Randomized LOBPCG algorithm with linear dimension reduction

Yanfei Xiang

RANDOMIZED LOBPCG ALGORITHM WITH LINEAR DIMENSION REDUCTION

YANFEI XIANG*

Research report

Abstract. We present a randomized variant of the locally optimal block preconditioned conjugate gradient (LOBPCG) method that incorporates linear dimension reduction for computing a group of eigenpairs of generalized eigenvalue problems. In contrast to the well-established LOBPCG method, we do not assert that our randomized variant always yields a superior converging eigensolver. Instead, our objective is to offer a preliminary exploration of merging statistical randomization into deterministic LOBPCG, aiming to enhance our understanding of the numerical performance of this randomized LOBPCG variant for solving eigenvalue problems. Additionally, this work paves some potential avenues for integrating dimension reduction techniques into other deterministic numerical linear algebra algorithms. Finally, we empirically demonstrate the performance of this novel randomized LOBPCG variant using extensive academic examples, revealing that its efficiency is comparable to, and in some cases surpasses, original LOBPCG.

Key words. LOBPCG, Block subspace eigensolver, Block Rayleigh-Ritz procedure, Extreme-scale eigenvalue problem, Randomized linear dimension reduction, Random sketching, Randomized numerical linear algebra

1. Introduction.

In this work, we consider the problem of computing a group of, says p, smallest eigenvalues together with its invariant subspace of eigenvectors of the generalized eigenvalue problem:

$$AX = BX\Lambda, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$ is a large sparse symmetric matrix, $B \in \mathbb{R}^{n\times n}$ is a sparse symmetric positive definite (SPD) matrix (like an identity matrix $B = I$ with size of matrix A), $\Lambda = \mathrm{Diag}(\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(p)}) \in \mathbb{R}^{p\times p}$ is a diagonal matrix with the approximated eigenvalues as the diagonal elements, and $X = [x^{(1)}, x^{(2)}, \ldots, x^{(p)}] \in \mathbb{R}^{n\times p}$ (of full rank with $1 < p \lll n$) are the corresponding eigenvectors to be computed. The (Λ, X) is a group of eigenpairs, and the eigenvalues of (A, B) may have arbitrary multiplicity and satisfy $0 \leq \lambda^{(1)} \leq \lambda^{(2)} \leq \ldots \leq \lambda^{(p)}$.

Numerous techniques within numerical linear algebra (NLA) are available to solve the eigenvalue problem (1.1). These encompass a variety of methods, including the locally optimal block preconditioned conjugate gradient (LOBPCG) method [17, 18], the preconditioned inverse iteration methods [24–26], the preconditioned gradient-based methods [27, 29], and other Krylov-type methods illustrated in [32].

The NLA algorithms are traditionally deterministic techniques. Over the recent decades, algorithm designers in the NLA community have been employed probabilistic random embeddings for many scientific computing tasks, which has now emerged as randomized NLA (simplified as RandNLA). Overall, the integration of probabilistic randomization and deterministic NLA can be categorized into three approaches: (1) *sketch-and-solve*, (2) *iterative-sketching*, and (3) *sketch-and-precondition*. As their name suggests, *sketch-and-solve* paradigm focus on exploiting sketching to solve a target problem directly; while *sketch-and-precondition* means using sketching to construct a properly preconditioner to accelerate the convergence. The *iterative-sketching* refers to applying sketching to a given vector or matrix recursively to improve the accuracy of the sketched variant. These three categories have been described in many RandNLA literatures, we refer the reader to a recent comprehensive survey by Martinsson and Tropp [21, Section 10] for more details on their differences. We also refer the reader to [10, 36] and their extensive bibliographies for a comprehensive treatment of the ideas of RandNLA.

In this work, we only focus on applying *sketch-and-solve* and *iterative-sketching* with subspace embeddings to the deterministic LOBPCG algorithm to solve the eigenvalue problem (1.1). To be specific, we apply *sketch-and-solve* with subspace embeddings to approximate the least squares problem [21, Section 10] and the matrix-matrix multiplications [7] occurred in parts of LOBPCG implementations. Then following the philosophy of *iterative-sketching*, the processes of applying subspace embeddings to LOBPCG will be executed iteratively along the iteration steps. Moreover, we summarize the pros and cons of such randomized LOBPCG variant (simplified as sLOBPCG, where "s" refers to sketching) to provide some hints for the question of when to use randomization for NLA. This work also includes computational evidence that the sLOBPCG algorithm is as efficient as the deterministic LOBPCG algorithm, or even outperforms LOBPCG for some particular classes of problems.

*yfxiang0amber@gmail.com. Concace team, Inria Centre at the University of Bordeaux, France. Part of this work was completed during the author's postdoctoral research with the Alpines team at Centre Inria de Paris, Laboratoire Jacques-Louis Lions (LJLL), France, under the supervision of Laura Grigori (EPFL and PSI, Switzerland).

The concept of incorporating randomization into deterministic algorithms, as considered in this paper, is not without precedent. For the example of applying subspace embeddings for the least-square problem, Balabanov and Grigori [4] have applied the subspace embeddings iteratively to the Gram-Schmidt process (with long-term recurrence) to obtain a randomized Gram-Schmidt process (simplified as RGS). Then this RGS has been applied to the generalized minimum residual norm (GMRES) method [33] to solve linear system with single right-hand side. In [5], they further developed a block version of the Arnoldi process with RGS for solving linear systems with multiple right-hand sides and addressing clustered eigenvalue problems. Parallel, similar literature has been published by Nakatsukasa and Tropp in [23] for solving eigenvalue problem. Furthermore, randomized embeddings have also been applied to the generalized conjugate residual with inner orthogonalization (GCRO) and its deflated restarting variant (GCRO-DR) [30], and the GMRES-DR method [22] in a recent randomized work [13] for addressing linear systems. For the examples of applying subspace embeddings to approximate the matrix-matrix multiplication, we refer the reader to [34, Algorithm 4], which presents the randomized version of the conjugate gradient algorithm with short-term recurrence.

