TOWARD GENUINE EFFICIENCY AND CLUSTER ROBUSTNESS OF PRECONDITIONED CG-LIKE EIGENSOLVERS

MING ZHOU AND KLAUS NEYMEYR

Abstract. The locally optimal block preconditioned conjugate gradient (LOBPCG) method is a popular solver for large and sparse Hermitian eigenvalue problems. However, recently proposed alternatives for its single-vector version LOPCG indicate certain problematic cases with less accurate preconditioners and clustered target eigenvalues. Therein LOPCG suffers from convergence delays with strongly oscillating reduction of residual norms. By interpreting LOPCG as a truncated generalized Davidson iteration, we propose utilizing some specific information from the convergence history to recover the major loss caused by truncation. Two new schemes are presented: LOPCGa, where the trial subspace is augmented and managed by an angle-based criterion; and TPCGa, which employs augmented two-term recurrences with a residual-based update strategy to reduce dependence on the threshold choice. Numerical experiments demonstrate that both schemes outperform LOPCG and its recent competitors regarding the number of required steps and the total computational time.

1. Introduction

The conjugate gradient method (CG) for solving linear systems of Hermitian positive semidefinite matrices has stimulated the development of efficient and memory-saving iterative solvers for more general non-quadratic optimization problems. In the context of computing extreme eigenvalues of large and sparse Hermitian matrices or matrix pairs, there exist various so-called CG-like eigensolvers. Two typical ones are the locally optimal block preconditioned conjugate gradient method using a CG-based three-term recurrence (LOBPCG) [12] and the Jacobi-Davidson method equipped with CG in its inner iterations where the stopping criterion depends on Rayleigh quotient residuals (JDCG) [21]. Although not revealed in the title, preconditioning is also an important feature of JDCG, and these two methods have been compared by numerical tests in a number of studies leading to somewhat disputable comments [2, 31, 23]. Ever more frequently, their convergence theory and alternative implementations are considered in further research [9, 19, 8, 33]. The present paper is motivated by the recent contributions [29, 30] concerning the single-vector version LOPCG of LOBPCG. Therein two novel methods with similar convergence behavior and memory requirement are provided together with convergence estimates in the style of a conjecture from [12]. The question arises how far the convergence analysis and the performance improvement of LOPCG can take advantage of these results. In particular, we tend to find out whether the associated estimates are meaningful in most cases and whether LOPCG can be substantially accelerated by certain non-block iterations without essential subspace expansion. An ideal acceleration by possibly adaptive vector iterations would be, after combined with deflation, superior to other eigensolvers while computing a multitude of (tightly) clustered eigenvalues.

The LOPCG method was introduced in [12] in the form of an algorithm for solving generalized eigenvalue problems of real symmetric matrices and can easily be applied to Hermitian matrices.

2010 *Mathematics Subject Classification.* Primary 65F15, 65N12, 65N25.

Key words and phrases. Hermitian eigenvalue problems, conjugate gradient, single-vector eigensolvers, cluster robustness, two-term recurrences, trial subspace augmentation. February 2026.

Here we focus on a problem setting that is mostly used in the cited references, namely,

$$Ax = \lambda Mx, \quad A, M \in \mathbb{C}^{n\times n} \text{ Hermitian}, \quad M \text{ positive definite}. \tag{1.1}$$

A more general formulation is introduced in the appendix concerning partial differential operators, Hermitian definite matrix pencils [16] and the linear response eigenvalue problem [4]. The fundamental idea of LOPCG was even earlier discussed in [10, 11]. Considering that in (1.1) the smallest eigenvalue λ_1 and an associated eigenvector x_1 are of interest, one can heuristically begin with the ideal case that λ_1 is available (if M is an identity matrix, λ_1 could be, but not necessarily efficiently, approximated by a classical implementation of the Lanczos method without determining approximate eigenvectors [6]). Thus it is possible to compute x_1 by solving the homogeneous linear system

$$A_{\lambda_1} x_1 = 0 \quad \text{for} \quad A_{\lambda_1} = A - \lambda_1 M \tag{1.2}$$

with CG thanks to the positive semidefiniteness of A_{λ_1}. The well-known CG form with two-term recurrences of iterates and search directions can be transformed into a three-term recurrence so that the new iterate $x^{(i+1)}$ depends on its predecessors $x^{(i-1)}$, $x^{(i)}$ and the residual $A_{\lambda_1}x^{(i)}$. Returning to the practical case that λ_1 is unknown, one has to use an approximate eigenvalue instead and determine the recurrence parameters in another way, e.g., implicitly by the Rayleigh-Ritz procedure as suggested in LOPCG. It is notable that for certain matrices this realistic approach converges almost as fast as the heuristic one, even if the utilized approximate eigenvalues are not close to λ_1 in the first steps [12]. In this sense, LOPCG is able to achieve a global optimality among eigensolvers with the same initial setting and memory requirement. However, its block version LOBPCG was not always the clear winner in numerical comparisons, especially for a large number of target eigenvalues with possibly small gaps [2, 31, 23]. Sometimes iterations based on two-term recurrences of CG can provide slightly better performance, and restarted generalized Davidson or Jacobi-Davidson schemes prevail at least in the case of less accurate preconditioners. Of course, this occasional suboptimality of LOBPCG is more or less problem-dependent and does not contradict its popularity and flexibility; cf. the applications [4, 17, 16, 18, 1, 28]. Recently some numerical tests directly concerning the single-vector form LOPCG are presented in [29, 30] which point out problematic cases and inspire us to accelerate LOPCG economically, e.g., with at most four-dimensional trial subspaces.

For this purpose, we need to classify the test problems with respect to the eigenvalue distribution and the preconditioning quality on the basis of the available convergence theory of LOPCG. The first-ever estimate is indeed a conjecture from [12] via the observation that expanding the trial subspace of LOPCG backwards by several more previous iterates cannot significantly accelerate the convergence. In such cases, LOPCG can be considered as an almost optimally restarted generalized Davidson iteration and, in particular, analyzed in terms of an Invert-Lanczos process when preconditioning is extremely fine [19]. The resulting bounds indicate that the convergence rate is mainly influenced by the gap $\lambda_2 - \lambda_1$ between the two smallest distinct eigenvalues. We note that the phenomenon observed in [12] is related to the harmless gap in the test problems therein. In contrast, if λ_1 and λ_2 are clustered, the eigenvalue approximation by LOPCG could have a staircase-shaped convergence behavior (cf. Figure 2.3 below) whereas Invert-Lanczos remains robust, and it would be a similar case for LOBPCG if the cluster size of target eigenvalues exceeds the block size. Furthermore, for LOPCG with less accurate preconditioners, one has to integrate certain quality parameters into the analysis. However, this does not mean a straightforward modification of the estimates for Invert-Lanczos. It is still challenging to fully derive the convergence factor suggested by the conjecture from [12]. Existing estimates are mostly derived by comparative analyses via simplified iterations such as the preconditioned steepest descent iteration (PSD) where the iterate $x^{(i-1)}$ is removed from the trial subspace of LOPCG. Among them are the sharp estimates for PSD with simple step sizes [14] or Rayleigh-Ritz step sizes [20]

where the convergence factors only depend on two or three eigenvalues and a quality parameter of preconditioning. A similar estimate from [22] additionally uses the current approximate eigenvalue to construct a dynamic convergence factor. Although these results are not sharp while applied to LOPCG, they still clearly reflect the fact that an outstanding convergence speed can be guaranteed by preconditioners well fulfilling the associated quality conditions. Summarizing the above, challenging cases for LOPCG can principally be built by selecting problems with clustered eigenvalues and less accurate preconditioners.

To improve the performance of LOPCG in problematic cases, we first construct a direct modification named LOPCGa where the trial subspace is augmented with an auxiliary vector which needs to be updated if its angle deviation from the current iterate is above a threshold. This technique dates back to an implementation of PSD with implicit deflation [5, 34] and has recently been applied to an eigensolver based on preconditioning and implicit convexity (EPIC) [29], but not utilized in a related scheme named Riemannian acceleration with preconditioning (RAP) [30]. The dimension of trial subspace reads 3 in RAP, LOPCG and 4 in EPIC, LOPCGa. According to the accompanying and our own numerical tests, EPIC is able to accelerate LOPCG with respect to the number of required steps and also the total computational time, whereas RAP is less competitive but still improves PSD, and LOPCGa is mostly more efficient than EPIC. Nevertheless, EPIC and RAP are parameter-dependent methods and can potentially be strengthened by advanced strategies in their further development. For LOPCGa, we subsequently enhance the simple update criterion of the auxiliary vector by detecting peaks in the convergence history of the residual norm, leading to even better performance. This encourages us to similarly modify the CG eigensolvers with two-term recurrences presented in [24] which are asymptotically equivalent to LOPCG.

We structure the remainder of this paper as follows: Section 2 introduces basic settings and reviews LOPCG concerning its motivation and limitation in comparison to eigensolvers with similar memory requirement including the modified method LOPCGa. Section 3 discusses the state-of-the-art estimates for LOPCG and related eigensolvers. Section 4 is devoted to appropriate update strategy of the auxiliary vector in LOPCGa and more efficient schemes with augmented two-term recurrences. Numerical tests are distributed among these sections. In addition to the briefly described CG-like eigensolvers in the above paragraph, i.e., LOPCG [12], EPIC [29], RAP [30], LOPCGa, the following schemes are also considered: PCG denotes a heuristic PCG scheme which is used to mark the optimal global convergence speed; LOPCGx denotes another augmented version of LOPCG with the directly previous iterate which however does not lead to a clear improvement; TPCGa denotes an augmented two-term PCG scheme where the augmentation is controlled by observing the residual norm reduction to reduce dependence on the threshold choice, and is superior to LOPCGa.

2. LOPCG and comparable schemes

We consider the generalized eigenvalue problem (1.1) with eigenvalues $\lambda_1 < \lambda_2 \leq \cdots \leq \lambda_n$ and focus on the numerical computation of an eigenvector x_1 associated with λ_1 by preconditioned CG-like iterations. The case $\lambda_1 = \cdots = \lambda_s < \lambda_{s+1}$ concerning multiple eigenvalues is neglected for better readability (a reformulation is straightforward by using projections onto eigenspaces). The vector iterates approximating x_1 are denoted by $x^{(i)}$ with the iteration index i. The corresponding approximate eigenvalues are determined by evaluating the Rayleigh quotient

$$\lambda : \mathbb{C}^n \backslash \{0\} \to \mathbb{R}, \quad \lambda(x) = \frac{x^* A x}{x^* M x} \tag{2.1}$$

where the asterisk * stands for the complex conjugate transpose. The value $\lambda(x^{(i)})$ is simply denoted by $\lambda^{(i)}$. In general, any Hermitian positive definite matrix $T \in \mathbb{C}^{n \times n}$ can be used as a

preconditioner and interpreted as

$$T \approx A_\sigma^{-1} \quad \text{with} \quad \kappa = \text{cond}_2(T^{1/2} A_\sigma T^{1/2}) \quad \text{for} \quad A_\sigma = A - \sigma M, \quad \sigma < \lambda_1 \tag{2.2}$$

where cond_2 denotes the spectral condition number, i.e., κ coincides with α_n/α_1 for the extreme eigenvalues α_1 and α_n of $T^{1/2} A_\sigma T^{1/2}$. The practical construction of T can utilize incomplete factorizations or approximate solutions of linear systems concerning A_σ. The so-called non-preconditioned case means that T is given by the $n \times n$ identity matrix I. We note that the non-preconditioned case can make the analysis easier only for standard eigenvalue problems (i.e. $M = I$), but not for generalized eigenvalue problems.

2.1. Benchmark method

The first scheme in our review is a heuristic benchmark suggested by Knyazev in [12, Section 3] where λ_1 is assumed to be available and the homogeneous linear system (1.2) is to be solved by the standard preconditioned CG method, e.g., in the form of Algorithm 2.1.

Algorithm 2.1: Heuristic PCG concerning the linear system (1.2)

generate $x^{(0)}$; set $r^{(0)} = -A_{\lambda_1} x^{(0)}$; $w = Tr^{(0)}$; $\gamma^{(0)} = w^* r^{(0)}$; $p^{(0)} = w$;
for $i = 0, 1, \ldots$ until convergence **do**
  $w = A_{\lambda_1} p^{(i)}$; $\quad \delta = \gamma^{(i)}/(w^* p^{(i)})$;
  $x^{(i+1)} = x^{(i)} + \delta p^{(i)}$;
  $r^{(i+1)} = r^{(i)} - \delta w$;
  $w = Tr^{(i+1)}$; $\quad \gamma^{(i+1)} = w^* r^{(i+1)}$;
  $p^{(i+1)} = w + (\gamma^{(i+1)}/\gamma^{(i)}) p^{(i)}$;

Some convergence properties have been described in [12], however, without an explicit estimate for approximate eigenvalues that can directly be compared with available estimates for related schemes. We fill this gap by adding some proofs and remarks. Therein the trivial (and practically rare) case that $x^{(0)}$ is an eigenvector associated with λ_1 or M-orthogonal to the corresponding eigenspace is neglected.

Lemma 2.1. *Consider Algorithm 2.1 with T from (2.2) and define $B = T^{1/2} A_{\lambda_1} T^{1/2}$. Let V be an arbitrary orthonormal basis matrix of the image $\text{im}(B)$ of B, then the vectors $\tilde{x}^{(i)} = V^* T^{-1/2} x^{(i)}$, $i = 0, 1, \ldots$ are CG-iterates for solving the linear system $\tilde{B}\tilde{x}_1 = 0$ for $\tilde{B} = V^* B V$.*

Proof. Noting that VV^* is the projection matrix of $\text{im}(B)$, we have $B = (VV^*)B(VV^*)$. Then the residual $r^{(i)}$ of $x^{(i)}$ can be converted to the residual $\tilde{r}^{(i)}$ of $\tilde{x}^{(i)}$ by

$$\begin{aligned} & r^{(i)} = -A_{\lambda_1} x^{(i)} \\ \Rightarrow \quad & T^{1/2} r^{(i)} = -T^{1/2} A_{\lambda_1} T^{1/2} T^{-1/2} x^{(i)} = -B T^{-1/2} x^{(i)} = -VV^* B V V^* T^{-1/2} x^{(i)} \\ \Rightarrow \quad & V^* T^{1/2} r^{(i)} = -V^* V V^* B V V^* T^{-1/2} x^{(i)} = -\tilde{B}\tilde{x}^{(i)} = \tilde{r}^{(i)}. \end{aligned}$$

Next, we check the conversions of the terms $\gamma^{(i)}$ and $Tr^{(i)}$ concerning the update of the search direction $p^{(i)}$, i.e.,

$$\begin{aligned} \gamma^{(i)} = r^{(i)*} T r^{(i)} &= (T^{1/2} r^{(i)})^* (T^{1/2} r^{(i)}) = (-VV^* B V V^* T^{-1/2} x^{(i)})^* (-VV^* B V V^* T^{-1/2} x^{(i)}) \\ &= (-V^* B V V^* T^{-1/2} x^{(i)})^* (-V^* B V V^* T^{-1/2} x^{(i)}) = \tilde{r}^{(i)*} \tilde{r}^{(i)}, \\ (V^* T^{-1/2}) T r^{(i)} &= V^* T^{1/2} r^{(i)} = \tilde{r}^{(i)}. \end{aligned}$$

Thus Algorithm 2.1 corresponds to a (non-preconditioned) CG-scheme for solving $\tilde{B}\tilde{x}_1 = 0$. □

Lemma 2.1 allows applying the well-known convergence estimates for CG since $\tilde{B}$ is evidently positive definite. Based on [26, Theorem 6.29], it holds that

$$\frac{\|\tilde{x}^{(i)}\|_{\tilde{B}}}{\|\tilde{x}^{(0)}\|_{\tilde{B}}} = \frac{\|0-\tilde{x}^{(i)}\|_{\tilde{B}}}{\|0-\tilde{x}^{(0)}\|_{\tilde{B}}} \leq \big(\mathcal{C}_i(\tilde{\varphi})\big)^{-1} \quad \text{for} \quad \tilde{\varphi} = \frac{\tilde{\beta}_m + \tilde{\beta}_1}{\tilde{\beta}_m - \tilde{\beta}_1} \tag{2.3}$$

with the Chebyshev polynomial $\mathcal{C}_i$ of degree i of the first kind and the smallest eigenvalue $\tilde{\beta}_1$ and the largest eigenvalue $\tilde{\beta}_m$ of $\tilde{B}$. Our next task is to extend (2.3) to an explicit estimate for Algorithm 2.1.

