrt{\|M\|_2} \le \sqrt{\|M\|_1}$ to conclude the proof. □

**Algorithm 3.1 Lanczos biorthogonalization procedure**

$v = \mathtt{randn}(n,1)$, $u = Mv$, $b = \sqrt{u^{\mathrm{H}}v}$, $u_1 = u/b$, $v_1 = v/b$, $b_0 = 0$.
**for** $i = 1 : k$ **do**
  $a_i = u_i^{\mathrm{H}} K u_i$.
  $\tilde{v} = Ku_i - a_i v_i - b_{i-1} v_{i-1}$, $\tilde{u} = M\tilde{v}$.
  $b_i = \sqrt{\tilde{u}^{\mathrm{H}}\tilde{v}}$.
  $u_{i+1} = \tilde{u}/b_i$, $v_{i+1} = \tilde{v}/b_i$.
**end**

The bounds (3.4) can be significantly simplified if the following Lanczos biorthogonalization procedure [23, subsection 3.1] is applied to compute $\theta_{\max}$. Suppose that Algorithm 3.1 successfully runs to its completion, i.e., $b_i \neq 0$ for all $i$. Denote $U_k = [u_1, u_2, \ldots, u_k]$, $V_k = [v_1, v_2, \ldots, v_k]$, and

$$T_k = \begin{bmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{k-2} & a_{k-1} & b_{k-1} \\ & & & b_{k-1} & a_k \end{bmatrix}. \tag{3.5}$$

Then

$$KU_k = V_k T_k + b_k v_{k+1} e_k^{\mathrm{T}}, \quad MV_k = U_k,$$

and also $U_k^{\mathrm{H}} V_k = I_k$, $U_k^{\mathrm{H}} K U_k = T_k$, and $V_k^{\mathrm{H}} M V_k = I_k$.

**Corollary 3.1** *In Algorithm 3.1, let $T_k w = \theta_{\max} w$, $\|w\|_2 = 1$, where $\theta_{\max}$ is the largest eigenvalue of $T_k$. If Eq. (3.2) holds, then*

$$\lambda_n^2 \le \theta_{\max} + \sqrt{\|M\|_2}\ \|b_k (e_k^{\mathrm{T}} w) v_{k+1}\|_2 \tag{3.6a}$$

$$\le \theta_{\max} + \sqrt{\|M\|_1}\ \|b_k (e_k^{\mathrm{T}} w) v_{k+1}\|_2. \tag{3.6b}$$

**Algorithm 3.2 An estimator for the bounds used in the first filtering step**

**Input**: $K$, $M$ and the number of the Lanczos step $k_p$.
**Output**: $\beta$, $\alpha$ and $\alpha_0$ as discussed in subsection 2.1.
1 Generate $V_k$, $T_k$ $v_{k+1}$ and $b_k$ by Algorithm 3.1 with $k = k_p$.
2 Compute eigen-decomposition $T_k = Z^{\mathrm{H}} \Theta Z$, and let $(\theta_{\max}, w)$ be the largest eigenpair of $T_k$, where $\|w\|_2 = 1$.
3 Compute $\beta = \theta_{\max} + \sqrt{\|M\|_1}\ \|b_k (e_k^{\mathrm{T}} w) v_{k+1}\|_2$ by (3.6b), and let $\alpha = \mathtt{median}(\Theta)$, $\alpha_0 = \min(\Theta)$.

With these preparations, we propose Algorithm 3.2 to compute an upper bound for $\mathrm{eig}(KM)$. To test the effectiveness of Algorithm 3.2, we compose 6 test problems, listed as TEST 1 to TEST 6 in Table 1. The $K$- and $M$-matrices of TEST 1 to TEST 3 come from the linear response analysis for Na2, Na4, and silane (SiH4) compound, respectively. These matrices are generated by the turboTDDFT code in QUANTUM ESPRESSO — an electronic structure calculation code that implements density

**Table 1** Test problems

| Problem | TEST 1 | TEST 2 | TEST 3 | TEST 4 | TEST 5 | TEST 6 |
|---|---|---|---|---|---|---|
| $n$ | 1862 | 2834 | 5660 | 5832 | 5743 | 74752 |
| $K$ | Na2 | Na4 | $SiH_4$ | Na5 | SiNa | $SiO_2$ |
| $M$ | Na2 | Na4 | $SiH_4$ | fv1 | fv2 | finan512 |

functional theory (DFT) using plane-waves as the basis set and pseudopotentials [9]. For Na2, Na4 and SiH4, the matrices $K$ and $M$ are symmetric positive definite. TEST 4 to TEST 6, artificially constructed, consist of matrices from the University of Florida Sparse Matrix Collection [7] to give $K$ and $M$. In the case when the two matrices from the collection have different dimensions, we extract out the leading principal submatrix of the larger one to give $K$ and $M$ of equal size. For TEST 4 and TEST 6, the $M$-matrices are SPD, but the $K$-matrices are indefinite.