The manuscript is structured as follows: Section 2 serves as the main body of this work. It includes the necessary background to introduce the notation required for describing the new algorithm and outlines its properties. Specifically, Section 2.1 revisits the fundamental concepts of the deterministic LOBPCG algorithm that is the basis of our approach. Section 2.2 reviews research on sketching techniques in NLA, including sketching realized by structured and unstructured embeddings. This section also explains how sketching can be integrated into deterministic algorithms to create corresponding RandNLA variants. Section 2.3 presents the new sLOBPCG algorithm with randomized sketching and analyzes its asymptotic complexity compared to LOBPCG. Section 3 provides numerical results and finally some concluding remarks are presented in Section 4.

Before delving into the main content, this paragraph introduces the notations used throughout the manuscript. The vectors are denoted by lowercase letters; matrices with multiple columns are described by uppercase letters. The Euclidean inner product is denoted as $(\cdot,\cdot)$. The notation $\mathbb{R}$ refers to the real number field. For convenience of the algorithm illustration and presentation, some MATLAB notation is used. The symbol $||\cdot||$ denotes the Euclidean norm default for both vectors and matrices, and the Frobenius norm is denoted with the subscript $_F$. The superscript ^{T} denotes the transpose. The superscript † refers to the Moore-Penrose inverse. Without special note, a subscript $_j$ for a vector or matrix is used to indicate that the vector or matrix is obtained at iteration j. A matrix $C \in \mathbb{R}^{m\times\ell}$ consisting of m rows and ℓ columns sometimes is denoted as $C_{m\times\ell}$ explicitly; furthermore, we denote by Span$\{C\}$ the space spanned by the columns of a nonsingular matrix C. The identity and null matrices of dimension m are denoted, respectively, by I_m and 0_m or by just I and 0 when the dimension is evident from the context.

2. The sketched LOBPCG algorithm with randomized embedding.

For the sake of completeness, this section contains the essential background which enables us to introduce the notation required to describe the new algorithm and detail its properties. In that respect, we first recall the main ingredients of the locally optimal block preconditioned conjugate gradient (LOBPCG) method [15,17,18] proposed by A. V. Knyazev, which is the deterministic algorithm that we rely on for this work. Next, we describe the rough information of randomized sketching in the randomized numerical linear algebra (RandNLA) [21], and discuss two of their common applications categorized in the *sketch-and-solve* paradigm. The driving ideas of these two applications with randomized sketching are finally merged in the LOBPCG context, leading to the new randomized sLOBPCG algorithm.

2.1. The LOBPCG algorithm.

The LOBPCG method is an attractive gradient-based eigensolver for the solution of (1.1). With a given random initial guess, the governing idea of LOBPCG is to iteratively approximate the eigenpairs of the original large matrix pencil $A - \lambda^{(i)}B$ $(i = 1,\ldots,p)$ by computing eigenpairs of a relatively smaller one obtained by the projection method. To be specific, the Rayleigh-Ritz (RR) projection described in Definition 1 is applied to a properly defined trail subspace with full-column rank. Then a relatively smaller system described in Proposition 1 is to be solved iteratively to approximate eigenpairs of original larger system.

DEFINITION 1 (Rayleigh-Ritz projection). *Consider a trial subspace $\mathcal{W}$ of $\mathbb{R}^n$ with full-column rank. Given two general nonsingular matrices $A, B \in \mathbb{R}^{n\times n}$, a scale $\lambda \in \mathbb{R}$, and a vector $y \in \mathcal{W}$, we see that (λ, y) is a Rayleigh-Ritz pair of the pencil $A - \lambda B$ with respect to the trial subspace $\mathcal{W}$ if and only if*

$$Ay - \lambda By \perp \mathcal{W}$$

or equivalently,

$$\forall w \in \mathrm{Range}(\mathcal{W}) \quad w^T\,(Ay - \lambda By) = 0.$$

The vector y is a Rayleigh-Ritz vector associated with the Rayleigh-Ritz value λ.

Denote the approximated eigenpairs of the generalized eigenvalue problem (1.1) obtained at the j-th ($j = 0, 1, \ldots$) iteration as (Λ_j, X_j), where $\Lambda_j \in \mathbb{R}^{p\times p}$ ($p > 1$) refers to the multiple eigenvalues to be approximated and $X_j \in \mathbb{R}^{n\times p}$ refers to the corresponding eigenvectros. The residuals of this eigenpairs could be formed as $R_j = AX_j - BX_j\Lambda_j$. With an extra recycling block search direction P_j initialized as zero (i.e., $P_0 = 0$), the trail subspace in the j-th iteration of LOBPCG is defined as

$$\mathcal{W}_j = \mathrm{Span}\{X_j, R_j, P_j\} \in \mathbb{R}^{n\times p_Y},\ (\text{where } p_Y = 3\times p \lll n). \tag{2.1}$$

With the RR process described in Definition 1 and the trail subspace $\mathcal{W}_j$ formed in Equation (2.1), the new approximated eigenvalues Λ_{j+1} of LOBPCG satisfy the relation $\Lambda_{j+1} \in \mathrm{Span}\{\mathcal{W}_j^T A\mathcal{W}_j(\mathcal{W}_j^T B\mathcal{W}_j)^{-1}\}$, referring to the Rayleigh quotient of the actual nonzero approximated egenvectors $X_j \in \mathcal{W}_j$.