Theorem 2.2. *The approximate eigenvalues $\lambda^{(i)} = \lambda(x^{(i)})$, $i = 0, 1, \ldots$ by Algorithm 2.1 fulfill*

$$\lambda^{(i)} - \lambda_1 \leq \big(\mathcal{C}_i(\varphi)\big)^{-2} \frac{\|x^{(0)}\|_M^2}{\|x^{(i)}\|_M^2} (\lambda^{(0)} - \lambda_1) \quad \textit{for} \quad \varphi = \frac{\eta+1}{\eta-1}, \quad \eta = \kappa \frac{(\lambda_n - \lambda_1)(\lambda_2 - \sigma)}{(\lambda_2 - \lambda_1)(\lambda_n - \sigma)} \tag{2.4}$$

where κ is the condition number defined in (2.2).

Proof. The matrix $B = T^{1/2} A_{\lambda_1} T^{1/2}$ given in Lemma 2.1 is positive semidefinite and has rank $n-1$ by checking the properties of A_{λ_1} and T. Denoting its eigenvalues by $\beta_1 \leq \beta_2 \leq \cdots \leq \beta_n$, it holds that $\beta_1 = 0$, $\beta_j > 0$ for $j = 2, \ldots, n$. Moreover, since $\tilde{B} = V^* B V$ is the restriction of B to $\mathrm{im}(B)$, the extreme eigenvalues $\tilde{\beta}_1$ and $\tilde{\beta}_m$ of $\tilde{B}$ coincide with β_2 and β_n, respectively.

On the other hand, the eigenvalues of $B = T^{1/2} A_{\lambda_1} T^{1/2}$ are also those of the matrix pair (A_{λ_1}, T^{-1}) and thus related to the Rayleigh quotient

$$\beta(y) = \frac{y^* A_{\lambda_1} y}{y^* T^{-1} y}$$

which can be rewritten as the product of two other Rayleigh quotients:

$$\beta(y) = \psi(y)\,\alpha(y) \quad \text{for} \quad \psi(y) = \frac{y^* A_{\lambda_1} y}{y^* A_\sigma y} \quad \text{and} \quad \alpha(y) = \frac{y^* A_\sigma y}{y^* T^{-1} y}.$$

Correspondingly, we denote the eigenvalues of the matrix pairs $(A_{\lambda_1}, A_\sigma)$ and (A_σ, T^{-1}) by $\psi_1 \leq \psi_2 \leq \cdots \leq \psi_n$ and $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_n$ (which are all non-negative). Then $\alpha_1 \psi(y) \leq \beta(y) \leq \alpha_n \psi(y)$ holds, and the Courant-Fischer principles imply

$$\begin{aligned} \beta_2 &= \min_{\dim \mathcal{Y} = 2} \max_{y \in \mathcal{Y} \setminus \{0\}} \beta(y) \geq \alpha_1 \min_{\dim \mathcal{Y} = 2} \max_{y \in \mathcal{Y} \setminus \{0\}} \psi(y) = \alpha_1 \psi_2, \\ \beta_n &= \max_{y \in \mathbb{C}^n \setminus \{0\}} \beta(y) \leq \alpha_n \max_{y \in \mathbb{C}^n \setminus \{0\}} \psi(y) = \alpha_n \psi_n. \end{aligned}$$

Since α_1 and α_n are also the extreme eigenvalues of $T^{1/2} A_\sigma T^{1/2}$, it holds that $\kappa = \alpha_n / \alpha_1$ by (2.2). The eigenvalues of $(A_{\lambda_1}, A_\sigma)$ are those of $A_\sigma^{-1} A_{\lambda_1}$ and can be represented in terms of the eigenvalues of (A, M). By using the diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ and a matrix X consisting of associated M-orthonormal eigenvectors as columns, we get the factorization

$$A_\sigma^{-1} A_{\lambda_1} = \big(X(\Lambda - \sigma I)^{-1} X^*\big)\big(M X (\Lambda - \lambda_1 I) X^* M\big) = X(\Lambda - \sigma I)^{-1}(\Lambda - \lambda_1 I) X^{-1}$$

so that the eigenvalues of $(A_{\lambda_1}, A_\sigma)$ are given by the diagonal of $(\Lambda - \sigma I)^{-1}(\Lambda - \lambda_1 I)$, i.e., $\psi_j = (\lambda_j - \lambda_1)/(\lambda_j - \sigma)$ for $j = 1, \ldots, n$. Summarizing the above,

$$\frac{\tilde{\beta}_m}{\tilde{\beta}_1} = \frac{\beta_n}{\beta_2} \leq \frac{\alpha_n \psi_n}{\alpha_1 \psi_2} = \kappa \frac{(\lambda_n - \lambda_1)(\lambda_2 - \sigma)}{(\lambda_2 - \lambda_1)(\lambda_n - \sigma)} = \eta, \tag{2.5}$$

and the Chebyshev term in (2.3) is bounded in terms of

$$\tilde{\varphi} = \frac{\tilde{\beta}_m + \tilde{\beta}_1}{\tilde{\beta}_m - \tilde{\beta}_1} = \frac{\tilde{\beta}_m/\tilde{\beta}_1 + 1}{\tilde{\beta}_m/\tilde{\beta}_1 - 1} \geq \frac{\eta+1}{\eta-1} = \varphi > 1 \quad \Rightarrow \quad \big(\mathcal{C}_i(\tilde{\varphi})\big)^{-1} \leq \big(\mathcal{C}_i(\varphi)\big)^{-1}. \tag{2.6}$$

Next, the norm transformation (using again $B = (VV^*)B(VV^*)$)

$$\begin{aligned}\|\tilde{x}^{(i)}\|_{\tilde{B}}^2 &= \tilde{x}^{(i)*}\tilde{B}\tilde{x}^{(i)} = (V^*T^{-1/2}x^{(i)})^*(V^*BV)(V^*T^{-1/2}x^{(i)})\\ &= x^{(i)*}T^{-1/2}BT^{-1/2}x^{(i)} = x^{(i)*}T^{-1/2}(T^{1/2}A_{\lambda_1}T^{1/2})T^{-1/2}x^{(i)}\\ &= x^{(i)*}A_{\lambda_1}x^{(i)} = x^{(i)*}Ax^{(i)} - x^{(i)*}\lambda_1 Mx^{(i)} = (\lambda^{(i)}-\lambda_1)\|x^{(i)}\|_M^2\end{aligned}$$

and its counterpart for $x^{(0)}$ complete the proof by combining (2.6) with (2.3). □

Some geometric relations between $x^{(i)}$ and the target eigenspace are useful for extending Theorem 2.2.

Lemma 2.3. *Let $\mathcal{X}_1$ be the eigenspace associated with λ_1. Then the iterates $x^{(i)}$, $i = 0, 1, \ldots$ by Algorithm 2.1 fulfill*

$$x^{(i)} - x^{(0)} \perp_{T^{-1}} \mathcal{X}_1 \quad \text{and} \quad \|x^{(i)}\|_M^2 \geq \mu_1 \|x^{(0)}\|_{T^{-1}}^2 \big(\cos\angle_{T^{-1}}(x^{(0)}, \mathcal{X}_1)\big)^2 \tag{2.7}$$

where μ_1 is the smallest eigenvalue of the matrix pair (M, T^{-1}) and $\angle_{T^{-1}}$ denotes angles with respect to the inner product induced by T^{-1}.

Proof. The CG property

$$\begin{aligned}x^{(i)} &\in x^{(0)} + \operatorname{span}\{Tr^{(0)}, (TA_{\lambda_1})\,Tr^{(0)}, \ldots, (TA_{\lambda_1})^{i-1}Tr^{(0)}\}\\ &= x^{(0)} + (TA_{\lambda_1})\operatorname{span}\{x^{(0)}, (TA_{\lambda_1})\,x^{(0)}, \ldots, (TA_{\lambda_1})^{i-1}x^{(0)}\}\end{aligned}$$

ensures $x^{(i)} - x^{(0)} = TA_{\lambda_1}w$ for certain w. Thus it holds for any $x_1 \in \mathcal{X}_1\backslash\{0\}$ that

$$x_1^*T^{-1}(x^{(i)} - x^{(0)}) = x_1^*T^{-1}TA_{\lambda_1}w = (A_{\lambda_1}x_1)^*w = 0$$

which yields the orthogonality in (2.7). We further denote by x_1 the T_{-1}-projection of $x^{(0)}$ onto $\mathcal{X}_1$, then this orthogonality implies

$$\begin{aligned}\cos\angle_{T^{-1}}(x^{(0)}, \mathcal{X}_1) = \cos\angle_{T^{-1}}(x^{(0)}, x_1) &= \frac{x_1^*T^{-1}x^{(0)}}{\|x_1\|_{T^{-1}}\|x^{(0)}\|_{T^{-1}}}\\ = \frac{x_1^*T^{-1}x^{(i)}}{\|x_1\|_{T^{-1}}\|x^{(0)}\|_{T^{-1}}} &\leq \frac{\|x_1\|_{T^{-1}}\|x^{(i)}\|_{T^{-1}}}{\|x_1\|_{T^{-1}}\|x^{(0)}\|_{T^{-1}}} = \frac{\|x^{(i)}\|_{T^{-1}}}{\|x^{(0)}\|_{T^{-1}}}.\end{aligned}$$

Moreover, by noting that $\|x^{(i)}\|_M^2/\|x^{(i)}\|_{T^{-1}}^2$ is the Rayleigh quotient of the matrix pair (M, T^{-1}) evaluated at $x^{(i)}$, a lower bound is given by μ_1. Thus

$$\|x^{(i)}\|_M^2 \geq \mu_1\|x^{(i)}\|_{T^{-1}}^2 \geq \mu_1\|x^{(0)}\|_{T^{-1}}^2\big(\cos\angle_{T^{-1}}(x^{(0)}, \mathcal{X}_1)\big)^2. \qquad \square$$

The T^{-1}-orthogonality in (2.7) indicates that the T^{-1}-angle between $x^{(i)}$ and $\mathcal{X}_1$ can measure the convergence of Algorithm 2.1, whereas the inequality in (2.7) can directly be added to (2.4) to eliminate the term $\|x^{(i)}\|_M^2$.

Remark 2.4. In the special case $T = M = I$, combining (2.7) with (2.4) implies an estimate for the Lanczos method. Therein $\mu_1 = 1$, and (2.7) gives

$$\|x^{(i)}\|_2^2 \geq \|x^{(0)}\|_2^2\big(\cos\angle_2(x^{(0)}, \mathcal{X}_1)\big)^2 \tag{2.8}$$

(where the subscript $_2$ denotes Euclidean terms). Moreover, going through the proof of Theorem 2.2 with $T = I$ leads to $\eta = (\lambda_n - \lambda_1)/(\lambda_2 - \lambda_1)$. Then (2.4) becomes, by using (2.8),

$$\lambda^{(i)} - \lambda_1 \leq \big(\mathcal{C}_i(\varphi)\big)^{-2}\big(\cos\angle_2(x^{(0)}, \mathcal{X}_1)\big)^{-2}(\lambda^{(0)} - \lambda_1) \quad \text{for} \quad \varphi = \frac{\eta+1}{\eta-1} = 1 + 2\,\frac{\lambda_2 - \lambda_1}{\lambda_n - \lambda_2} \tag{2.9}$$

which differs from its traditional counterpart from [25, Theorem 2] (by changing the eigenvalue arrangement) only in terms depending on the initial guess. Subsequently, (2.9) is directly applicable to the Lanczos method since for $T = M = I$ Algorithm 2.1 generates the same series of Krylov subspaces with respect to A, and $\lambda^{(i)}$ is not smaller than its counterpart in the Lanczos

method, i.e., the smallest Ritz value associated with the current Krylov subspace. In addition, (2.9) gives a tighter bound than the traditional one due to $\lambda^{(0)}-\lambda_1 \leq (\lambda_n-\lambda_1)\big(\sin\angle_2(x^{(0)},\mathcal{X}_1)\big)^2$. Analogously, in the special case $T=A_\sigma^{-1}$, Algorithm 2.1 produces Krylov subspaces with respect to $A_\sigma^{-1}M$ so that (2.4) implies an estimate for the Invert-Lanczos method which is comparable with the one from [19]. The case $T=I$, $M\neq I$ is, however, not related to the Lanczos and Invert-Lanczos methods.

Remark 2.5. In the final phase where $\lambda^{(i)}$ is close to λ_1, Algorithm 2.1 behaves like the generalized Davidson method so that a comparative analysis is conceivable. However, combining (2.7) with (2.4) could cause a substantial overestimation for less accurate preconditioners where the sharpness depends on the choice of $\sigma<\lambda_1$ in (2.2). Although the pseudoinverse of A_{λ_1} leads to a one-step convergence, selecting σ from the near vicinity of λ_1 is not reasonable for constructing practical preconditioners. For instance, as far as A is known to be positive definite, one usually sets $\sigma=0$ so that $A_\sigma=A$ and $\kappa=\mathrm{cond}_2(T^{1/2}AT^{1/2})$; cf. [12, 29]. Moreover, the multi-step convergence factor $\big(\mathcal{C}_i(\varphi)\big)^{-2}$ for $\varphi=(\eta+1)/(\eta-1)$ can be interpreted by an asymptotic average single-step convergence factor of the form $\big((\sqrt{\eta}-1)/(\sqrt{\eta}+1)\big)^2$ based on equivalent reformulations of Chebyshev polynomials. This interpretation is problematic for the first steps by noting that the first CG-step is still a simple gradient iteration so that the convergence factor therein should be close to $\big((\eta-1)/(\eta+1)\big)^2$. We continue this discussion in Section 3.

2.2. Practical eigensolvers

The only term in Algorithm 2.1 that hinders a practical implementation is A_{λ_1} since the eigenvalue λ_1 is actually the one to be computed. A simple way out is replacing λ_1 by a readily obtainable approximation, e.g., the Rayleigh quotient value $\lambda^{(i)}=\lambda(x^{(i)})$. Of course, this replacement alone would deteriorate the convergence, since the residual and the step size need to be redefined. By considering that λ_1 is the minimum of the Rayleigh quotient (2.1), a scaled gradient can be used as the residual. Moreover, the global minimization of (2.1) can be achieved by computing Ritz values for a series of updated subspaces.

Definition 2.6. We denote by

$$x' \xleftarrow{\text{RRw}} x+\mathcal{U} \quad : \quad x'\in x+\mathcal{U} \ \text{ such that } \ \lambda(x')=\min_{w\in\mathrm{span}\{x,\mathcal{U}\}\setminus\{0\}}\lambda(w) \tag{2.10}$$

the Rayleigh-Ritz procedure aiming at the smallest Ritz value but with a weighted output with respect to the first basis vector.

By using (2.10), one can modify Lines 3 to 5 in Algorithm 2.1 as

$$x^{(i+1)} \xleftarrow{\text{RRw}} x^{(i)}+\mathrm{span}\{p^{(i)}\}; \quad r^{(i+1)}=Ax^{(i+1)}-\lambda^{(i+1)}Mx^{(i+1)};$$

to maintain the convergence toward λ_1. In addition, some other formulas for the search directions $p^{(i)}$ result in better performance [23]. This kind of modification (or the block form) is, however, not as popular as the LO(B)PCG method. We discuss its possible improvement in Section 4.