We apply the built-in function `eigs` of MATLAB with default tolerance $\texttt{tol} = 2.22 \times 10^{-16}$ to compute the largest eigenvalue $\lambda_n$ of associated $H$ to the "exact" eigenvalue bound $\lambda_n^2$. These test problems are also used in the next section to test `BChebyDLR`($\ell$) (Algorithm 2.2).

Figure 1 plots the upper bounds by Algorithm 3.2 as $k_p$ varies for the six tests, where the red dotted lines are for the "exact" $\lambda_n^2$, and the blue solid lines are the upper bounds by Algorithm 3.2. It demonstrates that Algorithm 3.2 is rather efficient and produces a fairly tight upper bound on $\lambda_n^2$ in a few Lanczos steps, such as within 10 steps.

The inequality (3.6b) is proved under the condition of Eq. (3.2) which, nevertheless, may fail for the first few Lanczos steps. When it does, it may underestimate $\lambda_n^2$

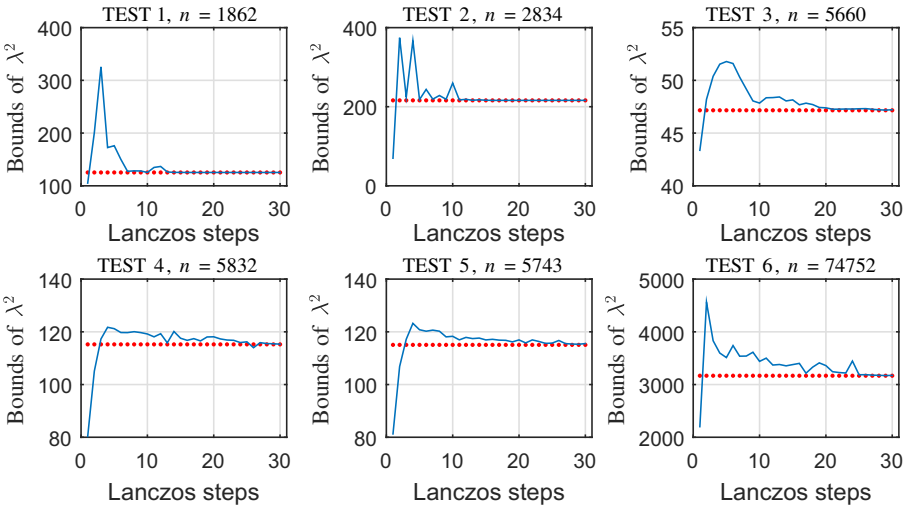



**Fig. 1** Behavior of the upper bound estimator. The red dotted lines are for the "exact" $\lambda_n^2$, and the blue solid lines are the upper bounds by Algorithm 3.2

as shown in Fig. 1. In practice such an underestimation does not cause any problem and can be overcome by using an adaptive strategy within Algorithm 2.2 to update the upper bound. Let $\rho_{\max}$ be the largest eigenvalue of $G$ in Algorithm 2.2. For any nonzero $g \in \mathbb{C}^k$, where $k$ is the dimension of $G$, we have

$$\begin{aligned}\rho_{\max} &= \max_{g\in\mathbb{C}^k} \frac{g^{\mathrm{H}} G g}{g^{\mathrm{H}} g} = \max_{g\in\mathbb{C}^k} \frac{g^{\mathrm{H}} V^{\mathrm{H}} M K M V g}{g^{\mathrm{H}} g} \\ &= \max_{g\in\mathbb{C}^k} \frac{g^{\mathrm{H}} V^{\mathrm{H}} X \Lambda^2 X^{\mathrm{H}} V g}{g^{\mathrm{H}} g} \qquad \text{(by Eq. 2.5)} \\ &\le \lambda_n^2 \max_{g\in\mathbb{C}^k} \frac{g^{\mathrm{H}} V^{\mathrm{H}} X X^{\mathrm{H}} V g}{g^{\mathrm{H}} g} \\ &= \lambda_n^2. \end{aligned} \tag{3.7}$$

Let $\beta$ be computed by Algorithm 3.2. Then by the sign of $\beta - \rho_{\max}$, we find whether we need to update the upper bound. Namely, if $\rho_{\max} > \beta$, then we update $\beta$ as $\beta = \rho_{\max}$. We demonstrate by numerical examples in the next section that this adaptive strategy upon integrating into `BChebyDLR` works very well.