PROPOSITION 1 (Based on Definition 1 and [17, Section 5]). *At each iteration of LOBPCG, the Rayleigh-Ritz method is applied for the pencil $A - \lambda^{(i)}B$ ($i = 1, \ldots, p$) with respect to the trail subspace $\mathcal{W}_j = [X_j, R_j, P_j] \in \mathbb{R}^{n\times p_Y}$ ($j = 0, 1, \ldots,\ P_0 = 0,\ p_Y = 3\times p$). This follows a generalized eigenvalue problem:*

$$G_A y_i = \lambda^{(i)} G_B y_i, \quad \textit{for } 1 \le i \le p_Y, \tag{2.2}$$

where

$$G_A = \mathcal{W}_j^T A\mathcal{W}_j = \begin{bmatrix} (X_j, AX_j) & (X_j, AR_j) & (X_j, AP_j) \\ (R_j, AX_j) & (R_j, AR_j) & (R_j, AP_j) \\ (P_j, AX_j) & (P_j, AR_j) & (P_j, AP_j) \end{bmatrix} \in \mathbb{R}^{p_Y\times p_Y} \tag{2.3}$$

and

$$G_B = \mathcal{W}_j^T B\mathcal{W}_j = \begin{bmatrix} (X_j, BX_j) & (X_j, BR_j) & (X_j, BP_j) \\ (R_j, BX_j) & (R_j, BR_j) & (R_j, BP_j) \\ (P_j, BX_j) & (P_j, BR_j) & (P_j, BP_j) \end{bmatrix} \in \mathbb{R}^{p_Y\times p_Y} \tag{2.4}$$

for the computation of the Rayleigh-Ritz pair $(\lambda^{(i)}, \mathcal{W}_j y_i)$ used for updating the trail subspace for the next iteration. Note that G_B reduces to the identity matrix (i.e., $G_B = I_{p_Y}$) when all the columns of $\mathcal{W}_j$ are B-orthonormal to each other.

Proof. The proofs basically rely on some matrix computations as briefly outlined below.

- *Case 1*: When all the columns of $\mathcal{W}_j$ are B-orthonormal to each other, according to Definition 1, each Rayleigh-Ritz pair $(\lambda^{(i)}, \mathcal{W}_j y_i)$ satisfies

  $$\forall w \in \mathrm{Range}(\mathcal{W}_j) \quad w^T\,(A\mathcal{W}_j y_i - \lambda^{(i)}\, B\mathcal{W}_j y_i) = 0,$$

  which is equivalent to

  $$\mathcal{W}_j^T\,(A\mathcal{W}_j y_i - \lambda^{(i)} B\mathcal{W}_j y_i) = 0 \text{ with } \mathcal{W}_j^T B\mathcal{W}_j = I_{p_Y}. \tag{2.5}$$

- *Case 2*: When the columns of $\mathcal{W}_j$ just be linearly independent rather than B-orthonormal, we denote its basis as $\tilde{\mathcal{W}}_j$ that with B-orthonormal columns. Thus there is a nonsigular square matrix $\Phi \neq I_{p_Y} \in \mathbb{R}^{p_Y\times p_Y}$ such that $\tilde{\mathcal{W}}_j = \mathcal{W}_j\Phi$. Apply the columns of $\tilde{\mathcal{W}}_j$ to the above *Case 1*, we have the Rayleigh-Ritz pair $(\lambda^{(i)}, \tilde{\mathcal{W}}_j\tilde{y}_i) = (\lambda^{(i)}, \mathcal{W}_j\Phi\tilde{y}_i) = (\lambda^{(i)}, \mathcal{W}_j y_i)$ (with $y_j = \Phi\tilde{y}_i$) satisfies

  $$\forall w \in \mathrm{Range}(\tilde{\mathcal{W}}_j) \quad w^T\,(A\tilde{\mathcal{W}}_j\tilde{y}_i - \lambda^{(i)}\, B\tilde{\mathcal{W}}_j\tilde{y}_i) = 0,$$

  which is equivalent to

  $$\tilde{\mathcal{W}}_j^T(A\tilde{\mathcal{W}}_j\tilde{y}_i - \lambda^{(i)}\, B\tilde{\mathcal{W}}_j\tilde{y}_i) = \Phi^T\mathcal{W}_j^T\,(A\mathcal{W}_j\Phi\tilde{y}_i - \lambda^{(i)} B\mathcal{W}_j\Phi\tilde{y}_i) = 0 \text{ because of } \tilde{\mathcal{W}}_j = \mathcal{W}_j\Phi.$$

  Furthermore, given the nosingularity of Φ and $y_j = \Phi\tilde{y}_i$, the above equation could be reformed as

  $$\mathcal{W}_j^T\,(A\mathcal{W}_j y_i - \lambda^{(i)} B\mathcal{W}_j y_i) = 0 \text{ with } \mathcal{W}_j^T B\mathcal{W}_j = (\Phi\Phi^T)^{-1}. \tag{2.6}$$

  Substituting Equation (2.1) into the left equation of (2.5) or (2.6) leads to the formulation (2.2) with Gram matrices G_A and G_B with form described in (2.3) and (2.4), respectively.

□

Repeat the above processes described in Proposition 1, some eigenvalues (like the smallest, largest, rightmost etc.,) $\lambda^{(i)}$ of the relatively smaller eigenvalue problem (2.2) in $\mathbb{R}^{p_Y \times p_Y}$ gradually approximate to some eigenvalues of the original larger one (1.1) in $\mathbb{R}^{n\times n}$ ($p_Y \lll n$). Assume the first p computed eigenvalues of (2.2) are $\lambda^{(1)}, \ldots, \lambda^{(p)}$ (with $\lambda^{(1)} \leq \ldots \leq \lambda^{(p)}$), then the corresponding eigenvectors are denoted as

$$Y = [y_1, \ldots, y_p] = \begin{bmatrix} Y_X \\ Y_R \\ Y_P \end{bmatrix} = [Y_X; Y_R; Y_P] \in \mathbb{R}^{p_{Y_J} \times p}. \tag{2.7}$$

According to the original LOBPCG method [15,17,18], with these updated Rayleigh-Ritz pairs $(\lambda^{(i)}, \mathcal{W}_j y_i)$ and the trial subspace $\mathcal{W}_j$ (2.1), the approximated eigenpairs (Λ_j, X_j) of (1.1) and other recycling tall and skinny matrices for the next iteration are formed as

$$\begin{aligned} X^{new} &= \mathcal{W}_j Y = X_j Y_X + R_j Y_R + P_j Y_P, && (2.8)\\ \Lambda^{new} &= \mathrm{Diag}(\lambda^{(1)}, \ldots, \lambda^{(p)}) = (X^{new}, AX^{new})(X^{new}, BX^{new})^{-1} && (2.9)\\ R^{new} &= AX^{new} - BX^{new}\Lambda^{new}, && (2.10)\\ P^{new} &= R_j Y_R + P_j Y_P = X^{new} - X_j Y_X\ , && (2.11) \end{aligned}$$

where the updated matrices R^{new} and P^{new} gradually converge to zero along the iteration.