A characteristic feature of LOPCG is to determine all step sizes implicitly within the Rayleigh-Ritz procedure. In Algorithm 2.1, by combining Lines 6, 7 with Line 4 in the next step, $x^{(i+2)}$ is evidently a linear combination of $x^{(i+1)}$, $Tr^{(i+1)}$ and $p^{(i)}$. This suggests a three-dimensional trial subspace and leads to a basic version of LOPCG as shown in Algorithm 2.2. Therein the iteration index of $p^{(i)}$ is shifted, and the step sizes in Algorithm 2.1 are replaced by certain by-products for determining $x^{(i+1)}$ as a Ritz vector. The trial subspace in Algorithm 2.2 is mostly more stable than its theoretically equivalent form using $x^{(i-1)}$ instead of $p^{(i)}$. Besides this well-known fact mentioned in [12], it seems to be less known that the weighted forms $x^{(i+1)}=x^{(i)}+\cdots$ (consistent with the heuristic PCG) and $x^{(i+1)}=Tr^{(i)}+\cdots$ (suggested in [12])

without any orthogonalization could show clearly different performance. Indeed, the practical code [15, `lobpcg.m`] based on [13] uses a partial orthogonalization where $Tr^{(i)}$ is orthogonalized against $x^{(i)}$. Moreover, $p^{(i)}$ will be removed from the trial space if the underlying Gram matrix is ill-conditioned. Another suggestion for stabilization [9, 8] is to apply a full M-orthonormalization of the three basis vectors, yet this strategy increases the computational time per step and occasionally, according to our recent tests, even the number of required steps; see Section 2.3. The two-term CG-schemes from [23] seem not to suffer from such instability.

Algorithm 2.2: LOPCG for computing (λ_1, x_1) in (1.1)

```
generate x^(0); set p^(0) = 0;
for i = 0, 1, ... until convergence do
    λ^(i) = λ(x^(i));   r^(i) = Ax^(i) − λ^(i) Mx^(i);
    x^(i+1)  ←RRw−  x^(i) + span{Tr^(i), p^(i)} as in (2.10);
    p^(i+1) = x^(i+1) − x^(i);
```

The performance of LOPCG has been shown in [12, Section 7] with diagonal A and $M = I$. The preconditioner T is constructed by random diagonal matrices multiplied by orthogonal matrices. The smallest two eigenvalues are well separated: $\lambda_1 = 1$ and $\lambda_2 = 2$. The conclusion is that LOPCG performs similarly to the heuristic PCG for both ideal T with $\kappa = 4$ and less accurate T with $\kappa = 1000$ with respect to the setting (2.2) for $\sigma = 0$. It is remarkable that although the first approximate eigenvalues $\lambda^{(i)}$ are usually far from λ_1 at the beginning, the goal interval (λ_1, λ_2) can still be reached within a few steps. This also gives a reason why some results in the available convergence theory simply begin with the assumption $\lambda^{(0)} < \theta$ for certain $\theta \in (\lambda_1, \lambda_2)$. Another fact is that LOPCG significantly accelerates the preconditioned steepest descent iteration (PSD) whose trial subspace is simply $\operatorname{span}\{x^{(i)}, Tr^{(i)}\}$, analogously to the comparison between their counterparts for solving linear systems.

In a number of performance studies, LOPCG and its block extension LOBPCG are numerically compared with other practical eigensolvers, however, not always with similar memory requirement. For instance, in [31, 33], LOPCG and two-block LOBPCG are considered together with Davidson-type methods which use up to 18-dimensional trial subspaces, even for computing one extreme eigenpair. We note that in the situations where LOPCG converges dramatically slower than Davidson-type methods, its convergence behavior also significantly differs from that of the heuristic PCG. In our opinion, a fairly meaningful comparative analysis of LOPCG should focus on eigensolvers using trial subspaces with dimensions 2 to 4, e.g., the two-term CG-schemes from [23] mentioned above as well as the recently developed methods EPIC [29] and RAP [30].

We briefly introduce some features of EPIC and RAP since they are less related to the heuristic PCG, but rather derived from minimizing certain functions other than the Rayleigh quotient associated with the considered eigenvalue problem. The approach in EPIC (where T is denoted by T^{-1}) utilizes an auxiliary vector q and considers M-normalized vectors x with $q^*Mx > 0$. Then an intermediate function defined for the representations of such x underlies the further construction by applying the convex optimization theory. Apart from differences in detail, the setting for preconditioners is indeed similar to (2.2), but with respect to the Hessian of the intermediate function. The associated condition number is represented in the form L/μ and shown in [29, Corollary 2.1] to be close to the parameter η in (2.4) for $\sigma = 0$ with a distance depending on $\lambda(q) - \lambda_1$. Interestingly, the parameters μ and L are also involved in the algorithm of EPIC, and thus need to be estimated before a practical implementation. Yet the choice $\mu = L = 6$ in [29] is quite empirical, since this corresponds to a perfect preconditioner which allows a one-step convergence, but the preconditioners chosen for the numerical tests therein

are evidently less accurate ones. Nevertheless, according to our additional attempt, using the exact η as L/μ does not lead to a faster convergence than the above choice; see Section 2.3. On the other hand, the intermediate function in RAP (where T is denoted by B^{-1}) is related to the Rayleigh quotient associated with the matrix pair $(T^{1/2}AT^{1/2}, T^{1/2}MT^{1/2})$ and does not depend on auxiliary vectors. The condition number of preconditioning also uses the form L/μ and asymptotically coincides with η in (2.4); cf. [30, Corollary 16]. The parameter choice $\mu = 8$, $L = 50$ in [30] seems to be determined by a concrete numerical example, but contradicts the condition $L \geq 9\mu$ in [30, Proposition 24, Theorem 26].

In direct comparison to LOPCG, the approaches in EPIC and RAP suggest novel trial subspaces of similar dimensions (4 and 3). According to the accompanying and our own numerical tests, EPIC can be superior to LOPCG whereas RAP mostly converges slower than LOPCG but still accelerates PSD. This inspires us to construct an efficient modification of LOPCG whose trial subspace is four-dimensional just like that of EPIC. However, as reported in [12], adding directly previous iterates to the trial subspace of LOPCG does not necessarily lead to a substantial reduction of the number of required steps. We denote by LOPCGx the simplest version of this type, namely, with $\mathrm{span}\{x^{(i)}, Tr^{(i)}, p^{(i)}, x^{(i-2)}\}$ as the trial subspace, and include LOPCGx in numerical comparisons in Section 2.3.

Both of LOPCG and LOPCGx can be regarded as truncated forms of the generalized Davidson method so that an ambitious goal for improving them is to be able to select certain previous iterates that bring back the major loss caused by the truncation. Using an unchanged vector for multiple steps might be more efficient than dynamically using the directly previous iterate. Keeping this in mind, we note that the auxiliary vector q in EPIC is initialized by the initial guess $x^{(0)}$ and that EPIC is restarted with $q = x^{(i)}$ as soon as $|q^*Mx^{(i)}|$ falls below the empirical value 0.5. This restart is necessary since the first q is usually not a good approximate eigenvector as described in the theoretical construction. Although the criterion was given without explanation, it is evident that the term $|q^*Mx^{(i)}|$ is the cosine of the M-angle between $x^{(i)}$ and $\mathrm{span}\{q\}$ since the EPIC iterates are always M-normalized. Thus the essential information is that an appropriate q should keep a small angle deviation from the current iterate. The practical effectiveness of q immediately motivates our first modification of LOPCG called "augmented LOPCG" (LOPCGa) where an auxiliary vector a is added to the trial subspace and updated by the criterion $|\cos\angle_M(a, x^{(i)})| < \tau$ with a threshold $\tau = 0.7$; see Algorithm 2.3. In contrast to the active role of q in EPIC, the vector a is a simple augmentation and not used to generate alternative basis vectors of the trial subspace. In other words, LOPCGa is still structurally similar to LOPCG and the heuristic PCG.

Another important feature of LOPCGa is inspired by [13] concerning stabilization via subspace reduction. As described in Algorithm 2.3, an M-orthogonalization in the trial subspace decides whether the auxiliary vector a and the search direction $p^{(i)}$ need to be dropped in the current step. More precisely for Lines 10 and 11, we denote the diagonal entries of the corresponding (diagonal) Gram matrix W^*MW by $\delta_1, \ldots, \delta_k$ ($k \in \{3, 4\}$) and utilize a threshold $\gamma = \texttt{1e26}$. Then the trial subspace is reduced to $\mathrm{span}\{x^{(i)}, w\}$ in the case $\delta_1 > \gamma\delta_3$ or to $\mathrm{span}\{x^{(i)}, w, p^{(i)}\}$ provided that $k = 4$ and $\gamma\delta_3 \geq \delta_1 > \gamma\delta_4$. This normalization-free technique makes LOPCGa more stable than its first version using ordinary orthonormalization in the sense of smoother reduction of the residual norm in the final phase. Before attempting to construct further modifications, we report the performance of LOPCGa within a few numerical examples.

Algorithm 2.3: Augmented LOPCG (LOPCGa) for computing (λ_1, x_1) in (1.1)

generate $x^{(0)}$; set $a = x^{(0)}$;
for $i = 0, 1, \ldots$ until convergence **do**
  $\lambda^{(i)} = \lambda(x^{(i)})$; $r^{(i)} = Ax^{(i)} - \lambda^{(i)}Mx^{(i)}$; $w = Tr^{(i)}$;
  if $i > 0$ **then**
    if $|\cos \angle_M(a, x^{(i)})| < \tau$ **then**
      $a = x^{(i)}$; $\mathcal{U} = \mathrm{span}\{w,\, p^{(i)}\}$;
    else
      $\mathcal{U} = \mathrm{span}\{w,\, p^{(i)}, a\}$;
    $W \xleftarrow{M\text{-orthogonalization}} \mathrm{span}\{x^{(i)}, \mathcal{U}\}$;
    if W^*MW ill-conditioned **then**
      $\mathcal{U} = \mathrm{span}\{w\}$ or $\mathcal{U} = \mathrm{span}\{w,\, p^{(i)}\}$;
  else
    $\mathcal{U} = \mathrm{span}\{w\}$;
  $x^{(i+1)} \xleftarrow{\text{RRw}} x^{(i)} + \mathcal{U}$ as in (2.10); $p^{(i+1)} = x^{(i+1)} - x^{(i)}$;

2.3. Numerical comparisons

We select three examples to indicate that LOPCGa can significantly accelerate LOPCG when there is still space for improving it.

$$\begin{array}{lllll} \text{(a)} & A = \texttt{boneS01}, & M = I, & n = 127224, & \lambda_1 \approx \texttt{2.847268e-3}, \quad \lambda_2 \approx \texttt{6.591631e-2} \\ \text{(b)} & A = \texttt{finan512}, & M = I, & n = 74752, & \lambda_1 \approx \texttt{9.474684e-1}, \quad \lambda_2 \approx \texttt{9.502419e-1} \\ \text{(c)} & (A, M) \text{ by finite elements}, & & n = 178820, & \lambda_1 \approx \texttt{1.973967e1}, \quad \lambda_2 \approx \texttt{1.973975e1} \end{array} \tag{2.11}$$

The examples (a) and (b) listed in (2.11) are selected from [7] and also used for numerical tests in [29] where LOPCG occasionally converges clearly slower than EPIC. The example (c) is derived from a finite elements discretization of the Laplacian eigenvalue problem on the rectangle domain $[0,\, 2]\times[0,\, 1]$ with the slit $\{1\}\times[0.1,\, 0.9]$ and homogeneous Dirichlet boundary conditions. Therein the two smallest eigenvalues are tightly clustered.

For each example, we generate six preconditioners by the function `ichol` in Matlab, namely, T_1 with `nofill` and T_i for $i = 2, \ldots, 6$ with `ict` using `droptol=1e-i`. To make `ichol` feasible for (2.11, a) with larger `droptol`, we additionally apply the `symamd` permutation to (2.11, a) and exceptionally set `droptol=2e-3` for its T_2. The associated condition numbers defined in (2.2) with $\sigma = 0$ are listed in Table 2.1. In addition, we test the heuristic PCG (Algorithm 2.1 using λ_1 by `eigs`) for these preconditioners with the M-normalized `ones` vector as initial guess. The convergence history is illustrated in Figure 2.1 with respect to the Ritz value error $\lambda^{(i)} - \lambda_1$ and the (relative) residual norm $\|r^{(i)}\|_2/\|x^{(i)}\|_M$. The performance is clearly related to the conditioner numbers as analyzed in Section 2.1, apart from the fact that the convergence delay before the final phase cannot be reflected by any available estimates. In particular, the residual norm reduction can strongly oscillate for less accurate preconditioners since the input value of λ_1 still differs from the exact eigenvalue. As a way out, we update this value by the current $\lambda^{(i)}$ in the final phase, leading to weaker oscillation as seen for T_3 and T_4 in the lower left subplot. This also indicates that a practical eigensolver with unstable final phase can be refined by switching to the heuristic PCG as soon as an unusual oscillation is detected. Concerning the Ritz value error, we note that $\lambda^{(i)}$ can fall below the input value of λ_1 so that $\lambda^{(i)} - \lambda_1$ cannot be displayed by log scaling. In this case, we draw $|\lambda^{(i)} - \lambda_1|$ by dotted curves; see the upper left/right subplot. A further fact is that the convergence of $\lambda^{(i)}$ in the heuristic PCG is not necessarily monotonic

TABLE 2.1. Condition numbers of preconditioners.

Example	T_1	T_2	T_3	T_4	T_5	T_6
(2.11, a)	4.61e6	1.15e5	5.98e4	4.75e3	3.19e2	6.86e0
(2.11, b)	1.59e0	1.53e0	1.07e0	1.01e0	1.00e0	1.00e0
(2.11, c)	9.31e3	4.75e2	5.78e1	7.04e0	1.45e0	1.02e0

TABLE 2.2. Parameters for numerical comparisons.

Example, T_i	droptol	κ	η	$\beta_{\min}$	$\beta_{\max}$	seed	Figure
(2.11, a), T_4	1e-4	4.75e3	4.96e3	2.31e-4	1.15e0	20	2.2 left
(2.11, a), T_3	1e-3	5.98e4	6.25e4	2.53e-5	1.58e0	99	2.2 middle
(2.11, a), T_2	2e-3	1.15e5	1.21e5	1.31e-5	1.58e0	258	2.2 right
(2.11, b), T_3	1e-3	1.07e0	3.53e2	2.82e-3	9.94e-1	246	2.3 left
(2.11, c), T_3	1e-3	5.78e1	1.40e7	7.98e-8	1.12e0	27	2.3 middle
(2.11, c), T_2	1e-2	4.75e2	1.15e8	1.12e-8	1.29e0	171	2.3 right

in contrast to iterations using the Rayleigh-Ritz procedure. Following the above preparation, we select six combinations for numerical comparisons between practical eigensolvers; see Table 2.2.

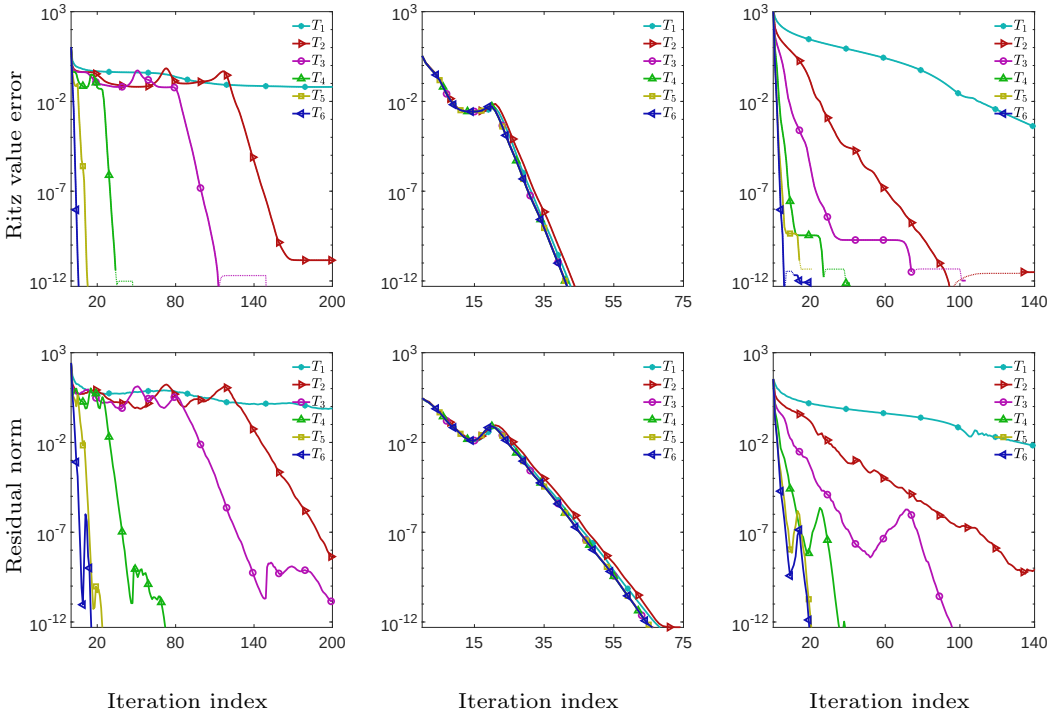

FIGURE 2.1. Selecting typical cases by observing the convergence history of the heuristic PCG (Algorithm 2.1) for examples (2.11) and preconditioners in Table 2.1.