## 4 Numerical results

We present some numerical results of Algorithm 2.2, denoted by `BChebyDLR`($\ell$), to compute the first $n_{\text{want}}$ eigenpairs. Recall that the first few $\lambda_i$ may be purely imaginary if $K$ is indefinite. A computed eigenpair $(\rho, \hat{z})$ is considered as converged when its relative residual norm is below a user specified `tol`, which by default is set to $10^{-8}$, i.e.,

$$\frac{\|H\hat{z} - \rho\hat{z}\|_1}{(\|H\|_1 + |\rho|)\|\hat{z}\|_1} \le \texttt{tol} = 10^{-8}.$$

The test matrices are TEST 1 to TEST 6 in Table 1. TEST 1 to TEST 3 are real LREPs from physics, and for the technical reason, the $K$- and $M$-matrices are dense and symmetric positive definite, whereas the matrices of TEST 4 to TEST 6 are sparse and symmetric with $K$-matrices being indefinite and $M$-matrices still being positive definite. We use these matrices to show the effectiveness of our proposed algorithm for eigenproblems of form (1.2), and this form includes LREP as a special case.

In all tests, we start with $V_0 = \texttt{randn}(n, \ell)$, where $\ell$ is the block size. The parameters $n_{\text{want}}$ and the Chebyshev filter degree $m$ are to be specified. The upper bound $\beta$ is obtained by Algorithm 3.2 with $k_p = 10$, and $s_{\max} = 2n_{\text{want}}$ when $n_{\text{want}} < 100$. But we use $s_{\max} \le \texttt{ceil}(1.5n_{\text{want}})$ when $n_{\text{want}} \ge 100$. For restart, we use $s_{\text{keep}} = n_{\text{want}}$.

The hardware used for the numerical computations is a laptop with 8G memory, the CPU type is Intel core i5-3210M @ 2.50GHz. We use MATLAB version 8.5 (R2015a).

*Example 4.1* We first test `BChebyDLR`$(\ell)$ on the three problems from the linear response analysis for Na2, Na4 and SiH4, i.e., TEST 1 to TEST 3. Since both $K$ and $M$

are positive definitive for these problems, we can compare BChebyDLR($\ell$) with the locally optimal block *preconditioned* 4-D CG method (LOBP4DCG) and the locally optimal block 4-D CG method (LOB4DCG) in [3].

For the LOBP4DCG method, we use the generic preconditioner

$$\Phi = \begin{bmatrix} 0 & M^{-1} \\ K^{-1} & 0 \end{bmatrix}. \tag{4.1}$$

The preconditioned search vectors $q_i$ and $p_i$ in [3] are computed by using the linear CG method [8]. Often very crude approximations of $q_i$ and $p_i$ are good enough. In this example, the linear CG iterations are set maximal 5 iterations. It is denoted by LOBP4DCG(5). The initial block size in LOBP4DCG(5) and LOB4DCG are chosen to be the same as $n_{\text{want}}$. A deflation procedure is built to purge out converged eigenvectors. So the block size decreases as the algorithms progress. As for BChebyDLR, converged eigenvectors are also deflated from the iteration subspace, and new basis vectors need to be orthogonalized against the deflated eigenvectors.

We report the total number of matrix-vector products (denoted by "#mvp"), iteration number (denoted by "#iter"), and CPU time in seconds for BChebyDLR, LOB4DCG and LOBP4DCG(5) in Table 2.