The main computational cost of LOBPCG in one iteration comes from the solving of the smaller generalized eigenvalue problem (2.2), which includes the cost in forming the Gram matrices G_A in (2.3) and G_B in (2.4), and the cost in computing the other recycling tall and skinny matrices listed in (2.8)-(2.11).

Note that so far, we have not made any further specific assumption about the definition of trial subspace for this RR process except that it has full-column rank. In the following sections, we will describe how to define a properly trial subspace from different aspects to further reduce these computational costs. To simplify the notation for the results of matrix-matrix multiplication, we define $W_{jA} = AX_j$, $W_{jB} = BX_j$, $Z_{jA} = AR_j$, $Z_{jB} = BR_j$, $Q_{jA} = AP_j$, $Q_{jB} = BP_j$ for the remainder of this manuscript.

2.1.1. The trial subspace with B-orthonormal block vectors. To reduce the cost of solving the smaller eigenvalue problem (2.2), in this section, we describe how to generate a B-orthonormal basis of the columns of block vectors $X_j \in \mathbb{R}^{n\times p}$, $R_j \in \mathbb{R}^{n\times p}$ and $P_j \in \mathbb{R}^{n\times p}$, respectively, for composing a trial subspace (2.1).

Let $\tilde{X}_j$ with tilde denotes the eigenvectors before B-orthonormalization, and the same rule applies to residuals $\tilde{R}_j$ and search directions $\tilde{P}_j$. For the B-orthonormalization of the columns of $\tilde{X}_j$, $\tilde{R}_j$ and $\tilde{P}_j$, the Cholesky decomposition (simplified as $\texttt{chol}(\cdot)$) of these three block vectors has been used in LOBPCG as

$$\begin{aligned} X_j &= \tilde{X}_j T_X^{-1}, W_{jB} = \tilde{W}_{jB} T_X^{-1}, \text{ with } T_X = \texttt{chol}((\tilde{X}_j, \tilde{W}_{jB})), && (2.12)\\ R_j &= \tilde{R}_j T_R^{-1}, Z_{jB} = \tilde{Z}_{jB} T_R^{-1}, \text{ with } T_R = \texttt{chol}((\tilde{R}_j, \tilde{Z}_{jB})), && (2.13)\\ P_j &= \tilde{P}_j T_P^{-1}, Q_{jB} = \tilde{Q}_{jB} T_P^{-1}, Q_{jA} = \tilde{Q}_{jA} T_P^{-1}, \text{ with } T_P = \texttt{chol}((\tilde{P}_j, \tilde{Q}_{jB})) && (2.14) \end{aligned}$$

for ensuring $(X_j, W_{jB}) = I$, $(R_j, Z_{jB}) = I$ and $(P_j, Q_{jB}) = I$. With these B-orthonormalizations and the recycling eigenpair information described from (2.8)-(2.9), the two Gram matrices in (2.3) and (2.4) can be approximated by

$$G_A = \begin{bmatrix} \Lambda_j & (X_j, Z_{jA}) & (X_j, Q_{jA}) \\ (R_j, W_{jA}) & (R_j, Z_{jA}) & (R_j, Q_{jA}) \\ (P_j, W_{jA}) & (P_j, Z_{jA}) & (P_j, Q_{jA}) \end{bmatrix} \in \mathbb{R}^{p_Y \times p_Y} \tag{2.15}$$

with a simplified form the first p principle submatrix, and

$$G_B = \begin{bmatrix} I & (X_j, Z_{jB}) & (X_j, Q_{jB}) \\ (R_j, W_{jB}) & I & (R_j, Q_{jB}) \\ (P_j, W_{jB}) & (P_j, Z_{jB}) & I \end{bmatrix} \in \mathbb{R}^{p_Y \times p_Y} \tag{2.16}$$

with simplified form for the three p block diagonal elements.

Assume $R_j \in \mathbb{R}^{n\times p}$ and $Z_{jB} \in \mathbb{R}^{n\times p}$ required in (2.13) are available, the matrix-matrix multiplication $(R_j, Z_{jB}) = R_j^T Z_{jB}$ in the $\texttt{chol}(\cdot)$ requires $\mathcal{O}((2n-1)p^2)$ floating-point operations. The same operation flops are required for the B-orthonormalization of the eigenvectors X_j shown in (2.12) and search directions P_j presented in (2.14).

Note that except the Cholesky decomposition, the Gram-Schmidt (GS) process [31] that requires $\mathcal{O}(np^2)$ flops in magnitude at each iteration, like the classical GS (CGS), the modified GS (MGS), the CGS process with re-orthogonalization (CGS2), and the MGS2 could also be used for these B-orthonormalization processes described in (2.12)-(2.14).

2.1.2. B-orthogonalize the block residual against the block eigenvector. According to LOBPCG [15, 17, 18], based on the setting of the trial subspace in formulation (2.1), the RR projection details in Definition 1, and the way to update the new block eigenvector described in Equation (2.8), it is easy to know that the block residual R_j is supposed to be B-orthogonal to the block eigenvector X_j (i.e., $R_j \perp_B X_j$). Besides after Equation (2.12), the columns of X_j own a B-orthonormal basis such that $X_j^T W_{jB} = I_p$.