Remark 2.7. As reported in [29, Fig. 2(b)], the example (2.11, a) is a problematic case for LOPCG whose number of required steps exceeds 1000 in a test where EPIC converges with about 500 steps. Here we consider similar cases and add more schemes to the comparison. For the applied preconditioners, Table 2.2 shows the associated `ichol` droptol values and quality parameters κ, η defined in (2.2), (2.4) for $\sigma = 0$. We additionally list the values $\beta_{\min} = \alpha_1\psi_2$ and $\beta_{\max} = \alpha_n\psi_n$ from the derivation (2.5) of η since $2\beta_{\min}$ and $2\beta_{\max}$ are close to theoretical settings of the parameters μ and L in EPIC and RAP.

We first implement LOPCG in the form of Algorithm 2.2 for various initial guesses with random number seeds among $\{0, \ldots, 300\}$. In this way we find a difficult case for LOPCG where it requires at least twice as many steps as EPIC. For completeness we also test two other versions of LOPCG. A version with full M-orthonormalization following [9] shows no visible difference to Algorithm 2.2, whereas the code [15, `lobpcg.m`] does not suffer from this troublesome case; see the left column in Figure 2.2 for the convergence history. Therein the solid curves for LOPCG, EPIC and RAP correspond to `lobpcg.m`, EPIC$(6,6)$ and RAP$(8,50)$ with empirical values of the parameters μ and L from [29, 30], and the dotted curves stand for Algorithm 2.2, EPIC$(2\beta_{\min}, 2\beta_{\max})$ and RAP$(2\beta_{\min}, 2\beta_{\max})$. Interestingly, although the empirical settings for EPIC and RAP seem inconsistent with the theoretical construction, they lead to better performance.

Remark 2.8. The performance of [15, `lobpcg.m`] (red solid curve) in the left column in Figure 2.2 is a good example for countering the stereotype "utilizing larger trial subspaces reduces the number of required steps" which is not always true for multiple steps of LOPCG and its direct algorithmic extensions like LOPCGx. The improvement by a higher dimension is only ensured in a single step provided that the current iterate is the same. In other words, the Rayleigh-Ritz procedure does not necessarily minimizes the residual norm so that the next iterate could contain ingredients that slow down further steps. Removing such ingredients is reasonable by the motto "less is more". More intuitively, by looking into the details of `lobpcg.m`, we note that its acceleration is enabled by switching to PSD (i.e., dropping $p^{(i)}$) in the 40th step where the condition number of the underlying Gram matrix exceeds a threshold. In our additional test motivated by the unusual oscillation of the residual norm around that step, a similar effective switch is made by observing the residual norm reduction. Without this switch, `lobpcg.m` would behave like Algorithm 2.2 (red dotted curve), whereas the simple extension LOPCGx of Algorithm 2.2 (by adding $x^{(i-2)}$ to the trial subspace) only enables a slight acceleration.

Furthermore, the adaptive extension LOPCGa (Algorithm 2.3) begins to accelerate only a few steps later than EPIC and gets a much better convergence rate in the final phase. The advantage of LOPCGa is more evident for the residual norm. We also illustrate the heuristic PCG (Algorithm 2.1) to indicate the space for improvement in further modifications. Another mentionable fact is that the code [15, `lobpcg.py`] in Python uses a simple residual criterion instead of the condition number criterion in [15, `lobpcg.m`]. However, the threshold therein is rather suitable for detecting a much stronger oscillation and thus less effective for the above example. Numerical tests in Python concerning LOPCG, LOPCGa and the final product TPCGa of this paper will be discussed at the end of Section 4.

The middle column in Figure 2.2 stands for a comparison where the initial guess is selected such that the convergence speed of LOPCG is average among random tests. In this case, the code `lobpcg.m` and Algorithm 2.2 have quite similar performance (so that the red dotted curve is covered) and only slightly differ from LOPCGx. Moreover, EPIC$(6,6)$ is still competitive and superior to EPIC$(2\beta_{\min}, 2\beta_{\max})$, but slower than LOPCG in the final phase. The two implementations of RAP behave like PSD, but RAP$(2\beta_{\min}, 2\beta_{\max})$ becomes the better version. The benefit of LOPCGa is substantial in the sense of closeness to the heuristic PCG.

The right column in Figure 2.2 is concerned with an even less accurate preconditioner and the worst case of LOPCG among random tests. We note that the number of required steps of EPIC$(6,6)$ is comparable with that in [29, Fig. 2(b)]. The two versions of LOPCG are not dramatically slower than EPIC$(6,6)$ and not accelerated by LOPCGx, whereas LOPCGa is again the closest one to the heuristic PCG. The other schemes are far less competitive.

Remark 2.9. Following [13] and [15, `lobpcg.m`], our implementations of LOPCG (Algorithm 2.2), LOPCGx and LOPCGa take use of implicit updates of the contained matrix-vector products

TABLE 2.3. Computational time per step of CG-like eigensolvers by using that of the heuristic PCG as the unit of measure.

Example, T_i	RAP		EPIC		LOPCG		LOPCGx	LOPCGa
$(2.11, \mathrm{a})$, T_4	1.36	(1.24)	1.44	(1.34)	1.18	(1.05)	1.07	1.06
$(2.11, \mathrm{a})$, T_3	1.62	(1.40)	1.70	(1.48)	1.26	(1.06)	1.08	1.08
$(2.11, \mathrm{a})$, T_2	1.77	(1.50)	1.85	(1.56)	1.31	(1.12)	1.13	1.12
$(2.11, \mathrm{b})$, T_3	2.83	(2.67)	2.94	(2.71)	2.03	(1.38)	1.49	1.44
$(2.11, \mathrm{c})$, T_3	2.14	(2.05)	2.22	(2.12)	1.95	(1.09)	1.19	1.13
$(2.11, \mathrm{c})$, T_2	2.61	(2.51)	2.73	(2.59)	2.31	(1.15)	1.33	1.21

so that each step (after initialization) only performs three explicit matrix-vector multiplications (MVMs) with large matrices, namely, $Tr^{(i)}$, Aw and Mw where w denotes $Tr^{(i)}$ or its M-orthogonalization against $x^{(i)}$. Further matrix-vector products are determined via `axpy`. In this sense, the main expense per step is of the same order as in the heuristic PCG (Algorithm 2.1). However, the concrete expense still varies due to further algorithmic details; see the comparison in Table 2.3. Therein the expense of each considered scheme is given by the ratio of its (average) computational time per step to that of the heuristic PCG (i.e., the unit of measure). In the column for LOPCG, we first list the expense of `lobpcg.m` and then add the expense of Algorithm 2.2 in parentheses. In comparison to the heuristic PCG, the extra expense in Algorithm 2.2 is essentially caused by its more `axpy` terms in the Rayleigh-Ritz procedure and thus reflects the dominance of the three explicit MVMs. The code `lobpcg.m` does not provide a tailored form for single-vector iterations so that some block-oriented subroutines further increase the expense, especially for the example $(2.11, \mathrm{c})$ with $M \neq I$. In contrast, LOPCGx and LOPCGa merely need to determine a few more `axpy` terms than Algorithm 2.2 and are thus only slightly more expensive. The column for RAP contains the expense of the code from [30] where the trial subspace is orthonormalized by the QR decomposition and more explicit MVMs are performed for stability. We additionally modify this code such that only three explicit MVMs remain, and list the reduced expense in parentheses. The column for EPIC is established in an analogous manner. The relatively high expenses of EPIC and RAP are apparently caused by their complicated construction of trial subspaces depending on more intermediate vectors.

Figure 2.3 shows comparisons for the examples $(2.11, \mathrm{b}, \mathrm{c})$ concerning clustered eigenvalues and the worst case of LOPCG among random tests. It is remarkable that LOPCGx becomes more robust and mostly accelerates LOPCG. Moreover, EPIC$(6, 6)$ is still the best one among the versions of EPIC and RAP or, in more detail, EPIC mostly benefits from empirical settings, but RAP rarely. Finally, the pleasing efficiency of LOPCGa motivates us to investigate whether it can be further improved by more reliable augmentation strategies.

Remark 2.10. The readers might be curious whether the threshold $\tau = 0.7$ is always the best choice in LOPCGa. Actually this value is selected in the example $(2.11, \mathrm{a})$ with T_4 by observing the performance of LOPCGa for $\tau \in \{0.1, 0.2, \ldots, 0.9\}$. We consistently demonstrate LOPCGa with $\tau = 0.7$ in the above numerical tests since this implementation already improves the other schemes. For completeness, we report in Table 2.4 the performance of LOPCGa with various τ values for all examples in terms of the number of required steps concerning the stopping criterion $\|r^{(i)}\|_2/\|x^{(i)}\|_M \leq$ `1e-12`. We note that none of the considered τ values is universally optimal, and $\tau = 0.7$ is very effective in three out of six cases. Another interesting question is whether using more augmentation vectors reduces the number of required steps. Therefore we extend LOPCGa such that up to k most recently suggested augmentation vectors are added to the trial subspace. Following the comparison in Table 2.4, the extended LOPCGa with $\tau = 0.7$ and $k = 2$ converges within (156, 417, 523, 88, 159, 831) steps and thus improves LOPCGa with $\tau = 0.7$

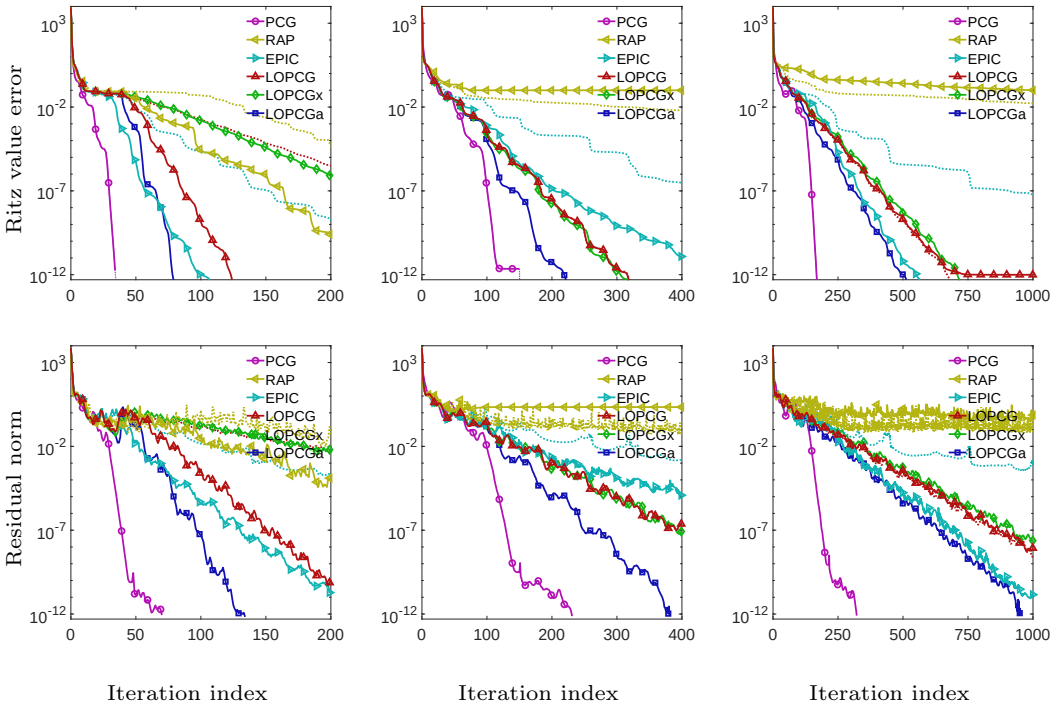

FIGURE 2.2. Numerical comparisons between CG-like eigensolvers for the example (2.11, a); see Section 2.3 for details. The subplots include three cases where LOPCG implementations are entirely/not/partially inferior to EPIC implementations. The augmented scheme LOPCGa (Algorithm 2.3) is clearly more efficient.

TABLE 2.4. Number of required steps of LOPCGa for reaching $\|r^{(i)}\|_2/\|x^{(i)}\|_M \leq$ `1e-12` with various values of τ.

Example, T_i	τ=0.1	τ=0.2	τ=0.3	τ=0.4	τ=0.5	τ=0.6	τ=0.7	τ=0.8	τ=0.9
(2.11, a), T_4	196	201	224	148	234	157	134	164	198
(2.11, a), T_3	399	652	440	619	510	594	381	536	424
(2.11, a), T_2	>1000	845	>1000	574	851	753	944	811	537
(2.11, b), T_3	134	87	87	189	87	184	88	130	178
(2.11, c), T_3	187	566	>1000	171	300	137	223	156	982
(2.11, c), T_2	>1000	832	965	824	886	835	854	868	470

in its worse cases. More tests with larger k do not show significant further improvements. In particular for (2.11, c), T_2, it seems to be difficult to break the record 470 by enlarging k without changing $\tau = 0.7$. This observation enhances our motivation for reducing dependence on the threshold choice.

3. Open issues in the convergence theory of LOPCG

Despite the fairly early beginning of convergence analyses of CG-like eigensolvers including a concept of LOPCG [10, Section 1.5], direct estimates are only available for certain schemes related to LOPCG, i.e., on the basis of comparative investigations. The first-ever scheme is called an abstract two-stage method [10, Section 3.2] and has with our settings the form

$$x^{(i+1)} = f\big(T(A - \lambda^{(i)}M)\big)\, x^{(i)} \tag{3.1}$$

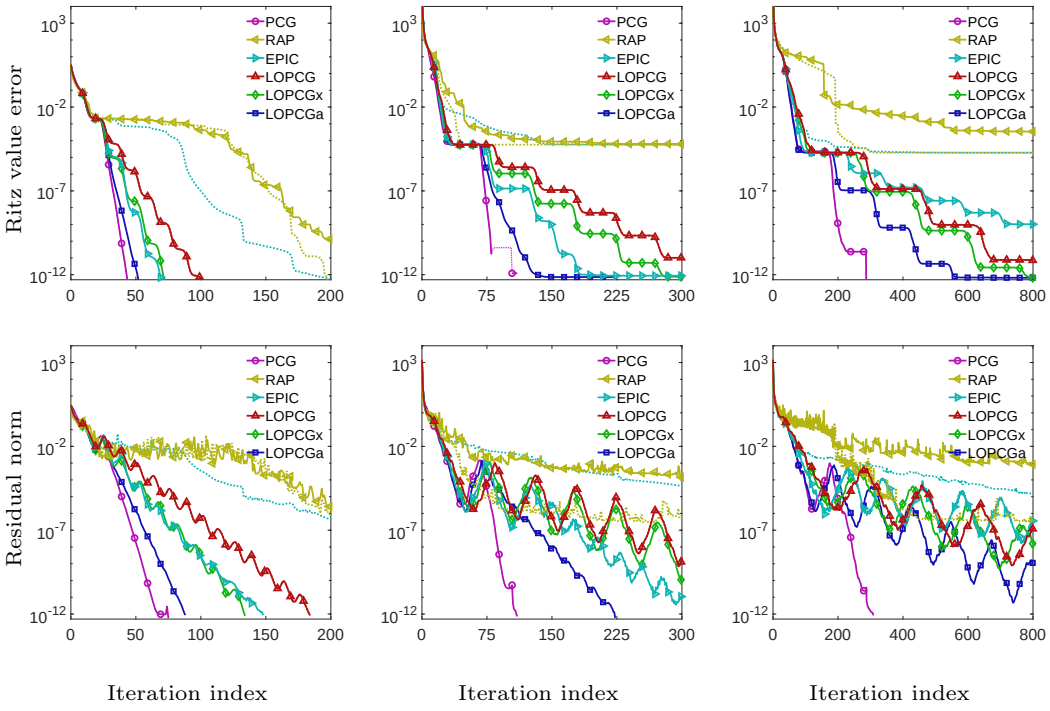

FIGURE 2.3. Numerical comparisons between CG-like eigensolvers for the examples (2.11, b, c); see Section 2.3 for details. The subplots particularly indicate the cluster robustness of LOPCGa (Algorithm 2.3) in the sense that the residual norm reduction is straightforward or oscillates more weakly.

where f denotes an abstract matrix function. Although this scheme does not really cover LOPCG due to the incompatible CG-term $p^{(i)}$, it corresponds to the weaker scheme PSD (i.e. LOPCG without $p^{(i)}$) by specifying f as a linear polynomial so that indirect pessimistic estimates for LOPCG can be obtained. Moreover, if f is specified as a shifted Chebyshev polynomial, (3.1) can be interpreted as an acceleration of PSD or a Lanczos-type scheme. This asymptotically matches multiple steps of the generalized Davidson method in the final phase where $\lambda^{(i)}$ only varies slightly, or of the heuristic PCG for sufficiently small $\lambda^{(i)}-\lambda_1$, leading to indirect optimistic estimates for LOPCG. The framework of such estimates reads

$$\frac{\lambda^{(i+1)}-\lambda_1}{\lambda_2-\lambda^{(i+1)}} \le \xi \frac{\lambda^{(i)}-\lambda_1}{\lambda_2-\lambda^{(i)}} \tag{3.2}$$

under the assumption $\lambda^{(i)} < \lambda_2$. The convergence factor ξ can be given in a cumbersome form involving $\lambda^{(i)}$ or in a concise form without $\lambda^{(i)}$. In particular, ξ in the Chebyshev specification of (3.1) is comparable with $\big(\mathcal{C}_i(\varphi)\big)^{-2}$ from the estimate (2.4) for the heuristic PCG.