We make a few comments on the two methods applied to these definite LREPs. First, for the three tests as well as other tests not reported here, BChebyDLR($\ell$) appears to be close to an order of magnitude faster than LOBP4DCG. However, we also observe that for some other problems not reported here, the LOBP4DCG can be an order of magnitude faster than BChebyDLR($\ell$). The main reason for this quite drastic difference is the conditioning of the $KM$-matrix. This may be similar to SVD calculations, e.g., squaring of a matrix makes the smallest singular values harder to converge to high accuracy. Our algorithm appears to be more sensitive to the conditioning of $M$. If $M$ is well-conditioned, then generally BChebyDLR($\ell$) is much faster than LOBP4DCG; while for $M$ that has a relatively large condition number, the LOBP4DCG is usually much more efficient than BChebyDLR($\ell$). Therefore, the two methods appear to be complementary to each other. Second, the effectiveness

**Table 2** The number of matrix-vector products (#mvp), number of iterations (#iter), and CPU time in seconds for computing $n_{\text{want}} = 180$ eigenpairs. For BChebyDLR the filter degree used is $m = 25$, and the block size is $\ell = 15$. For LOB4DCG and LOBP4DCG(5) the initial block size is $n_{\text{want}}$

| | BChebyDLR($\ell$) | | | LOB4DCG | | | LOBP4DCG(5) | | |
|---|---|---|---|---|---|---|---|---|---|
| | #mvp | #iter | CPU | #mvp | #iter | CPU | #mvp | #iter | CPU |
| TEST 1 | 26520 | 34 | 49.9 | 40040 | 120 | 201.9 | 48600 | 61 | 235.9 |
| TEST 2 | 71760 | 92 | 223.5 | 63712 | 812 | 462.8 | 56394 | 239 | 305.4 |
| TEST 3 | 37440 | 48 | 295.5 | 109568 | 233 | 1028.5 | 161658 | 142 | 2960.2 |

The maximum subspace dimension used for BChebyDLR is 233, while LOB4DCG and LOBP4DCG(5) uses a subspace of maximum dimension $3n_{\text{want}} = 540$

of LOBP4DCG is closely related to the preconditioner used. We should note that the generic preconditioner (4.1) is not the natural preconditioner for TEST 1 to TEST 3. For plane wave-based calculations, it is more natural to use a proper scaled diagonal-like preconditioner proposed in [24]. In fact, $K$ and $M$ are not needed explicitly in plane wave-based calculations and they only exist in certain structural form, but $K$ and $M$ in our test were output from turboTDDFT runs. In such a case, the scaled diagonal preconditioner is lost once $K$ and $M$ are output explicitly as matrices. A future work is to compare them directly in turboTDDFT runs. Third, the advantage of BChebyDLR($\ell$) is that it does not need to apply preconditioners to solve linear equations. However, this convenience comes with a cost, namely that the current method is particularly effective only for the cases in which $M$ is well-conditioned. Further research is required to make the method also efficient for $M$ that has a relatively large condition number.

*Example 4.2* When the $K$-matrix is indefinite, LOBP4DCG becomes not applicable. Therefore we can only report numerical results of Algorithm 2.2 without any comparison to LOBP4DCG. Since no other iterative methods seem to exploit the special structure of (1.2), known eigenvalue algorithms would treat (1.2) as a size $2n \times 2n$ standard eigenvalue problem, which is clearly an expensive thing to do. Therefore we do not compare with any general eigensolver here either. Instead, we use this example to demonstrate some common properties of BChebyDLR($\ell$), such as the effect of block sizes (see Fig. 2).

We apply BChebyDLR($\ell$) to the matrices listed as TEST 4, TEST 5, and TEST 6 in Table 1. Since the $K$-matrix for each of these tests contains negative eigenvalues, the first few eigenvalues of the simulated LREP of form (1.2) are purely imaginary. As shown in Fig. 2, BChebyDLR($\ell$) has no difficulty at all with the indefiniteness of $K$. This is also true when we vary the polynomial degree $m$, as shown in part of the next example (Fig. 3).

It is clear from Fig. 2 that the single vector version ($\ell = 1$) is significantly slower than its blocked counterparts ($\ell > 1$). However, when $n_{\text{want}}$ is not large, as for the shown $n_{\text{want}} = 90$ case, using a too large block size may not lead to improved efficiency.

*Example 4.3* We report the effect of the polynomial degree $m$ in BChebyDLR($\ell$) on TEST 4 and TEST 6 in Fig. 3. Results on other tests are similar thus omitted. We let $m$ vary from 8 to 38 with a stride length 3. All other parameters are fixed for these tests. Figure 3 plots the total CPU time in seconds, total number of iterations (#iter), and total number of matrix-vector products (#mvp) vs. the polynomial degree $m$.