However, we may gradually loss this theoretically B-orthogonalization condition $R_j \perp_B X_j$ in the practical finite precision arithmetic thus causing the instability. As a remedy, in this section, we discuss how to implement an explicit B-orthogonalization of the R_j against X_j. Specifically, define a projector $\mathcal{P}_j = I_n - X_j X_j^T B^T = I_n - X_j {W_{jB}}^T \in \mathbb{R}^{n\times n}$ for each iteration step, then this B-orthogonalization could be formed as a projection or least-square problem

$$\underset{R_j\in\mathbb{R}^{n\times p}}{\operatorname{argmin}} \|R_j\| = \underset{R_j\in\mathbb{R}^{n\times p}}{\operatorname{argmin}} \|\mathcal{P}_j R_j\| = \underset{{W_{jB}}^T R_j\in\mathbb{R}^{p\times p}}{\operatorname{argmin}} \|R_j - X_j {W_{jB}}^T R_j\|, \tag{2.17}$$

which requires $\mathcal{O}((4n-1)p^2)$ floating-point operations such that ${W_{jB}}^T R_j = 0$.

With the B-orthogonality of $R_j \in \mathbb{R}^{n\times p}$ to $X_j \in \mathbb{R}^{n\times p}$, we have $X_j^T Z_{jB} = X_j^T B R_j = 0$ and $R_j^T W_{jB} = R_j^T B X_j = 0$, thus the Gram matrix G_B shown in Equation (2.16) can be further simplified as

$$G_B = \begin{bmatrix} I & 0 & (X_j, Q_{jB}) \\ 0 & I & (R_j, Q_{jB}) \\ (P_j, W_{jB}) & (P_j, Z_{jB}) & I \end{bmatrix} \in \mathbb{R}^{p_Y\times p_Y}. \tag{2.18}$$

2.1.3. Strategy for partial convergence. When several eigenvectors are computed simultaneously, it is often the case that some eigenvectors converge faster than the others. Moreover, since the columns of $R_j \in \mathbb{R}^{n\times p}$ and $P_j \in \mathbb{R}^{n\times p}$ converge to zero at different rates, the block matrices involved in the Cholesky decomposition or the MGS process become extremely poorly scaled. To avoid unnecessary computations and instability caused by partial convergence, one can deflate or remove the column-index of eigenvectors that have already converged within the required tolerance, continuing the iteration with the *activate* (i.e., those that have not yet converged). This approach is known as the locking strategy. Knyazev's LOBPCG [18] outlines two locking strategies for defining a trial subspace, summarized as follows.

- Soft-locking: define $\mathcal{W}_j = \text{Span}\{X_j, R_{j\,J}, P_{j\,J}\} \in \mathbb{R}^{n\times p_Y}$ $(p_Y = p + p_{R_{j\,J}} + p_{P_{j\,J}} \lll n)$,
- Hard-locking: define $\mathcal{W}_j = \text{Span}\{X_{j\,J}, R_{j\,J}, P_{j\,J}\} \in \mathbb{R}^{n\times p_Y}$ $(p_Y = p_{X_{j\,J}} + p_{R_{j\,J}} + p_{P_{j\,J}} \lll n)$,

where the column-index set J is initialized as $\{1, ..., p\}$, and in each iteration, exclude from the index set J the indices that correspond to residual vectors for which the residual norm become smaller than a tolerance. Thus $0 \le p_{R_{j\,J}} \le p$ refers to the number of *activate* columns in R_j, same rule applies for $p_{P_{j\,J}}$ and $p_{X_{j\,J}}$.

According to the detailed remarks on computational and algorithmic aspects presented in Section 2.1 and Section 2.1.1-2.1.3, the associated pseudocode of Knyazev's LOBPCG algorithm is presented in Algorithm 1 [1] with the simplified Gram matrices and soft-locking strategy.

2.1.4. Some further developments in the basis selection in LOBPCG. After wide application of Knyazev's LOBPCG algorithm [17, 18], some researchers noticed that an inappropriate choice of the basis of the trial subspace can lead to ill-conditioned Gram matrices in the RR process that can delay convergence or produce inaccurate eigenpairs. For remedy, [11, 28] proposed a robust version of LOBPCG with different ways of the basis selection for the case of standard eigenvalue problem (i.e., $B = I_n$ in (1.1)). Refer to Appendix A for more of its details.

[1] refer to https://github.com/lobpcg/blopex/blob/master/blopex_tools/matlab/lobpcg/lobpcg.m for the code implementation.

Algorithm 1 *Detailed description of the LOBPCG Algorithm [18, Section 2.2] or [17, Algorithm 5.1]:*

Require: p starting linearly independent vectors in $X_0 \in \mathbb{R}^{n\times p}$, ℓ linearly independent constraint vectors in $Y \in \mathbb{R}^{n\times \ell}$, devices to compute $W_A = AX$, $W_B = BX$ and MX (the $M \in \mathbb{R}^{n\times n}$ is an approximation of the inverse of A) for a given vector X, and define the B-orthonormalized X as $(X, W_B) = I$

1: Let $j = 0$, allocate memory for $X_j, R_j, P_j, W_{jA}, Z_{jA}, Q_{jA} \in \mathbb{R}^{n\times p}$, and $W_{jB}, Z_{jB}, Q_{jB} \in \mathbb{R}^{n\times p}$, $BY \in \mathbb{R}^{n\times \ell}$ when $B \neq I$

2: Apply constraints to X_0: $X_0 = X_0 - Y(Y, BY)^{-1}((BY)^T X_0)$

3: B-orthonormalize X_0 (s.t., $(X_0, W_{jB}) = I$): $T = \texttt{chol}((X_0, W_{0B}))$ or MGS(X_0); $X_0 = X_0 T^{-1}$; $W_{0B} = W_{0B}T^{-1}$; $W_{0A} = A \times X_0$ (Note: "chol" is the Cholesky decomposition)

/ Perform the Rayleigh–Ritz (RR) procedure for the pencil $A - B\Lambda$ with the B-orthonormalized subspace Span$\{X_0\}$ to compute the initial Ritz vectors X_1 and corresponding Ritz values Λ_1: */*

4: Solve eigenproblem $(X_0, W_{0A})t = It\Lambda_1$; and compute $X_1 = X_0 t$; $W_{1A} = W_{0A}t$; $W_{1B} = W_{0B}t$

5: Define the index set J of active iterates to be $\{1, ..., p\}$

6: **for** $j = 1, 2, \ldots, MaxIterations$: **do**

7: Compute the residuals: $R_j = W_{jA} - W_{jB}\Lambda_j$ (we have $R_j \perp X_j$ because of Step 4, 12, 22)

8: Exclude from the index set J the indices that correspond to residual vectors for which the norm has become smaller than the tolerance. If J then becomes empty, exit loop.