Nevertheless, using optimistic estimates is sometimes problematic, especially in the troublesome cases for LOPCG mentioned in Section 2.3. In addition, deriving reasonable pessimistic estimates for LOPCG, e.g., via sharp estimates for PSD with Rayleigh-Ritz step sizes, was technically more challenging so that the analysis in [10, Section 3.4] actually arrived at a weaker convergence factor which can be written as $\xi = 1-\eta^{-1}$ by using the condition number η defined in (2.4). The next notable estimate for PSD is presented in [22, Section 6] with ξ depending on $\lambda^{(i)}$, whereas the final form $\xi = \big((\eta-1)/(\eta+1)\big)^2$ expected in [10] was fully proved in [20] after a number of investigations of PSD with simple step sizes [14]. Here we reformulate the main result from [20] for further discussion.

Lemma 3.1. *Consider an arbitrary $x \in \mathbb{C}^n\backslash\{0\}$ that is not an eigenvector of (A, M) and let θ' be the minimum of the Rayleigh quotient (2.1) in $\operatorname{span}\{x, T(Ax - \theta\, Mx)\}$ for $\theta = \lambda(x)$. Then $\theta' < \theta$, and*

$$\frac{\theta' - \lambda_j}{\lambda_{j+1} - \theta'} \leq \xi_j \frac{\theta - \lambda_j}{\lambda_{j+1} - \theta} \quad \textit{for} \quad \xi_j = \Big(\frac{\eta_j - 1}{\eta_j + 1}\Big)^2, \quad \eta_j = \kappa \frac{(\lambda_n - \lambda_j)(\lambda_{j+1} - \sigma)}{(\lambda_{j+1} - \lambda_j)(\lambda_n - \sigma)} \tag{3.3}$$

with respect to the setting (2.2) for preconditioners provided that $\lambda_j \leq \theta < \lambda_{j+1}$.

The original formulation in [20] describes the preconditioner T with a parameter corresponding to $(\kappa - 1)/(\kappa + 1)$ by (2.2). The analysis therein is motivated by accelerating a perturbed shift-and-invert eigensolver rather than comparing PSD with a gradient iteration for solving the homogeneous linear system (1.2).

Remark 3.2. Applying Lemma 3.1 to PSD by considering $\theta = \lambda^{(i)}$ and $\theta' = \lambda^{(i+1)}$ indicates that the vector iterates can at least reach an eigenvector associated with an interior eigenvalue λ_j where a stagnation at λ_j without eigenvector convergence is excluded by the simple inequality $\theta' < \theta$. The convergence toward λ_1 cannot be ensured by Lemma 3.1 for arbitrary T from (2.2) without further assumptions due to the fact that there exist examples fulfilling the equality in (3.3). A technical reason is that the underlying analysis is eigenvalue-oriented and related to a three-dimensional subspace concerning the slowest convergence. An exception is the special case $T = A_\sigma^{-1}$ where the convergence toward λ_1 can be shown on the basis of angle estimates like [10, Theorem 1.2]. Furthermore, Lemma 3.1 provides intermediate bounds for LOPCG whose trial subspace is a superset of that of PSD so that inequalities between eigenvalue approximations are obtained by Courant-Fischer principles. In contrast, angle estimates for PSD are not (directly) applicable to LOPCG.

Remark 3.3. The convergence factor ξ_j in (3.3) does not depend on any iterates so that the estimate can easily be applied to multiple steps. In particular, (3.3) results in the following estimate concerning the final phase of PSD with $\lambda^{(i)} < \lambda_2$:

$$\frac{\lambda^{(i+m)} - \lambda_1}{\lambda_2 - \lambda^{(i+m)}} \leq \xi^m \frac{\lambda^{(i)} - \lambda_1}{\lambda_2 - \lambda^{(i)}} \quad \text{for} \quad \xi = \Big(\frac{\eta - 1}{\eta + 1}\Big)^2, \quad \eta = \kappa \frac{(\lambda_n - \lambda_1)(\lambda_2 - \sigma)}{(\lambda_2 - \lambda_1)(\lambda_n - \sigma)}. \tag{3.4}$$

With the same settings, the first-ever convergence estimate for PSD by Samokish [27] can be rewritten as

$$\lambda^{(i+1)} - \lambda_1 \leq \xi(\lambda^{(i)} - \lambda_1)\big(1 + \mathcal{O}(\sqrt{\lambda^{(i)} - \lambda_1}\,)\big), \tag{3.5}$$

whereas (3.4) with $m = 1$ implies

$$\lambda^{(i+1)} - \lambda_1 \leq \xi(\lambda^{(i)} - \lambda_1)\Big(1 + \frac{(\lambda^{(i)} - \lambda_1) - (\lambda^{(i+1)} - \lambda_1)}{(\lambda_2 - \lambda_1) - (\lambda^{(i)} - \lambda_1)}\Big) = \xi(\lambda^{(i)} - \lambda_1)\big(1 + \mathcal{O}(\lambda^{(i)} - \lambda_1)\big)$$

containing a slightly better asymptotic term. In contrast to this, the non-asymptotic estimate $\lambda^{(i+1)} - \lambda_1 \leq \xi(\lambda^{(i)} - \lambda_1)$ does not hold in general and requires a larger convergence factor instead of ξ; see [22, Theorems 2.1 and 6.2] with convergence factors depending on $\lambda^{(i)}$. It is still difficult to derive a multi-step estimate like $\lambda^{(i+m)} - \lambda_1 \leq \tilde{\xi}^m(\lambda^{(i)} - \lambda_1)$ without using $\lambda^{(i)}$ to build an appropriate convergence factor $\tilde{\xi}$. Nevertheless, the asymptotic estimate (3.5) can be upgraded for the restarted generalized Davidson method as mentioned in [24, Eq. (5.9)]. This corresponds to

$$\lambda^{(i+m)} - \lambda_1 \leq \big(\mathcal{C}_m(\varphi)\big)^{-2}(\lambda^{(i)} - \lambda_1)\big(1 + \mathcal{O}(\sqrt{\lambda^{(i)} - \lambda_1}\,)\big) \quad \text{for} \quad \varphi = \frac{\eta + 1}{\eta - 1} \tag{3.6}$$

with the settings for (3.4) and the Chebyshev polynomial $\mathcal{C}_m$. It should be noted that (3.6) also requires the assumption $\lambda^{(i)} < \lambda_2$ and cannot directly be derived on the basis of the similar estimate (2.4) for the heuristic PCG. Furthermore, (3.4) with $\big(\mathcal{C}_m(\varphi)\big)^{-2}$ instead of ξ^m

corresponds to the conjectured estimate for LOPCG in [12]. The multi-step convergence factor $\big(\mathcal{C}_m(\varphi)\big)^{-2}$ can also be transformed into

$$\big(\mathcal{C}_m(\varphi)\big)^{-2} = \Big(\frac{2\psi^m}{1+\psi^{2m}}\Big)^2 \quad \text{for} \quad \psi = \frac{\sqrt{\eta}-1}{\sqrt{\eta}+1} \tag{3.7}$$

and then interpreted by the average single-step convergence factor ψ^2. However, this does not mean that replacing ξ by ψ^2 in (3.5) provides a suitable single-step estimate for LOPCG. In Figure 3.1, we compare LOPCG with PSD and the non-restarted generalized Davidson method (GD) concerning the first, fourth and fifth cases from Table 2.2. The Ritz value error $\lambda^{(i)}-\lambda_1$ is illustrated in the upper row where the two horizontal lines denote $\lambda_2-\lambda_1$ and $(\lambda_2-\lambda_1)/2$. The lower row shows the associated convergence factor $(\lambda^{(i+1)}-\lambda_1)/(\lambda^{(i)}-\lambda_1)$ together with two horizontal lines corresponding to ξ and ψ^2 defined in (3.4) and (3.7). Obviously, each method is slowed down as soon as $\lambda^{(i)}$ reaches λ_2. At this moment, the convergence factor is "reset" to ξ even for GD and causes a stagnation. This phenomenon can repeatedly occur during further stagnations of LOPCG so that its convergence rate cannot simply be bounded by ψ^2. On the other hand, ψ^2 does not provide an appropriate estimation for the convergence acceleration of GD or LOPCG out of stagnations, especially for less accurate preconditioners and clustered eigenvalues where ψ^2 is close to ξ; cf. the left and right columns in Figure 3.1.

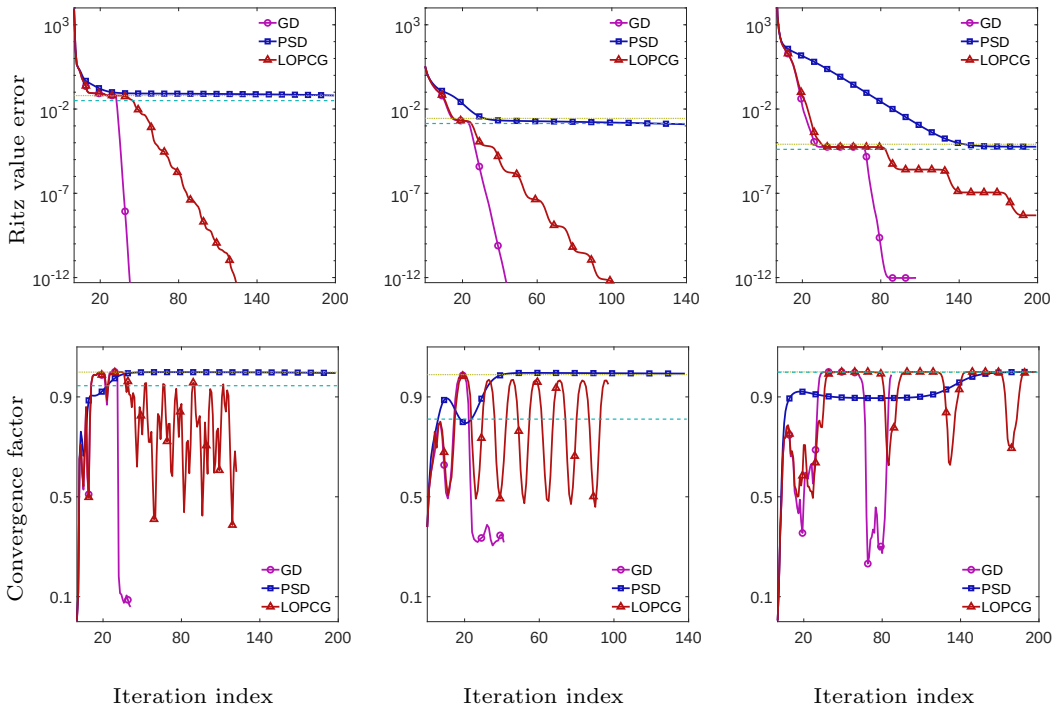

FIGURE 3.1. Convergence phenomena beyond the available estimates for CG-like eigensolvers (Remark 3.3). Upper row: convergence history of $\lambda^{(i)}-\lambda_1$ in comparison to $\lambda_2-\lambda_1$ and $(\lambda_2-\lambda_1)/2$ (horizontal lines) which roughly mark the beginning of the final phase. Lower row: single-step convergence factor $(\lambda^{(i+1)}-\lambda_1)/(\lambda^{(i)}-\lambda_1)$ in comparison to the estimated values ξ and ψ^2 (horizontal lines) from (3.4) and (3.7).

Remark 3.4. The recent studies [29, 30] related to the convergence theory of LOPCG present estimates for two alternative methods EPIC and RAP. We restate these results with the settings

for (3.4) for a brief discussion. The estimate [29, Eq. (5.15)] for EPIC (after correcting a typo) corresponds to

$$\lambda^{(i+m)} - \lambda_1 \leq 2\big(1 - \sqrt{\tilde{\eta}^{-1}}\big)^m (\lambda^{(i)} - \lambda_1) \quad \text{for} \quad \tilde{\eta} = \eta + \mathcal{O}\big(\lambda(q) - \lambda_1\big) \tag{3.8}$$

depending on an auxiliary vector q, and the estimate [30, Eq. (3)] for RAP can be written as

$$\lambda^{(i+m)} - \lambda_1 \leq 2\Big(1 - \frac{1}{2}\sqrt{\tilde{\eta}^{-1}}\Big)^m (\lambda^{(i)} - \lambda_1) \quad \text{for} \quad \tilde{\eta} \approx \kappa\, \frac{\lambda_2 - \sigma}{\lambda_2 - \lambda_1} \tag{3.9}$$

where the exact definition of $\tilde{\eta}$ requires an auxiliary scalar ρ between λ_1 and $(\lambda_1 + \lambda_2)/2$. The estimates (3.8) and (3.9) can easily be compared with (3.6) by using the transformation (3.7) and neglecting the auxiliary terms. The convergence factor in (3.6) is clearly smaller than those in (3.8) and (3.9) and even than the corresponding squared values. In comparison to (3.8), the added term $\frac{1}{2}$ in (3.9) is problematic in the sense that the single-step convergence factor is at least $\frac{1}{2}$, leading to overestimations for well-separated eigenvalues and more precise preconditioners. Moreover, (3.8) and (3.9) are weaker asymptotic estimates since their auxiliary terms cannot be interpreted as monotonic decreasing like $\mathcal{O}(\sqrt{\lambda^{(i)} - \lambda_1})$, e.g., the last auxiliary vector q of EPIC is mostly chosen before the final phase due to [29, Fig. 2].