As expected, the number of total iterations decreases as $m$ increases, while the total number of matrix-vector products does not change monotonically with $m$. The figure shows that the total #mvp is not a deciding factor on total CPU time. When $m$ is larger, the method likely uses more #mvp, but the significantly reduced #iter leads to smaller total CPU time. For these examples, the degree $m$ around 25 to 35 all appear to be cost-effective; further increasing $m$ over 40 usually does not have as significant effect as increasing $m$ when it is below 15. The least efficient case is when $m$ is too small. In comparison, the non-filtered case ($m = 1$), which in theory is

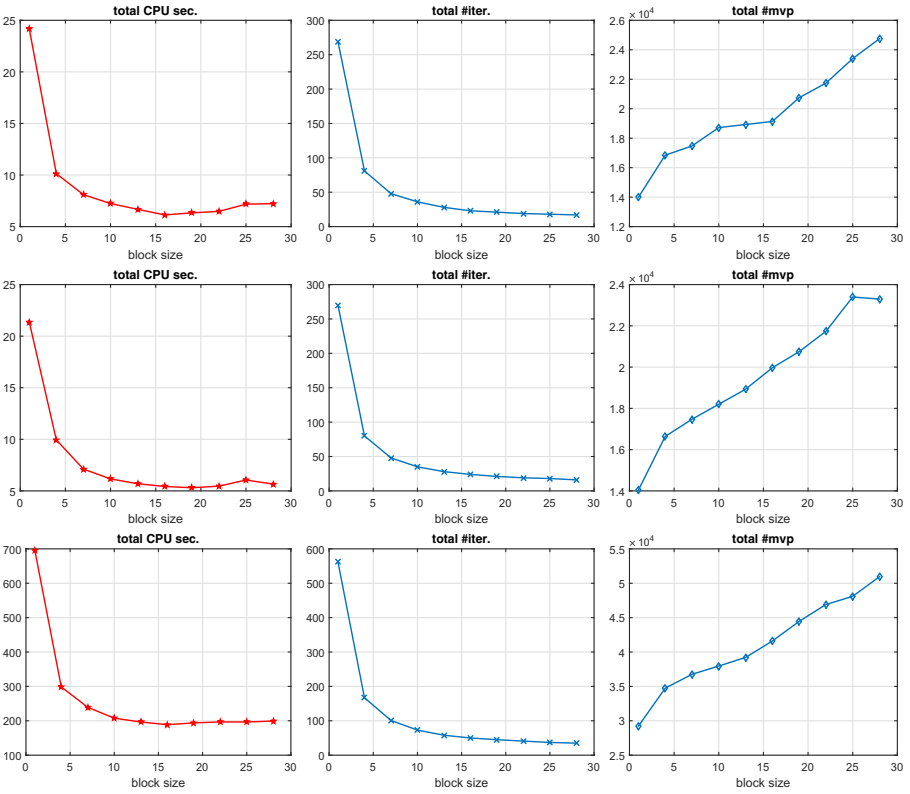



**Fig. 2** Cost in computing the first $n_{\text{want}} = 90$ eigenpairs of TEST 4 (*top*), TEST 5 (*middle*), and TEST 6 (*bottom*). The filter degree is fixed at $m = 25$. The block size $\ell$ of `BChebyDLR`($\ell$) varies from 1 to 28 with a stride length 3

equivalent to the Lanczos method [18, 21, 22], is significantly slower than the filtered method with a moderate $m$. For the non-filtered method to be competitive in speed, one would need to use a much larger dimensional subspace (a number usually much greater than $2n_{\text{want}}$), but the memory constraint can make it impractical for larger problems.

As seen from Fig. 3, it is rather easy to choose and fix an $m$ for the `BChebyDLR` method to be cost-effective, but we mention that [1] contains an interesting idea on adaptively updating $m$ during the iterations.