9: Compute the active vectors with the update index set J (here length$(J) \leq p$): $R_{j\,J} = R_j(:, J)$, $Q_{jB\,J} = BP_j(:, J) \in \mathbb{R}^{p\times p_{R_{j\,J}}}$; $P_{j\,J} = P_j(:, J)$, $Q_{jA\,J} = AP_j(:, J) \in \mathbb{R}^{p\times p_{P_{j\,J}}}$

10: Apply the preconditioner M to the active residuals: $R_{j\,J} = MR_{j\,J}$

11: Apply the constraints to preconditioned active residuals: $R_{j\,J} = R_{j\,J} - Y(Y, BY)^{-1}((BY)^T R_{j\,J})$

12: B-orthogonalize (i.e., $(W_{jB}, R_{j\,J}) = 0$) preconditioned $R_{j\,J}$ to X_j as Equation (2.17) in Section 2.1.2

13: Compute $Z_{jB\,J}$ as $Z_{jB\,J} = BR_{j\,J}$, and B-orthonormalize $R_{j\,J}$ (s.t., $(R_{j\,J}, Z_{jB\,J}) = I$) as process in Equation (2.13) in Section 2.1.1 ; compute $Z_{jA\,J}$ as $Z_{jA\,J} = AR_{j\,J}$

14: **if** $j > 1$ **then**

15: B-orthonormalize $P_{j\,J}$ (s.t., $(P_{j\,J}, Q_{jB\,J}) = I$) as process in Equation (2.14) in Section 2.1.1

16: **end if**

/ Perform the RR procedure for the pencil $A - B\Lambda$ in the trial space: $\mathcal{W}_j =$ Span$\{X_j, R_{j\,J}, P_{j\,J}\} \in \mathbb{R}^{n\times p_Y}$ $(p_Y = p + p_{R_{j\,J}} + p_{P_{j\,J}} \lll n)$ to update Ritz values Λ and corresponding Ritz vectors X. Compute the symmetric Gram matrices G_A, G_B shown in Equation (2.2):*/*

17: **if** $j > 1$ **then**

18: $G_A \in \mathbb{R}^{p_Y\times p_Y}$ in (2.15) and $G_B \in \mathbb{R}^{p_Y\times p_Y}$ in (2.18) with J for columns of $R_{j\,J}$ and $P_{j\,J}$

19: **else**

20: $G_A = \begin{bmatrix} \Lambda_j & (X_j, Z_{jA\,J}) \\ (R_{j\,J}, W_{jA}) & (R_{j\,J}, Z_{jA\,J}) \end{bmatrix} \in \mathbb{R}^{(p+p_{R_{j\,J}})\times(p+p_{R_{j\,J}})}$, $G_B = I_{p+p_{R_{j\,J}}}$

21: **end if**

22: Solve the generalized eigenvalue problem: $G_A Y = G_B Y \Lambda_{j+1}$, where the first p eigenvalues in increasing order are in the diagonal matrix $\Lambda_{j+1} \in \mathbb{R}^{p\times p}$, and the G_B-orthonormalized eigenvectors are the columns of Y (i.e., $(Y, G_B\, Y) = I, (Y, G_A\, Y) = \Lambda_{j+1}$)

/ Update Ritz vectors and the recycling tall matrices: */*

23: **if** $j > 1$ **then**

24: Partition Y as form (2.7) according to the number of columns in X_j, $R_{j\,J}$, and $P_{j\,J}$

25: Update $P_{j+1} = R_{j\,J}Y_R + P_{j\,J}Y_P$; $Q_{j+1\,A} = Z_{jA\,J}Y_R + Q_{jA\,J}Y_P$; $Q_{j+1\,B} = Z_{jB\,J}Y_R + Q_{jB\,J}Y_P$

26: Update $X_{j+1} = X_j Y_X + P_{j+1}$; $W_{j+1\,A} = W_{jA}Y_X + Q_{j+1\,A}$; $W_{j+1\,B} = W_{jB}Y_X + Q_{j+1\,B}$

27: **else**

28: Partition $Y = \begin{bmatrix} Y_X \\ Y_R \end{bmatrix} \in \mathbb{R}^{(p+p_{R_{j\,J}})\times p}$ according to the number of columns in X_j, $R_{j\,J}$

29: Update $P_{j+1} = R_{j\,J}Y_R$; $Q_{j+1\,A} = Z_{jA\,J}Y_R$; $Q_{j+1\,B} = Z_{jB\,J}Y_R$

30: Update $X_{j+1} = X_j Y_X + P_{j+1}$; $W_{j+1\,A} = W_{jA}Y_X + Q_{j+1\,A}$; $W_{j+1\,B} = W_{jB}Y_X + Q_{j+1\,B}$

31: **end if**

32: B-orthonormalize X_{j+1} (s.t., $(X_{j+1}, W_{j+1B}) = I$) as process in Equation (2.12) in Section 2.1.1

33: **end for**

34: **return** the approximations Λ_{j+1} and X_{j+1} to the smallest eigenvalues and corresponding eigenvectors

Note that we only focus on how to apply randomized sketching to the original Knyazev's LOBPCG. Details of applying randomization to other related LOBPCG variants with modification of basis selection could be deduced similarly (refer to Appendix B for details) thus be omitted in the main manuscript.