Remark 3.5. The comparison in Remark 3.4, especially the exponent difference between the convergence factors, indicates that LOPCG cannot thoroughly be investigated by comparative approaches concerning more general problem settings. We further note that some potential drawbacks of applying the theory of non-quadratic CG minimization to CG-like eigensolvers have been pointed out in [24, Section 4], followed by the derivation of a direct two-step estimate for LOPCG on the basis of asymptotically equivalent two-term CG-schemes. The explicit estimate essentially coincides with the special form of (3.6) for $m = 2$ and can indeed be reconstructed by considering that the decelerated form

$$x^{(i+1)} \xleftarrow{\text{RRw}} x^{(i)} + \operatorname{span}\{Tr^{(i)}, p^{(i)}\}; \quad x^{(i+2)} \xleftarrow{\text{RRw}} x^{(i+1)} + \operatorname{span}\{Tr^{(i+1)}\}; \tag{3.10}$$

of LOPCG in its every second step corresponds to a two-step restarted version of the generalized Davidson method. Furthermore, [24, Theorem 4.1] provides a remarkable asymptotic estimate for LOPCG: in the final phase with $\lambda^{(i)} < (\lambda_1 + \lambda_2)/2$, it holds that

$$(\lambda^{(i)} - \lambda^{(i+1)})^{-1} + (\lambda^{(i+1)} - \lambda^{(i+2)})^{-1} = (\lambda^{(i+1)} - \tilde{\lambda}^{(i+2)})^{-1}\big(1 + \mathcal{O}(\sqrt{\lambda^{(i)} - \lambda_1}\,)\big) \tag{3.11}$$

with the approximate eigenvalue $\tilde{\lambda}^{(i+2)}$ given by one PSD step applied to $x^{(i+1)}$. The explicit estimate in [24, Theorem 4.2] is based on

$$\frac{\lambda^{(i+1)} - \lambda_1}{\lambda^{(i)} - \lambda^{(i+1)}} + \frac{\lambda^{(i+2)} - \lambda_1}{\lambda^{(i+1)} - \lambda^{(i+2)}} = \frac{\tilde{\lambda}^{(i+2)} - \lambda_1}{\lambda^{(i+1)} - \tilde{\lambda}^{(i+2)}} + \mathcal{O}(\sqrt{\lambda^{(i)} - \lambda_1}\,) \tag{3.12}$$

which is a transformation of (3.11), namely, both sides of (3.11) are multiplied by $\lambda^{(i+1)} - \lambda_1$ and then decreased by 1. However, this approach is somewhat problematic for clustered eigenvalues. In Figure 3.2, we consider again the first, fourth and fifth cases from Table 2.2 together with the form $\delta = \mathcal{O}(\sqrt{\lambda^{(i)} - \lambda_1}\,)$ associated with the above estimates. The corresponding δ_1 for (3.11) and δ_2 for (3.12) are compared with $\sqrt{\lambda^{(i)} - \lambda_1}$. In contrast to the left subplot with well-separated eigenvalues, the other two subplots indicate that δ_1 and δ_2 can clearly exceed $\sqrt{\lambda^{(i)} - \lambda_1}$ after several stagnations caused by clustered eigenvalues. In addition, we note that the approach from [24] could be extended by using

$$\delta_3 = \big((\lambda^{(i)} - \lambda_1)^{-1/2} + (\lambda^{(i+2)} - \lambda_1)^{-1/2} - 2(\tilde{\lambda}^{(i+2)} - \lambda_1)^{-1/2}\big)^{-1}$$

via three-term recurrences of Chebyshev polynomials to achieve the form (3.6). Nevertheless, also δ_3 is not comparable with $\sqrt{\lambda^{(i)} - \lambda_1}$ for clustered eigenvalues as seen in Figure 3.2.

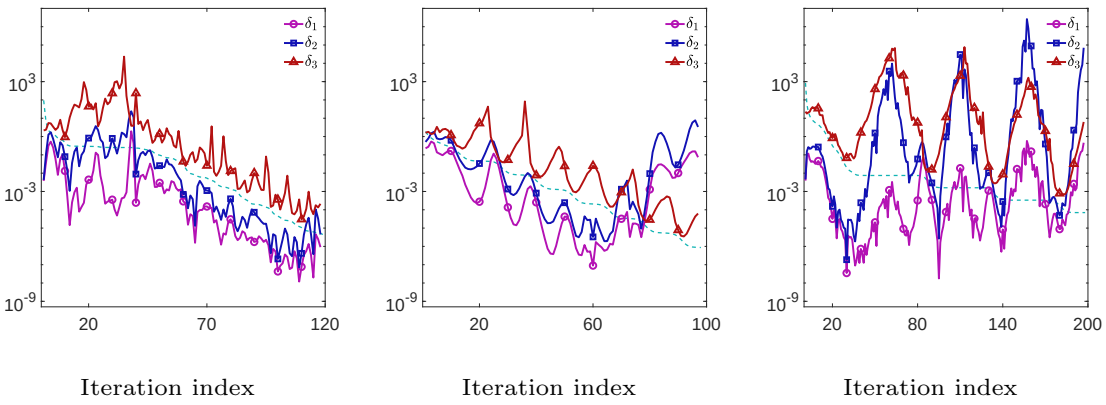

FIGURE 3.2. Limitation of asymptotic estimates for LOPCG (Remark 3.5). The asymptotic terms δ_1, δ_2, δ_3 for (3.11), (3.12), (3.6) can substantially deviate from the estimated size $\mathcal{O}(\sqrt{\lambda^{(i)}-\lambda_1}\,)$ for clustered eigenvalues (middle and right subplots).

Summarizing the above, the existing estimates are meaningful for LOPCG concerning non-clustered eigenvalues, yet the staircase-shaped convergence behavior of LOPCG in the case $\lambda_1 \approx \lambda_2$ still cannot be fully explained due to $\big(\mathcal{C}_m(\varphi)\big)^{-2} \approx 1$. In particular, the acceleration between two delays needs to be investigated by some other approaches in future research.

4. Two-term PCG (TPCG) schemes

We turn attention to the two-term alternatives for LOPCG from [23, 24] which can be formulated as direct modifications of Algorithm 2.2 as in Algorithm 4.1.

Algorithm 4.1: TPCG for computing (λ_1, x_1) in (1.1)

1 generate $x^{(0)}$; set $p^{(0)} = 0$;
2 **for** $i = 0, 1, \ldots$ until convergence **do**
3 $\quad \lambda^{(i)} = \lambda(x^{(i)})$; $\quad r^{(i)} = Ax^{(i)} - \lambda^{(i)} M x^{(i)}$;
4 $\quad$ select $\iota^{(i)}, \tau^{(i)}$; $\quad p^{(i+1)} = \iota^{(i)}\, Tr^{(i)} + \tau^{(i)} p^{(i)}$;
5 $\quad x^{(i+1)} \xleftarrow{\text{RRw}} x^{(i)} + \operatorname{span}\{p^{(i+1)}\}$ as in (2.10);

Therein the essential change is that the update of the search direction $p^{(i)}$ does not depend on the Rayleigh-Ritz procedure. Moreover, according to our numerical tests, the weighted form $x^{(i+1)} = x^{(i)} + \cdots$ is particularly appropriate for TPCG without frequent normalizations since it avoids overflows of $\|x^{(i+1)}\|_M$. The variants of Algorithm 4.1 only differ in the scalar parameters $\iota^{(i)}$ and $\tau^{(i)}$. For instance, with an auxiliary $\gamma^{(-1)} = 1$, one can define

$$
\begin{aligned}
&\text{(a)} \quad \iota^{(i)} = 2\,\|x^{(i)}\|_M^{-2} \quad \text{and} \quad \tau^{(i)} = \gamma^{(i)}/\gamma^{(i-1)} \quad \text{for} \quad \gamma^{(i)} = (\iota^{(i)})^2\,\|r^{(i)}\|_T^2,\\
&\text{(b)} \quad \text{alternative for (a)} \quad \text{with} \quad \tau^{(i)} = \big(\gamma^{(i)} - (\iota^{(i)} r^{(i)})^* T(\iota^{(i-1)} r^{(i-1)})\big)/\gamma^{(i-1)},\\
&\text{(c)} \quad \iota^{(i)} = 1, \quad \tau^{(0)} = 1 \quad \text{and} \quad \tau^{(i)} = -\big(p^{(i)*} B T r^{(i)}\big)/\big(p^{(i)*} B p^{(i)}\big) \quad \text{for} \quad i > 0\\
&\qquad \text{with} \quad B = Q_\alpha^*(A - \beta M)Q_\alpha, \quad Q_\alpha = I - \alpha \|x^{(i)}\|_M^{-2}\, x^{(i)} x^{(i)*} M \quad \text{and} \quad \beta \le \lambda^{(i)}.
\end{aligned}
\tag{4.1}
$$

In particular, (4.1, a) can be regarded as a modification of the heuristic PCG Algorithm 2.1 where the gradient of the quadratic function $f(x) = \frac{1}{2}(x^* A_{\lambda_1} x)$ is replaced by that of the Rayleigh quotient (2.1), i.e., $\iota^{(i)} r^{(i)}$. Furthermore, (4.1, a) and (4.1, b) correspond to the Bradbury-Fletcher and Polak-Ribiére schemes mentioned in [24, Section 3], whereas (4.1, c) matches the Daniel (with $\alpha = 2$, $\beta = \lambda^{(i)}$), Perdon-Gambolati (with $\alpha = 0$, $\beta < \lambda_1$) and Jacobi (with $\alpha = 1$,

$\beta \leq \lambda^{(i)}$) schemes therein, and directly defines the conjugation between two consecutive search directions by a B-orthogonality. The matrix B is a scaled Hessian of the Rayleigh quotient (2.1) in the Daniel scheme and an approximate Hessian of the quadratic function $f(x) = \frac{1}{2}(x^* A_{\lambda_1} x)$ in the Perdon-Gambolati scheme.

The Jacobi scheme is less related to the Hessian of an objective function, but rather to the Jacobi orthogonal complement correction (JOCC) equation [32]. We briefly introduce the JOCC equation with the settings for Algorithm 4.1: by neglecting the trivial case that $x^{(i)}$ is collinear with or M-orthogonal to a target eigenvector x_1, any nonzero vector $y \in \operatorname{span}\{x_1, x^{(i)}\} \cap \operatorname{span}\{x^{(i)}\}^{\perp_M}$ (where the superscript $^{\perp_M}$ denotes M-orthogonal complements) can be scaled as $\tilde{x}^{(i)} = -x^{(i)} + \tau x_1$ so that

$$(A - \lambda_1 M)\tilde{x}^{(i)} = -(A - \lambda_1 M)x^{(i)} + \tau(A - \lambda_1 M)x_1 = -r^{(i)} - (\lambda^{(i)} - \lambda_1)Mx^{(i)}.$$

By using the projection matrix of $\operatorname{span}\{x^{(i)}\}^{\perp_M}$, i.e., $Q^{(i)} = Q_1$ from (4.1, c), and the property $\tilde{x}^{(i)} = Q^{(i)}\tilde{x}^{(i)}$, one gets the JOCC equation

$$J_{\lambda_1}^{(i)}\tilde{x}^{(i)} = -r^{(i)} \quad \text{for} \quad J_{\lambda_1}^{(i)} = Q^{(i)*}(A - \lambda_1 M)Q^{(i)}, \quad Q^{(i)} = I - \|x^{(i)}\|_M^{-2}\, x^{(i)} x^{(i)*} M. \tag{4.2}$$

The JOCC matrix $J_{\lambda_1}^{(i)}$ in (4.2) is evidently Hermitian positive semidefinite so that one can heuristically solve (4.2) by a PCG iteration to determine the eigenvector $x_1 = \tau^{-1}(x^{(i)} + \tilde{x}^{(i)})$. In practice, the unknown eigenvalue λ_1 is replaced by a guess $\beta \leq \lambda^{(i)}$ so that the search directions are conjugate with respect to $J_\beta^{(i)} = Q^{(i)*}(A - \beta M)Q^{(i)}$, i.e., a special form of B in (4.1, c). Solving this practical version of (4.2) is associated with inner steps of the Jacobi-Davidson CG (JDCG) scheme [21]. The Jacobi scheme from [24] can roughly be regarded as an alternative for JDCG with degenerate inner steps where each $J_\beta^{(i)}$ is used only once and acts as a connector of two outer steps.

Moreover, the notable analysis in [24, Section 3] shows that the TPCG schemes except the Perdon-Gambolati scheme are asymptotically equivalent to LOPCG in the final phase; see also the numerical comparison between their block versions in [23]. Thus it is possible to reduce the total computational time of LOPCG by using TPCG. We prefer the Jacobi scheme since its construction is closely associated with LOPCG so that it is reasonable to extend it as an alternative for LOPCGa using 3D instead of 4D trial subspaces. However, we note that the Jacobi scheme is implemented in [23] with another approach for updating the search direction. For Algorithm 4.1, this means that $p^{(i+1)}$ is determined by a linear combination of $Tr^{(i)}$ and a non-target Ritz vector from the previous step instead of $p^{(i)}$. This difference motivates us to formulate a compact derivation and investigate whether a direct update using $p^{(i)}$ impedes proper performance.

The following derivation of the Jacobi scheme is a heuristic reformulation of LOPCG based on [24, Section 3.2]. In Algorithm 2.2, the first two trial subspaces can be represented by

$$\mathcal{V} = \operatorname{span}\{x^{(0)}, x^{(1)}\} \quad \text{and} \quad \mathcal{W} = \operatorname{span}\{x^{(0)}, x^{(1)}, Tr^{(1)}\} = \operatorname{span}\{x^{(0)}, x^{(1)}, x^{(2)}\}.$$

By using the projection matrix $Q^{(1)}$ defined in (4.2) and the search direction $p^{(1)} = x^{(1)} - x^{(0)} \in \mathcal{V}$, we consider $q^{(1)} = Q^{(1)}p^{(1)} \in \mathcal{V} \subset \mathcal{W}$. Since $(\lambda^{(i)}, x^{(i)})$ is a Ritz pair by construction, the fact that the Ritz vector residual is orthogonal to the corresponding subspace, implies

$$0 = q^{(1)*}(A - \lambda^{(1)}M)x^{(1)} = q^{(1)*}(A - \lambda^{(2)}M)x^{(1)} - (\lambda^{(1)} - \lambda^{(2)})q^{(1)*}Mx^{(1)} = q^{(1)*}(A - \lambda^{(2)}M)x^{(1)}$$

and $0 = q^{(1)*}(A - \lambda^{(2)}M)x^{(2)}$. Thus $(A - \lambda^{(2)}M)q^{(1)}$ is orthogonal to any linear combination of $x^{(1)}$ and $x^{(2)}$, e.g., $Q^{(1)}x^{(2)}$. In other words, we get

$$0 = (Q^{(1)}x^{(2)})^*(A - \lambda^{(2)}M)q^{(1)} = x^{(2)*}y \quad \text{for} \quad y = Q^{(1)*}(A - \lambda^{(2)}M)Q^{(1)}p^{(1)}$$

so that $x^{(2)}$ belongs to $\mathcal{U} = \mathcal{W} \cap \operatorname{span}\{y\}^\perp$. Apart from trivial cases, $\mathcal{U}$ has dimension 2 and contains $x^{(1)}$ due to $Q^{(1)}x^{(1)} = 0$. A further vector in $\mathcal{U}$ can be obtained by orthogonalizing

$Tr^{(1)} \in \mathcal{W}$ against y and acts as the next search direction $p^{(2)}$, namely,

$$p^{(2)} = Tr^{(1)} - \frac{y^* Tr^{(1)}}{y^* p^{(1)}}\, p^{(1)} = Tr^{(1)} - \frac{p^{(1)*} Q^{(1)*} (A - \lambda^{(2)} M) Q^{(1)} Tr^{(1)}}{p^{(1)*} Q^{(1)*} (A - \lambda^{(2)} M) Q^{(1)} p^{(1)}}\, p^{(1)}. \tag{4.3}$$

Then applying the Rayleigh-Ritz procedure to $\mathrm{span}\{x^{(1)}, p^{(2)}\}$ yields $x^{(2)}$, and the reduced trial subspace $\mathrm{span}\{x^{(1)}, p^{(2)}\} = \mathrm{span}\{x^{(1)}, x^{(2)}\}$ can be used to reformulate the next step of LOPCG in the same way. This successive reformulation leads to Algorithm 4.1 where the counterpart of (4.3) for the $(i{+}1)$th step turns into the update formula (4.1, c) with $\alpha = 1$ and a shift $\beta \le \lambda^{(i)}$ instead of $\lambda^{(i+1)}$.