*Example 4.4* In the previous examples, we use the estimated $\beta$ by Algorithm 3.2 as an upper bound for $\lambda_n^2$. It works well in general. However, that $\beta$ is not guaranteed to be a true upper bound. It may be smaller than $\lambda_n^2$. Here we test the effectiveness of the adaptive strategy discussed at the end of Section 3. For this experiment, we

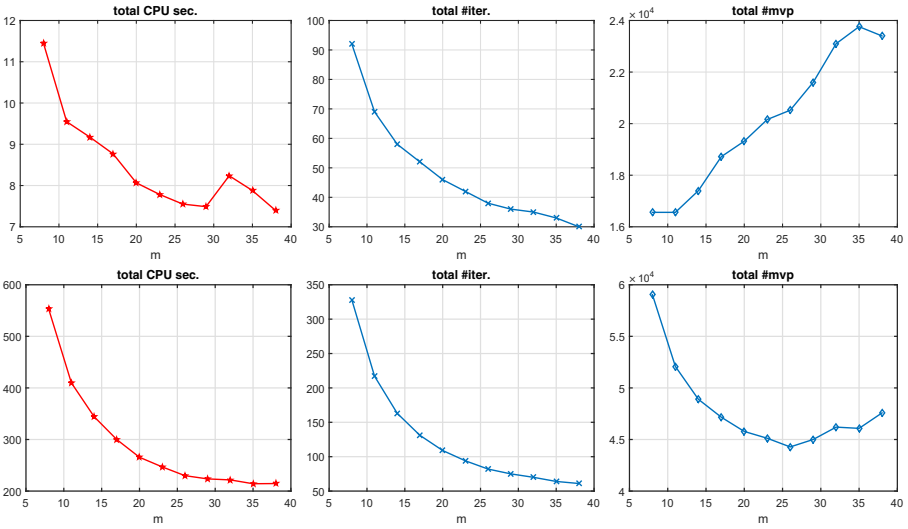



**Fig. 3** Computing the first $n_{\text{want}} = 100$ eigenpairs of TEST 4 (*top*) and TEST 6 (*bottom*). The maximum subspace dimension used is 130. The filter degree $m$ varies from 8 to 38 with a stride length 3. The block size $\ell = 10$ is used for all test runs

purposely begin with a $\beta$ that underestimates $\lambda_n^2$ so that it can be updated by the adaptive strategy. We denote the resulting algorithm as `a-BChebyDLR`. Again we use the test cases TEST 1 to TEST 6. We compare `a-BChebyDLR` and `BChebyDLR` with block size $\ell = 15$. The upper bound $\beta$ and the parameters $\alpha$ and $\alpha_0$ obtained in Algorithm 3.2 are used in `BChebyDLR`. We purposely choose an initial $\beta := \beta_a = \lambda_{\max}(T_k)$, where $T_k$ is symmetric tridiagonal matrix (3.5) computed in Algorithm 3.1, for `a-BChebyDLR`. This $\beta$ is an underestimate, since $\lambda_{\max}(T_k) \leq \lambda_n^2$ owing to Eq. (3.7). Other parameters are fixed to be the same for both schemes. We plot the number of iterations and matrix-vector products in Fig. 4. This figure shows that even with an initial filter upper bound that underestimate the largest eigenvalue, one can readily make `BChebyDLR` work properly, as demonstrated by the result of `a-BChebyDLR`. The main trick is to simply update the filter upper bound to be the largest Ritz value $\rho_{\max}$ (as defined in Eq. (3.7)), when it is found that $\rho_{\max} > \beta$.

An interesting feature of `BChebyDLR` is that the total iteration number `#iter` can be much smaller than the number of computed eigenpairs $n_{\text{want}}$, as seen in Table 2 and Figs. 2 and 3. This means that `BChebyDLR`, with suitable block size $\ell$ and degree $m$, can converge more than one eigenpair within one single iteration step. Similar effects of $m$ and $\ell$ using Chebyshev filters for different applications from LREP were also observed in [31]. The high efficiency of Chebyshev filters, as constructed in [33, 34] and also here, has attracted attention and contributed to wide spread usage of Chebyshev filter based methods, e.g., in [4, 11, 13], for large-scale DFT calculations.