2.2. Randomized numerical linear algebra with sketching. Dimension reduction is an elegant idea from computer science that has found applications in numerical linear algebra (NLA), which forms a new research trend called randomized numerical linear algebra (RandNLA). Here is the basic concept: Solving a computational problem in high-dimensional space can often be made more efficient by first transforming the problem into a lower-dimensional space while preserving its essential structure. The randomized embedding is a popular dimension reduction technique. This approach is usually traced to the celebrated paper of Johnson and Lindenstrauss published in 1986 [14], which says that any d-point subset in n dimensions Euclidean space can be embedded onto a random subspace of $t = \mathcal{O}(\log n/\varepsilon^2)$ dimensions without distorting the distances between any pair of inter-points by more than a factor of $(1 \pm \varepsilon)$ with positive probability, for any $0 < \varepsilon < 1$. Around two decades later, some researchers have shown that this result is essentially tight. For example in [8], Dasgupta and Gupta proved another version of this result with a higher bound on $t = \mathcal{O}(4(\varepsilon^2/2 - \varepsilon^3/3)^{-1} \ln n)$. These details are recollected in Theorem 1.

THEOREM 1 (Johnson-Lindenstrauss theorem [8, 14]). *For any $0 < \varepsilon < 1$ and any integer n, let t ($t \ll n$) be a positive integer such that*

$$t \geq \log n/\varepsilon^2,$$

which follows the bound in Johnson and Lindenstrauss's paper [14], or a recent new bound in [8] as

$$t \geq 4(\varepsilon^2/2 - \varepsilon^3/3)^{-1} \ln n.$$

Then, for any subset V of d-point in $\mathbb{R}^n$, there is a linear map $f : \mathbb{R}^n \to \mathbb{R}^t$ such that for all $u,\ v \in V \subset \mathbb{R}^n$,

$$(1 - \varepsilon)\|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \varepsilon)\|u - v\|^2. \tag{2.19}$$

Furthermore, this map can be found in randomized polynomial time.

In the literature of RandNLA, the dimension reduction map f is referred as *subspace embedding* or *random sketching*. Using this terminology, the relation (2.19) conveys that the embedding f should preserve the geometry, such as the norm, distance, or inner product, of the original subset V. However, the subset V is not always known in advance. If the embedding f can be constructed without prior knowledge of V, apart from its dimension d, the embedding f is said to be oblivious and is referred to as an (ε, δ, d)-*oblivious subspace embedding* (OSE) defined in Definition 2. Since dimension transformation can be realized through the application of a matrix, we denote the embedding f as a sketching matrix $\Theta \in \mathbb{R}^{t \times n}$, assuming the embedding dimension t is already known.

DEFINITION 2 (Oblivious Subspace Embedding). *Let n and t be two positive integers and $t \ll n$. Let $0 < \delta < 1$ and $0 < \varepsilon < 1$. The sketching matrix $\Theta \in \mathbb{R}^{t \times n}$ is an (ε, δ, d)-oblivious subspace embedding for the norm $\|\cdot\|$ when for any fixed d-dimensional subspace $V \subset \mathbb{R}^n$,*

$$(1 - \varepsilon)\|x\|^2 \leq \|\Theta x\|^2 \leq (1 + \varepsilon)\|x\|^2 \ \text{ for } \forall x \in V$$

holds with probability at least $1 - \delta$.

In general, the sketching could be realized by unstructured or structured embedding, like the unstructured Gaussian matrices or the subsampled randomized Hadamard transform (SRHT) embeddings. In early work, these (oblivious) subspace embeddings were commonly accomplished with uniformly random projectors, such as an unstructured random Gaussian embedding. It is a random matrix $\Theta \in \mathbb{R}^{t \times n}$ ($t \ll n$) with i.i.d (i.e., "independent and identically distributed") entries $(\Theta)_{i,j} \sim \texttt{NORMAL}(0, t^{-1})$, $1 \leq i \leq t$, $1 \leq j \leq n$. The cost of explicitly storing unstructured Gaussian embedding is $\mathcal{O}(tn)$, and the cost of applying it to a matrix, like project $X \in \mathbb{R}^{n \times p}$ to ΘX, is $\mathcal{O}(tnp)$. Gaussian embeddings work extremely well. However, they are not suitable for practical applications because they are expensive to construct, to store, and to apply to a matrix or vector. Instead, one may prefer to implement more structured embedding matrices that alleviate these burdens. The SRHT technique introduced by Woolfe et. al., [37] is one type of structured dimension reduction map, which contains a subset of the columns of a randomized fast Walsh–Hadamard matrix as described in the following Definitions 3 and 4.

DEFINITION 3 (normalized Walsh–Hadamard matrix [7]). *Fix an integer $n = 2^p$ for $p = 1, 2, 3, \ldots$. The (nonnormalized) $n \times n$ matrix of the Walsh–Hadamard transform is defined recursively as,*

$$\boldsymbol{H}_n = \begin{bmatrix} \boldsymbol{H}_{n/2} & \boldsymbol{H}_{n/2} \\ \boldsymbol{H}_{n/2} & -\boldsymbol{H}_{n/2} \end{bmatrix} \text{ with } \boldsymbol{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The $n \times n$ normalized matrix of the Walsh–Hadamard transform is equal to $\boldsymbol{H} = \frac{1}{\sqrt{n}}\boldsymbol{H}_n \in \mathbb{R}^{n\times n}$.

DEFINITION 4 (Subsampled Randomized Hadamard Transform (SRHT) matrix [7]). *Fix integers t and $n = 2^p$ with $t < n$ and $p = 1, 2, 3, \ldots$. An SRHT sketching $\Theta \in \mathbb{R}^{t\times n}$ is a matrix of the form*

$$\Theta = \sqrt{\frac{n}{t}}\boldsymbol{RHD}; \tag{2.20}$$

- $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ *is a random diagonal matrix whose entries are independent random signs, i.e., random variables uniformly distributed on ± 1;*
- $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ *is a normalized Walsh–Hadamard matrix;*
- $\boldsymbol{R} \in \mathbb{R}^{t\times n}$ *is a subset of t rows from the $n \times n$ identity matrix, where the rows are chosen uniformly at random and without replacement.*

Note that other alternative form of SRHT is available as well. Such as the Fast Johnson-Lindenstrauss Transform (FJLT) [1], which was realized by Ailon and Chazelle with a randomized Fourier transform (i.e., the change in **H** in the right-hand side of Equation (2.20).