Furthermore, since the reduced trial subspaces are two-dimensional, $q^{(1)} = Q^{(1)} p^{(1)}$ and its counterparts $q^{(i)} = Q^{(i)} p^{(i)}$ in further steps are actually non-target Ritz vectors as used in the implementation in [23]. The purpose of using these Ritz vectors is likely to bypass the orthogonalization in $Q^{(i)} p^{(i)}$. However, we note that $Q^{(i)} p^{(i)}$ can be computed in an equivalent form with ready-to-use scalars so that a direct update using $p^{(i)}$ is still efficient. More precisely, for $Q^{(i)} p^{(i)} = p^{(i)} - \|x^{(i)}\|_M^{-2}\, x^{(i)} x^{(i)*} M p^{(i)}$, one can evaluate the dot product

$$x^{(i)*} M p^{(i)} = (x^{(i-1)} + \delta p^{(i)})^* M p^{(i)} = x^{(i-1)*} M p^{(i)} + \delta p^{(i)*} M p^{(i)}$$

by using components of the Gram matrix $U^* M U$ for $U = [x^{(i-1)},\, p^{(i)}]$ and the step size δ from the previous step. Thus $Q^{(i)} p^{(i)}$ is obtainable by a simple vector update.

As another important factor for practical implementation, the shift $\beta \le \lambda^{(i)}$ needs to be specified. We note that the choice $\beta = \lambda^{(i)}$ in [23] is not always close to the heuristic value $\lambda^{(i+1)}$, especially in the starting phase. In this sense, setting $\beta < \lambda^{(i)}$ is more meaningful, e.g.,

$$\beta = \max\left\{(\sigma + \lambda^{(i)})/2,\;\; 2\lambda^{(i)} - \lambda^{(i-1)}\right\} \tag{4.4}$$

with an estimated lower bound $\sigma < \lambda_1$ concerning the preconditioner (2.2). Interestingly, this shift is also appropriate for a slightly modified search direction update using $Q^{(i-1)} p^{(i)}$ (inspired by similar algorithmic simplifications like Gauss-Seidel). This combination does not deteriorate, but even slightly improves the overall performance.

Algorithm 4.2: Two specified TPCG schemes for computing (λ_1, x_1) in (1.1)

1 generate $x^{(0)}$;
2 **for** $i = 0, 1, \ldots$ until convergence **do**
3 $\quad \lambda^{(i)} = \lambda(x^{(i)}); \quad r^{(i)} = Ax^{(i)} - \lambda^{(i)} M x^{(i)}$;
4 $\quad$ **if** $i > 0$ **then**
5 $\quad\quad v = Q^{(i)} p^{(i)}$ or $Q^{(i-1)} p^{(i)}$;
6 $\quad\quad w = Av - \beta M v$ with β defined in (4.4);
7 $\quad\quad \tau^{(i)} = -(w^* Tr^{(i)})/(w^* v)$;
8 $\quad\quad p^{(i+1)} = Tr^{(i)} + \tau^{(i)} p^{(i)}$ or $Tr^{(i)} + \tau^{(i)} v$;
9 $\quad$ **else**
10 $\quad\quad p^{(i+1)} = Tr^{(i)}$;
11 $\quad x^{(i+1)} \xleftarrow{\text{RRw}} x^{(i)} + \mathrm{span}\{p^{(i+1)}\}$ or $x^{(i)} + \mathrm{span}\{Q^{(i)} p^{(i+1)}\}$ as in (2.10);

The above discussion leads to two specified TPCG schemes in Algorithm 4.2 whose differences can be seen in Lines 5, 8 and 11. For the first (and standard) version, one needs to note that the term $w^* Tr^{(i)}$ in Line 7 coincides with $w^* Q^{(i)} Tr^{(i)}$ and thus does not contradicts the description in (4.1, c). This coincidence directly follows from

$$w^* x^{(i)} = p^{(i)*} Q^{(i)*} (A - \beta M) x^{(i)} = p^{(i)*} Q^{(i)*} (A - \lambda^{(i)} M) x^{(i)} = p^{(i)*} r^{(i)} = 0$$

where the last equality uses the fact that $r^{(i)}$ is a Ritz vector residual of the previous trial subspace containing $p^{(i)}$. Concerning the performance of Algorithm 4.2, the Rayleigh-Ritz procedure applied to the reduced trial subspaces is merely an elementary calculation so that the expense per step is only slightly more than that of the heuristic PCG (Algorithm 2.1) and less than that of LOPCG ([15, `lobpcg.m`]). The concrete difference depends on how strongly the three indispensable explicit matrix-vector multiplications (with A, M and T) dominate the step (or equivalently, on the nonzero component density of these matrices). In particular, the expense per step of the second version of Algorithm 4.2 reads (1.06, 1.07, 1.12, 1.39, 1.11, 1.20) following the comparison in Table 2.3.

Algorithm 4.2 can easily be upgraded to a scheme that is more efficient than the winner LOPCGa in the numerical comparisons in Section 2.3. The trial subspaces are also augmented by an auxiliary vector a, yet the simple strategy $|\cos\angle_M(a, x^{(i)})| < 0.7$ for updating a in LOPCGa is refined by observing residual norms motivated by the following test. Therein the second version of Algorithm 4.2 is denoted by TPCG and compared with LOPCG and LOPCGa concerning the fifth case from Table 2.2 (containing clustered eigenvalues). As seen in Figure 4.1, their convergence history is illustrated in terms of

$$\lambda:\ \lambda^{(i)}-\lambda_1,\quad \nu:\ \|r^{(i)}\|_2/\|x^{(i)}\|_M,\quad \phi:\ \phi^{(i)}=|\cos\angle_M(a,x^{(i)})|$$
$$\Delta\lambda:\ |\lambda^{(i)}/\lambda^{(i-1)}-1|^{1/2},\quad \Delta\phi:\ |\phi^{(i)}/\phi^{(i-1)}-1|. \tag{4.5}$$

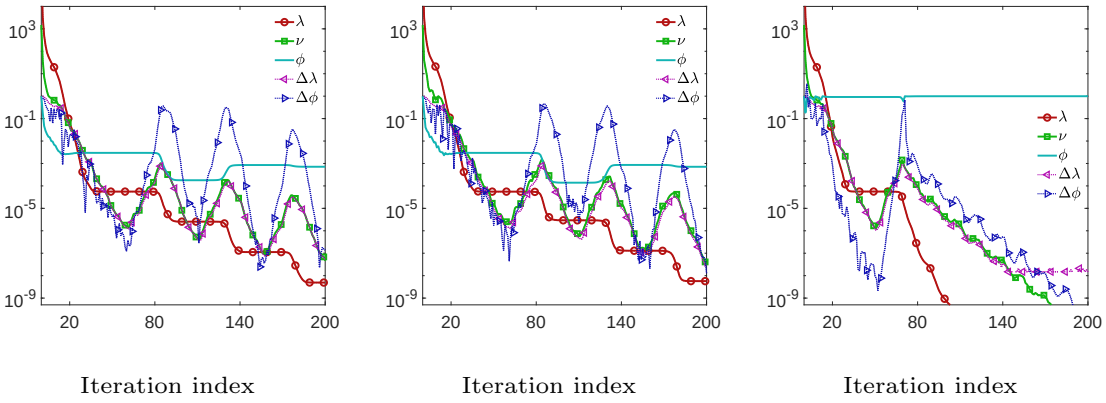

FIGURE 4.1. Comparing LOPCG and TPCG with LOPCGa in terms of (4.5) to suggest timely augmentation. Changes of the residual norm ν can provide good timing.

This comparison clearly reflects the asymptotic equivalence between LOPCG (left subplot) and TPCG (middle subplot). Moreover, since the previous test in Figure 2.3 shows that the starting phase of LOPCG cannot be significantly accelerated, we only need to observe the data as of the first delay. Guided by the strategy $\phi^{(i)} = |\cos\angle_M(a, x^{(i)})| < 0.7$ for LOPCGa (right subplot), we note that substantial changes of $\phi^{(i)}$ for $a = x^{(0)}$ are almost simultaneous with peaks of ν. This relation is more obvious by observing $\Delta\phi$. Therefore a more reasonable augmentation strategy is to detect the residual peaks. Another remarkable phenomenon is that $\Delta\lambda$ almost matches with ν, and is probably useful for a novel approach in future investigation of LOPCG and TPCG.

We further note that the simple strategy $\phi^{(i)} < \tau$ with $\tau = 0.7$ is not suitable for the augmentation of TPCG and even causes a deceleration. The left and middle subplots in Figure 4.2 illustrate the acceptable performance by using two other thresholds: $\tau = 0.99$ and $\tau = 0.001$. The latter is more effective than the former in the sense of fewer updates in the starting phase and no further delays in the final phase. Moreover, a is initialized by $x^{(0)}$ and updated by the

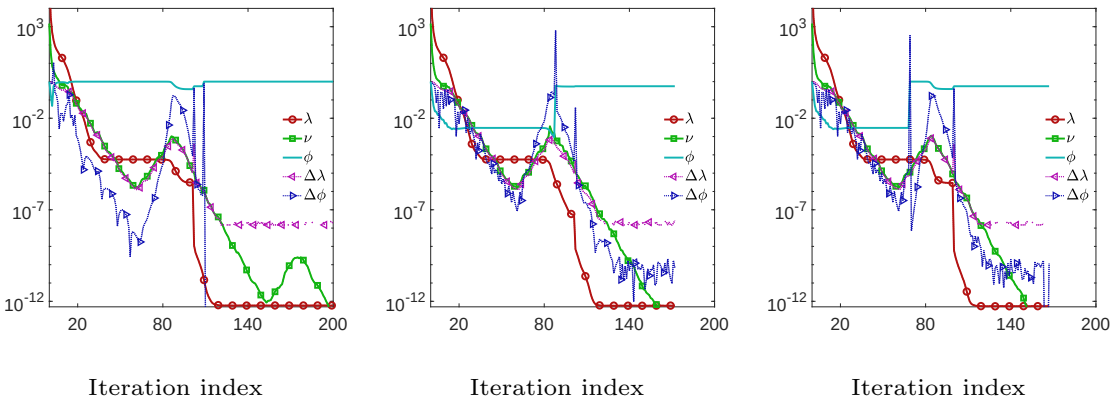

FIGURE 4.2. Comparing augmented versions of TPCG using the angle-based criterion $\phi^{(i)} = |\cos \angle_M(a, x^{(i)})| < \tau$ or a residual-based criterion. Left: with $\tau = 0.99$. Middle: with $\tau = 0.001$. Right: update when the residual norm ν decreases again after a substantial increase.

iterate corresponding to the minimum of the (already computed) residual norm ν. This mostly improves the trivial update setting $a = x^{(i)}$ as concluded from our additional tests. Next, the right subplot presents a convenient residual-based strategy which allows reducing dependence on the threshold choice. In our attempts behind that, three time points are tested to update the augmentation: (i) directly after a substantial increase (valley) of the residual norm ν; (ii) when ν decreases again; (iii) when ν becomes smaller than its valley value. The approaches (i) and (ii) well avoid the delays of the non-augmented TPCG or LOPCG. However, (i) slightly slows down the uphill steps after the valley of ν so that a better timing is given by (ii). The approach (iii) is less appropriate due to a further (but only one) delay after the update. Thus the approach (ii) is demonstrated in Figure 4.2 (right) where the hurdle-free downhill path of ν is comparable with that by using the threshold $\tau = 0.001$ in Figure 4.2 (middle). In addition, ν falls below `1e-12` before the 160th step which means a significant improvement in comparison to that before the 225th step of LOPCGa observed in Figure 2.3 (middle).

Summarizing the above, we suggest to modify Algorithm 4.2 by defining $\nu^{(i)} = \|r^{(i)}\|_2 / \|x^{(i)}\|_M$ and adding to

$$\begin{aligned} &\text{Line 3}: \quad \text{store the iterate } \check{x} \text{ corresponding to } \nu_{\min} = \min_{j \le i} \nu^{(j)}, \\ &\text{Line 11}: \quad \text{augment the trial subspace by } \check{x} \text{ if } \nu^{(i)} \text{ reaches a peak} \end{aligned}$$

where the peak can be detected by getting `flag=2` from

$$\begin{aligned} &\text{if } \texttt{flag==0} \text{ and } \nu^{(i)} > 1.5\,\nu_{\min}, \quad \text{set } \texttt{flag=1}, \\ &\text{if } \texttt{flag==1} \text{ and } \nu^{(i)} < \nu^{(i-1)}, \quad \text{set } \texttt{flag=2}. \end{aligned} \tag{4.6}$$

The expense per step of the resulting "TPCGa" scheme is close to that of LOPCGa by taking the comparison in Table 2.3. Therefore the reduction of the total computational time is directly reflected by the number of required steps. Moreover, we note that (4.6) is particularly appropriate for computing clustered eigenvalues. In other cases such as the examples in Figure 2.2, the residual norm could oscillate very frequently so that a perfect timing cannot easily be predicted. Then it is more effective to check a multi(e.g. 3)-step decrease of ν instead of the single-step decrease $\nu^{(i)} < \nu^{(i-1)}$. Last but not least, the residual-based criterion allows us to drop the augmentation in the starting phase so that the total expense can be further reduced. Following Figures 2.2 and 2.3, we compare this final version of TPCGa with LOPCGa in Figure 4.3. Therein λ_a and ν_a denote the terms λ and ν from (4.5) applied to LOPCGa.

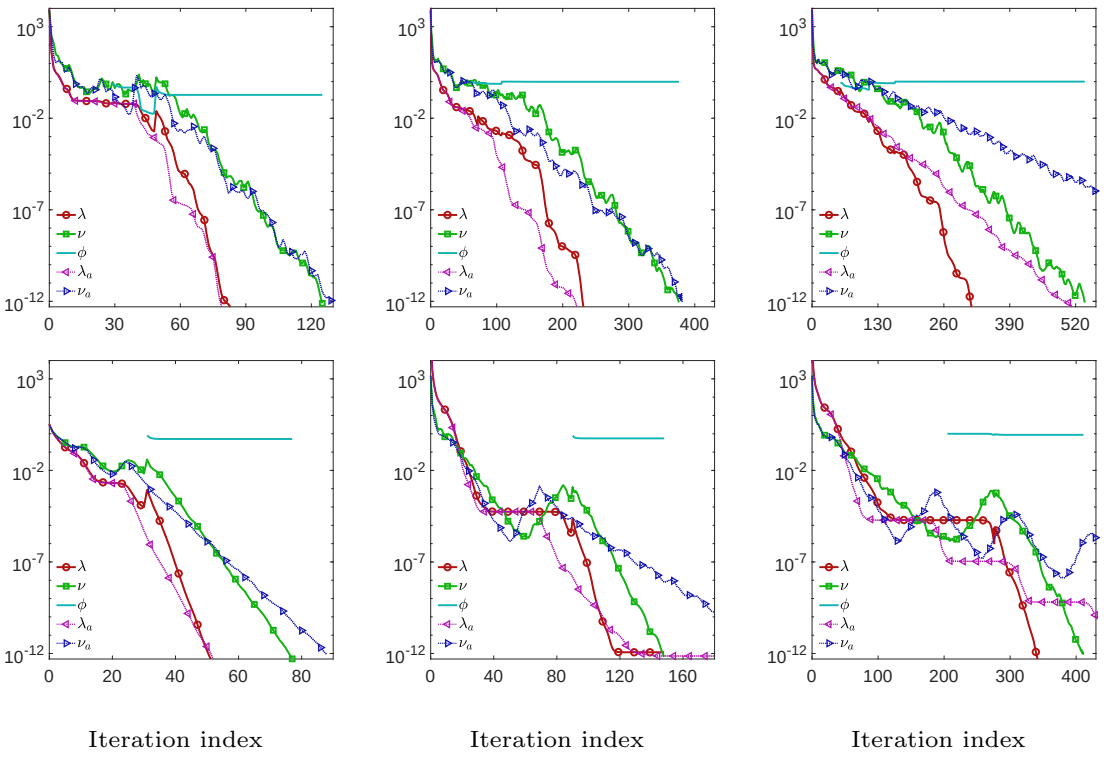

FIGURE 4.3. Comparing TPCGa with LOPCGa in terms of (4.5) where λ_a and ν_a stand for λ (Ritz value error) and ν (residual norm) applied to LOPCGa. The upper/lower row can be combined with the comparison in Figure 2.2/2.3. The benefit of TPCGa is significant for less accurate preconditioners and clustered eigenvalues.