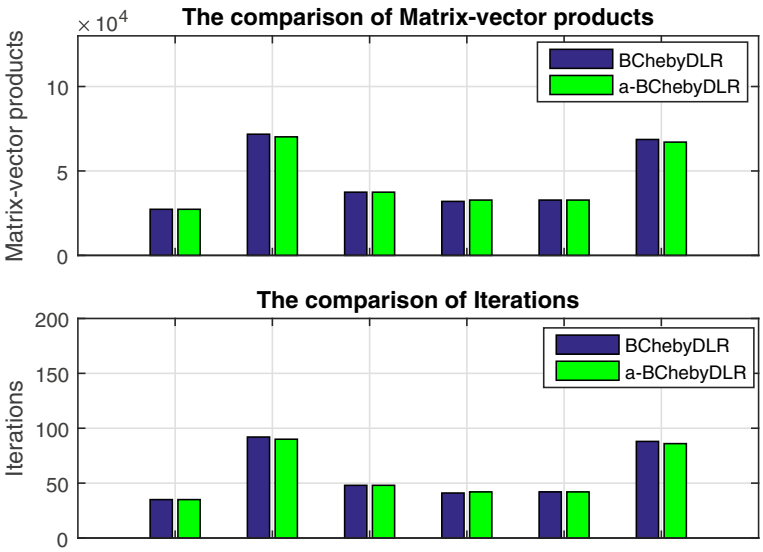



**Fig. 4** Comparison of the number of iterations and the number of matrix-vector products by `BChebyDLR` and `a-BChebyDLR` (which employs the adaptive strategy), using the same block size $\ell = 15$ and the same polynomial degree $m = 25$ to compute the first $n_{\text{want}} = 180$ eigenpairs. The performances by both are comparable

## 5 Conclusion

We propose a Chebyshev-Davidson method `BChebyDLR` for solving the linear response eigenvalue problem (LREP). This method can effectively calculate the smallest positive eigenvalues and associated eigenvectors of LREP. In Theorem 2.2 we provide a convergence rate estimate for computing the smallest eigenvalue.

Numerical examples demonstrate that `BChebyDLR` is competitive to the recently proposed `LOBP4DCG` method [2]. `BChebyDLR` needs an upper bound that is no smaller than the largest $\lambda_n^2$ in order to be effective. Although the bound computed by Algorithm 3.2 usually works well, it needs conditions as specified in Theorem 3.1 to guarantee a strict upper bound. These conditions are non-trivial to verify by computation. We propose a simple adaptive strategy that can obtain an effective upper bound during the later filtering steps in `BChebyDLR` if the bound from Algorithm 3.2 is an underestimate of $\lambda_n^2$ for the first filtering step.

We emphasize that the proposed `BChebyDLR` works for both the definite (both $K$ and $M$ are positive definite) and indefinite (one of $K$ and $M$ is indefinite) cases (see Example 4.2), while in contrast `LOBP4DCG` is applicable only to the definite LREP.

**Acknowledgments** The authors thank the anonymous referees for their constructive comments that helped improve this paper.

**Compliance with Ethical Standards** The authors confirm the compliance with the ethical standards set for the *Advances in Computational Mathematics* Journal. The authors had and have no conflict of interests. All three authors contributed equally on the design of the algorithm, implementation, testing of algorithm, and the writing and revision of this submission. The submission contains original work and is not being considered elsewhere.