Remark 4.1. Since the above numerical tests are performed in Matlab with double precision (following [12, 29, 30]), a further issue is whether LOPCGa and TPCGa can improve LOPCG in some other environments. Therefore we have rewritten our codes in Python and compared them with [15, `lobpcg.py`]. Figure 4.4 is concerned with a numerical example using $A = \mathrm{diag}(1.998, 1.999, 3, \ldots, n)$ and $M = T = I$. In single precision, random tests for $n \geq 1000$ indicate that the residual norm reduction by `lobpcg.py` frequently oscillates and can only reach `2e-5`. Also LOPCGa appears less stable, but can be stabilized simply by normalizing the augmentation vector. With this adjustment, the residual norm reduction by LOPCGa can fall below `6e-6` as by TPCGa. In double precision, the improvements yielded by LOPCGa and TPCGa can be stated as also observed in the Matlab tests. In Figure 4.4, we particularly illustrate three Python tests with $n \in \{1000, 5000, 10000\}$ and $x^{(0)} = (1, \ldots, 1)^*$, indicating the usefulness of TPCGa.

Conclusion

Competitive eigensolvers can be constructed by mimicking or modifying the traditional CG iteration for solving linear systems. However, the optimality of the original search directions is not preserved, but occasionally heavily disturbed as observed in the staircase-shaped convergence history of LOPCG and its recent alternatives while computing clustered eigenvalues. Our single-vector scheme TPCGa with augmented two-term recurrences overcomes this drawback by detecting peaks of residual norms and punctually renewing the augmentation. The low dimension (two or three) of the trial subspace in TPCGa enables an economical implementation

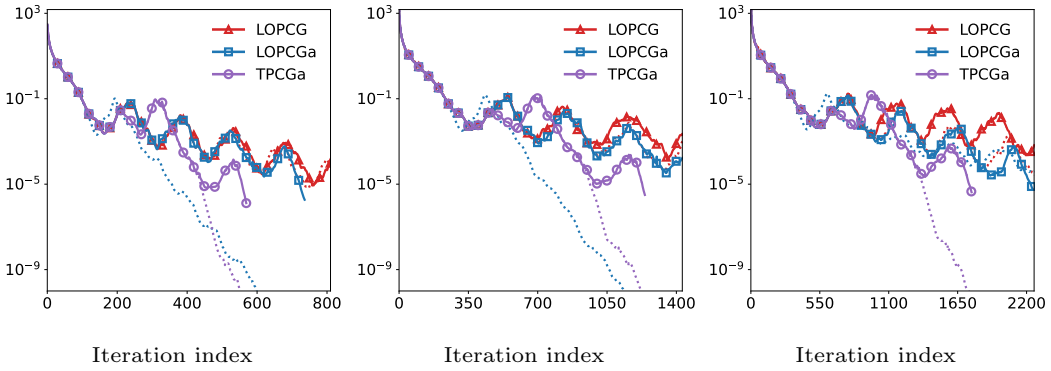

FIGURE 4.4. Residual norm reduction of LOPCG, LOPCGa and TPCGa in Python for the numerical example in Remark 4.1. Left: $n = 1000$. Middle: $n = 5000$. Right: $n = 10000$. Solid curves: single precision. Dotted curves: double precision.

with implicit updates of matrix-vector products as in LOPCG, leading to notably better performance with respect to the number of required steps and also the total computational time (not restricted to the case of clustered eigenvalues). This approach also induces future topics concerning blockwise implementations and convergence analyses.

Appendix A. Typical generalized eigenvalue problems

The introduction of LOPCG in [12] is concerned with computing the largest eigenvalue of

$$Mx = \mu Ax, \quad M, A \in \mathbb{C}^{n\times n} \text{ Hermitian}, \quad A \text{ positive definite}. \tag{A.1}$$

This covers two cases of the eigenvalue problem

$$Lu = \lambda Su, \quad L, S \in \mathbb{C}^{n\times n} \text{ Hermitian}, \quad S \text{ positive definite} \tag{A.2}$$

arising from a discretization of a self-adjoint elliptic partial differential operator.

The first case coincides with the case introduced in Section 1 (by setting $L = A$, $S = M$), i.e., the smallest eigenvalue λ_1 of the matrix pair (L, S) is of interest. For matching the form (A.1), one uses a shift $\sigma < \lambda_1$ so that (A.2) can be reformulated as $\tilde{L}u = (\lambda - \sigma)Su$ with the shifted matrix $\tilde{L} = L - \sigma S$ which is evidently positive definite. Then (A.1) corresponds to (A.2) by setting $M = S$, $A = \tilde{L}$ and $\mu = (\lambda - \sigma)^{-1}$. Moreover, in a practical treatment with preconditioned eigensolvers, linear systems of the form $\tilde{L}v = w$ are to be solved approximately.

The second case deals with a shift σ close to but not equal to an interior eigenvalue, i.e., $\tilde{L}$ is indefinite and invertible. Then (A.2) can be reformulated as $(\tilde{L}S^{-1}\tilde{L})u = (\lambda - \sigma)\tilde{L}u$ and matches (A.1) with $M = \pm\tilde{L}$, $A = \tilde{L}S^{-1}\tilde{L}$ and $\mu = \pm(\lambda - \sigma)^{-1}$. Therein the symbol $\pm$ means determining eigenvalues either larger or smaller than σ. An alternative representation uses $M = S$, $A = \tilde{L}S^{-1}\tilde{L}$ and $\mu = (\lambda - \sigma)^{-2}$. Applying preconditioned eigensolvers to this case requires approximately solving linear systems of the form $(\tilde{L}S^{-1}\tilde{L})v = w$ which corresponds to two linear systems for the matrix $\tilde{L}$ in the implementation.

A further special form of (A.1) concerns Hermitian definite matrix pencils [16]

$$Lu = \lambda Su, \quad L, S \in \mathbb{C}^{n\times n} \text{ Hermitian}, \quad \tilde{L} = L - \sigma S \text{ definite for a certain } \sigma \in \mathbb{R}. \tag{A.3}$$

If $\tilde{L}$ is positive definite, the transformation into (A.1) is the same as in the first case for (A.2). For negative definite $\tilde{L}$, one sets $M = S$, $A = -\tilde{L}$ and $\mu = -(\lambda - \sigma)^{-1}$ instead. Moreover, changing the signs of M and μ (in both cases) allows determining eigenvalues on both sides of

σ. A practical eigenvalue arrangement with respect to the inertia of the possibly indefinite and singular matrix S is suggested in [16, Theorem 2.1].

One can also use (A.1) for describing the linear response eigenvalue problem [3]. Therein the smallest positive eigenvalues of the block anti-diagonal matrix

$$\begin{bmatrix} O & \tilde{L} \\ \tilde{S} & O \end{bmatrix}, \quad \tilde{L}, \tilde{S} \in \mathbb{C}^{\tilde{n}\times\tilde{n}} \text{ Hermitian positive semidefinite, one of them definite} \tag{A.4}$$

are to be computed. A simple reformulation uses the fact that the target eigenvalues are square roots of the smallest nonzero eigenvalues of the matrix pair $(\tilde{S}, \tilde{L}^{-1})$ or $(\tilde{L}, \tilde{S}^{-1})$ provided that $\tilde{L}$ or $\tilde{S}$ is definite. This is evidently included in the first case for (A.2). If both of $\tilde{L}$ and $\tilde{S}$ are definite, a reformulation suggested in [16] reads

$$Lu = \lambda S u \quad \text{with} \quad L = \begin{bmatrix} \tilde{L} & O \\ O & \tilde{S} \end{bmatrix} \quad \text{and} \quad S = \begin{bmatrix} O & \tilde{I} \\ \tilde{I} & O \end{bmatrix} \tag{A.5}$$

where $\tilde{I}$ denotes the $\tilde{n}\times\tilde{n}$ identity matrix. This matches (A.3) with $\sigma = 0$. Transformations of (A.2) and (A.3) into (A.1) enable constructing suitable eigensolvers based on those for (A.1).

References

[1] H.M. Aktulga, M. Afibuzzaman, S. Williams, A. Buluç, M. Shao, C. Yang, E.G. Ng, P. Maris, and J.P. Vary, *A high performance block eigensolver for nuclear configuration interaction calculations*, IEEE TPDS 28 (2017), 1550–1563. https://doi.org/10.1109/TPDS.2016.2630699

[2] P. Arbenz, U.L. Hetmaniuk, R.B. Lehoucq, and R.S. Tuminaro, *A comparison of eigensolvers for large-scale 3D modal analysis using AMG-preconditioned iterative methods*, Int. J. Numer. Meth. Engng 64 (2005), 204–236. https://doi.org/10.1002/nme.1365

[3] Z. Bai and R.C. Li, *Minimization principles for the linear response eigenvalue problem I: Theory*, SIAM J. Matrix Anal. Appl. 33 (2012), 1075–1100. https://doi.org/10.1137/110838960

[4] Z. Bai and R.C. Li, *Minimization principles for the linear response eigenvalue problem II: Computation*, SIAM J. Matrix Anal. Appl. 34 (2013), 392–416. https://doi.org/10.1137/110838972

[5] Y. Cai, Z. Bai, J.E. Pask, and N. Sukumar, *Convergence analysis of a locally accelerated preconditioned steepest descent method for Hermitian-definite generalized eigenvalue problems*, J. Comp. Math. 36 (2018), 739–760. https://doi.org/10.4208/jcm.1703-m2016-0580

[6] J.K. Cullum and R.A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Vol. I: Theory*, SIAM, Philadelphia, 2002. https://doi.org/10.1137/1.9780898719192

[7] T.A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Softw. 38 (2011), 1–25. https://doi.org/10.1145/2049662.2049663

[8] J.A. Duersch, M. Shao, C. Yang, and M. Gu, *A robust and efficient implementation of LOBPCG*, SIAM J. Sci. Comput. 40 (2018), C655–C676. https://doi.org/10.1137/17M1129830

[9] U. Hetmaniuk and R. Lehoucq, *Basis selection in LOBPCG*, J. Comput. Phys. 218 (2006), 324–332. https://doi.org/10.1016/j.jcp.2006.02.007

[10] A.V. Knyazev, *Convergence rate estimates for iterative methods for a mesh symmetric eigenvalue problem*, Russian J. Numer. Anal. Math. Modelling 2 (1987), 371–396. https://doi.org/10.1515/rnam.1987.2.5.371

[11] A.V. Knyazev, *Preconditioned eigensolvers—an oxymoron?* Electron. Trans. Numer. Anal. 7 (1998), 104–123. https://etna.ricam.oeaw.ac.at/volumes/1993-2000/vol7

[12] A.V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput. 23 (2001), 517–541. https://doi.org/10.1137/S1064827500366124

[13] A.V. Knyazev, M. E. Argentati, I. Lashuk, and E.E. Ovtchinnikov *Block locally optimal preconditioned eigenvalue xolvers (BLOPEX) in hypre and PETSc*, SIAM J. Sci. Comput. 29 (2007), 2224–2239. https://doi.org/10.1137/060661624

[14] A.V. Knyazev and K. Neymeyr, *A geometric theory for preconditioned inverse iteration III: A short and sharp convergence estimate for generalized eigenvalue problems*, Linear Algebra Appl. 358 (2003), 95–114. https://doi.org/10.1016/S0024-3795(01)00461-X

[15] A.V. Knyazev et al., *Codes of LOBPCG in Matlab and Python.* https://github.com/lobpcg/blopex/blob/master/blopex_tools/matlab/lobpcg/lobpcg.m (version 26-April-2021) ; https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.lobpcg.html (version 29-August-2025)

[16] D. Kressner, M.M. Pandur, and M. Shao, *An indefinite variant of LOBPCG for definite matrix pencils*, Numer. Algor. 66 (2014), 681–703. https://doi.org/10.1007/s11075-013-9754-3

[17] L. Lin, S. Shao, and W. E, *Efficient iterative method for solving the Dirac-Kohn-Sham density functional theory*, J. Comput. Phys. 245 (2013), 205–217. https://doi.org/10.1016/j.jcp.2013.03.030

[18] A. Levitt and M. Torrent, *Parallel eigensolvers in plane-wave density functional theory*, Comput. Phys. Commun. 187 (2015), 98–105. https://doi.org/10.1016/j.cpc.2014.10.015

[19] K. Neymeyr, *On preconditioned eigensolvers and Invert-Lanczos processes*, Linear Algebra Appl. 430 (2009), 1039–1056. https://doi.org/10.1016/j.laa.2008.10.016

[20] K. Neymeyr, *A geometric convergence theory for the preconditioned steepest descent iteration*, SIAM J. Numer. Anal. 50 (2012), 3188–3207. https://doi.org/10.1137/11084488X

[21] Y. Notay, *Combination of Jacobi-Davidson and conjugate gradients for the partial symmetric eigenproblem*, Numer. Linear Algebra Appl. 9 (2002), 21–44. https://doi.org/10.1002/nla.246

[22] E.E. Ovtchinnikov, *Sharp convergence estimates for the preconditioned steepest descent method for Hermitian eigenvalue problems*, SIAM J. Numer. Anal. 43 (2006), 2668–2689. https://doi.org/10.1137/040620643

[23] E.E. Ovtchinnikov, *Computing several eigenpairs of Hermitian problems by conjugate gradient iterations*, J. Comput. Phys. 227 (2008), 9477–9497. https://doi.org/10.1016/j.jcp.2008.06.038

[24] E.E. Ovtchinnikov, *Jacobi correction equation, line search, and conjugate gradients in Hermitian eigenvalue computation I: Computing an extreme eigenvalue*, SIAM J. Numer. Anal. 46 (2008), 2567–2592. https://doi.org/10.1137/070688742

[25] Y. Saad, *On the rates of convergence of the Lanczos and the block-Lanczos methods*, SIAM J. Numer. Anal. 17 (1980), 687–706. https://doi.org/10.1137/0717059

[26] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2003. https://doi.org/10.1137/1.9780898718003

[27] B.A. Samokish, *The steepest descent method for an eigenvalue problem with semi-bounded operators*, Izv. Vyssh. Uchebn. Zaved. Mat. 5 (1958), 105–114 (in Russian). https://www.mathnet.ru/eng/ivm2983

[28] M. Shao, H.M. Aktulga, C. Yang, E.G. Ng, P. Maris, and J.P. Vary, *Accelerating nuclear configuration interaction calculations through a preconditioned block iterative eigensolver*, Comput. Phys. Commun. 222 (2018), 1–13. https://doi.org/10.1016/j.cpc.2017.09.004

[29] N. Shao, W. Chen, and Z. Bai, *EPIC: A provable accelerated eigensolver based on preconditioning and implicit convexity*, SIAM J. Matrix Anal. Appl. 46 (2025), 45–73. https://doi.org/10.1137/24M1641440

[30] N. Shao and W. Chen, *Riemannian acceleration with preconditioning for symmetric eigenvalue problems*, Numer. Math. 157 (2025), 307–354. https://doi.org/10.1007/s00211-025-01451-0

[31] A. Stathopoulos, *Nearly optimal preconditioned methods for Hermitian eigenproblems under limited memory. Part I: Seeking one eigenvalue*, SIAM J. Sci. Comput. 29 (2007), 481–514. https://doi.org/10.1137/050631574

[32] G.L.G. Sleijpen and H.A. Van der Vorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl. 17 (1996), 401–425. https://doi.org/10.1137/S0895479894270427

[33] L. Wu, F. Xue, and A. Stathopoulos, *TRPL+K: Thick-restart preconditioned Lanczos+K method for large symmetric eigenvalue problems*, SIAM J. Sci. Comput. 41 (2019), A1013–A1040. https://doi.org/10.1137/17M1157568

[34] M. Zhou, Z. Bai, Y. Cai, and K. Neymeyr, *Convergence analysis of a block preconditioned steepest descent eigensolver with implicit deflation*, Numer. Linear Algebra Appl. 30 (2023), e2498. https://doi.org/10.1002/nla.2498

Universität Rostock, Institut für Mathematik, Ulmenstrasse 69, 18055 Rostock, Germany

E-mail address: ming.zhou at uni-rostock (dot) de

Universität Rostock, Institut für Mathematik, Ulmenstrasse 69, 18055 Rostock, Germany

E-mail address: klaus.neymeyr at uni-rostock (dot) de