## References


1. Anderson, C.R.: A Rayleigh-Chebyshev procedure for finding the smallest eigenvalues and associated eigenvectors of large sparse Hermitian matrices. J. Comput. Phys. **229**(19), 7477–7487 (2010)
2. Bai, Z., Li, R.C.: Minimization principle for linear response eigenvalue problem, I theory. SIAM J. Matrix Anal. Appl. **33**(4), 1075–1100 (2012)
3. Bai, Z., Li, R.C.: Minimization principles for the linear response eigenvalue problem II Computation. SIAM J. Matrix Anal. Appl. **34**(2), 392–416 (2013)
4. Banerjee, A.S., Elliott, R.S., James, R.D.: A spectral scheme for Kohn-Sham density functional theory of clusters. J. Comput. Phys. **287**, 226–253 (2015)
5. Cao, Z.-H., Xie, J.-J., Li, R.-C.: A sharp version of Kahan's theorem on clustered eigenvalues. Linear Algebra Appl. **245**, 147–155 (1996)
6. Cheney, E.W.: Introduction to approximation theory, 2nd edn. Chelsea Publishing Company, New York (1982)
7. Davis, T., Hu, Y.: The university of florida sparse matrix collection. ACM t. Math. Softw **38**(1), 1:1–1:25 (2011)
8. Demmel, J.W.: Applied numerical linear algebra. SIAM (1997)
9. Giannozzi, P., Baroni, S., Bonini, N., Calandra, M., Car, R., Cavazzoni, C., Ceresoli, D., Chiarotti, G.L., Cococcioni, M., Dabo, I., et al.: QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. J. Phys. Condens. Matter **21**(39), 395502 (2009)
10. Knyazev, A.V.: Convergence rate estimates for iterative methods for a mesh symmetric eigenvalue problem. Soviet J. Numer. Anal. Math. Modelling **2**(5), 371–396 (1987)
11. Levitt, A., Torrent, M.: Parallel eigensolvers in plane-wave density functional theory. Comp. Phys. Comm. **187**, 98–105 (2015)
12. Morgan, R.B.: GMRESwith deflated restarting. SIAMJ Sci. Comput. **24**(1), 20–37 (2002)
13. Motamarri, P., Gavini, V.: A subquadratic-scaling subspace projection method for large-scale Kohn-Sham density functional theory calculations using spectral finite-element discretization. Phys. Rev. B **90**, 115127 (2014)
14. Oliveira, S.: On the convergence rate of a preconditioned subspace eigensolver. Computing **63**(3), 219–231 (1999)
15. Ovthinnikov, E.: Convergence estimates for the generalized Davidson method for symmetric eigenvalue problems I: the preconditioning aspect. SIAM J. Numer. Anal. **41**(1), 258–271 (2003)
16. Ovthinnikov, E.: Convergence estimates for the generalized Davidson method for symmetric eigenvalue problems II: the subspace acceleration. SIAM J. Numer. Anal. **41**(1), 272–286 (2003)
17. Papakonstantinou, P.: Reduction of the RPA eigenvalue problem and a generalized Cholesky decomposition for real-symmetric matrices. EPL (Europhysics Letters) **78**(1), 12001 (2007)
18. Parlett, B.N.: The symmetric eigenvalue problem. Number 20 in classics in applied mathematics. SIAM, Philadelphia (1998)
19. Rocca, D.: Time-dependent density functional perturbation theory: new algorithms with applications to molecular spectra. PhD thesis, The International School for Advanced Studies, Trieste (2007)
20. Rocca, D., Bai, Z., Li, R.-C., Galli, G.: A block variational procedure for the iterative diagonalization of non-Hermitian random-phase approximation matrices. J. Chem. Phys. **136**, 034111 (2012)
21. Saad, Y.: Numerical methods for large eigenvalue problems. Wiley (1992)
22. Stewart, G.W.: Matrix algorithms, volume II: eigensystems. SIAM, Philadephia (2001)
23. Teng, Z., Li, R.-C.: Convergence analysis of Lanczos-type methods for the linear response eigenvalue problem. J. Comput. Appl. Math. **247**, 17–33 (2013)
24. Teter, M.P., Payne, M.C., Allan, D.C.: Solution of Schrödinger's equation for large systems. Phys. Rev. B **40**(18), 12255–12263 (1989)

25. Thouless, D.J.: Vibrational states of nuclei in the random phase approximation. Nucl. Phys. **22**(1), 78–95 (1961)
26. Thouless, D.J.: The quantum mechanics of Many-Body systems. Academic (1972)
27. Tsiper, E.V.: A classical mechanics technique for quantum linear response. J. Phys. B Atomic Mol. Phys. **34**(12), L401–L407 (2001)
28. Yamazaki, I., Bai, Z.J., Simon, H., Wang, L.W., Wu, K.S.: Adaptive projection subspace dimension for the thick-restart Lanczos method. ACM T. Math Software **37**(3), 27:1–27:18 (2010)
29. Zhang, L.-H., Lin, W.-W., Li, R.-C.: Backward perturbation analysis and residual-based error bounds for the linear response eigenvalue problem. BIT Numer. Math. **55**(3), 869–896 (2015)
30. Zhou, Y.: A block Chebyshev-Davidson method with inner-outer restart for large eigenvalue problems. J. Comput. Phys. **229**(24), 9188–9200 (2010)
31. Zhou, Y., Chelikowsky, J.R., Saad, Y.: Chebyshev-filtered subspace iteration method free of sparse diagonalization for solving the Kohn-Sham equation. J. Comput. Phys. **274**, 770–782 (2014)
32. Zhou, Y., Li, R.-C.: Bounding the spectrum of large Hermitian matrices. Linear Algebra Appl. **435**(3), 480–493 (2011)
33. Zhou, Y., Saad, Y.: A Chebyshev-Davidson algorithm for large symmetric eigenproblems. SIAM J Matrix Anal. Appl. **29**(3), 954–971 (2007)
34. Zhou, Y., Saad, Y., Tiago, M.L., Chelikowsky, J.R.: Parallel self-consistent-field calculations using Chebyshev-filtered subspace acceleration. Phys. Rev. E. **74**(6), 066704 (2006)