In practical applications, the SRHT-based low-rank matrix approximation technique is of particular interest because the highly structured nature of map $\Theta \in \mathbb{R}^{t\times n}$ can be exploited to reduce the time of computing ΘX from $\mathcal{O}(tnp)$ to $\mathcal{O}(np\log_2(n))$, or a much sharper bound $\mathcal{O}(np\log_2(t+1))$ as described in [2, 7]. For this type of structured SRHT embedding map Θ, Tropp demonstrated that it preserves the Euclidean geometry of an entire subspace of vectors in $\mathbb{R}^{n\times p}$ if the embedding dimension $t \sim plog(p)/\varepsilon^2$ [35, Theorem 1.3, Theorem 3.1, and Theorem 3.2]. This is the essential ingredient required to show that the SRHT map can be used in the deterministic algorithms for RandNLA.

Note that we only consider the SRHT embedding with the form described in (2.20) as the sketching matrix Θ for the rest of this manuscript.

2.3. The sketched LOBPCG algorithm. The RandNLA methods based on sketching fall into three rough categories: (1) *sketch-and-solve*, (2) *iterative-sketching*, and (3) *sketch-and-precondition*. In this work, we only focus on applying the first two categories to Knyazev's LOBPCG method [17, 18] to deduce its randomized variant denoted as sLOBPCG. Specifically, the sketching technique for approximating the matrix-matrix multiplication [7], which appears in orthonormalization, is described in Section 2.3.1, and sketching for solving the least-square problems [21, Section 10] is illustrated in Section 2.3.2. The sLOBPCG method with the standard Rayleigh-Ritz procedure is summarized in Section 2.3.3, followed by a preliminary discussion on convergence analysis in Section 2.3.5. Finally, additional remarks on the sLOBPCG method with the sketched Rayleigh-Ritz procedure are presented in Section 2.3.6.

2.3.1. Sketching for the B-orthonormalization. In this section, we apply sketching to parts of the matrix-matrix multiplication appeared in the B-orthonormalization described in Section 2.1.1. Without loss of generality, we use Equation (2.13) as an example for illustration. Within the traditional NLA area, the matrix-matrix multiplication $R_j^T Z_{jB}$ of matrix $R_j \in \mathbb{R}^{n\times p}$ requires at most $(2n-1)p^2$ flop operations. These computations are deterministic, i.e., ensure that the solution of the underlying problem is returned after the corresponding operation count.

Although these algorithms are numerically stable and run in polynomial time, $\mathcal{O}(np^2)$ arithmetic operations can be prohibitive for many applications when the size of the matrix is large, e.g., on the order of millions or billions. One way to speed up these algorithms is to reduce the size of R_j, and then apply standard deterministic procedures to the embedding matrix. In more detail, for an SRHT matrix $\Theta \in \mathbb{R}^{t\times n}$ $(n > t = o(n))$, let matrix $Y = \Theta R_j = {R_j}^{\Theta} \in \mathbb{R}^{t\times p}$, $Y_B = \Theta Z_{jB} = {Z_{jB}}^{\Theta} \in \mathbb{R}^{t\times p}$ with a superscript $^{\Theta}$ denote the sketched matrix. According to Section 2.2, Θ is referred to as the dimension reduction matrix, ensuring that Y and Y_B retain as much information from R_j and Z_{jB} as possible. Consider applying sketching Θ for the matrix-matrix multiplication operation mentioned above, we rewrite

the B-orthonormalization parts formed in Equation (2.12)-(2.14) as

$$X_j = \tilde{X}_j T_X^{-1}, W_{jB} = \tilde{W}_{jB} T_X^{-1}, \text{ with } T_X = \texttt{chol}((\tilde{X}_j{}^{\Theta}, \tilde{W}_{jB}{}^{\Theta})), \tag{2.21}$$

$$R_j = \tilde{R}_j T_R^{-1}, Z_{jB} = \tilde{Z}_{jB} T_R^{-1}, \text{ with } T_R = \texttt{chol}((\tilde{R}_j{}^{\Theta}, \tilde{Z}_{jB}{}^{\Theta})), \tag{2.22}$$

$$P_j = \tilde{P}_j T_P^{-1}, Q_{jB} = \tilde{Q}_{jB} T_P^{-1}, Q_{jA} = \tilde{Q}_{jA} T_P^{-1}, \text{ with } T_P = \texttt{chol}((\tilde{P}_j{}^{\Theta}, \tilde{Q}_{jB}{}^{\Theta})) \tag{2.23}$$

for ensuring $(X_j{}^{\Theta}, W_{jB}{}^{\Theta}) = I$, $(R_j{}^{\Theta}, Z_{jB}{}^{\Theta}) = I$ and $(P_j{}^{\Theta}, Q_{jB}{}^{\Theta}) = I$. Note that in this case $(X_j, W_{jB}) \neq I$, $(R_j, Z_{jB}) \neq I$ and $(P_j, Q_{jB}) \neq I$. Besides the sketched B-orthogonalization processes described in Equation (2.21)-(2.23) is referred as B-Θ-orthogonalization, which can also be realized by the randomized Gram-Schmidt (RGS) process [4].

In this B-Θ-orthogonalization setting, one can compute $Y^T Y_B$ instead of $R_j^T Z_{jB}$. If Θ is chosen carefully (in terms of unstructured or structured embedding and the embedding dimension), then $Y^T Y_B = (R_j{}^{\Theta})^T Z_{jB}{}^{\Theta} \approx R_j^T Z_{jB}$ and the number of operations needed to compute $Y^T Y_B$ is at most $(2t-1)p^2$ or $o(np^2)$. As a supplement, refer to [7, Lemma 4.11] [38] for the bound of developing an approximated matrix multiplication method based on dimension embedding.

2.3.2. Sketching for the least-square problem. Overdetermined least-square problems sometimes arise in statistics and data-analysis applications. We may imagine that some of the data in these problems is redundant. As such, we could reduce the size of the problem to accelerate computation without too much loss in accuracy. Instead of the classical approach such as MGS at a cost of $(4n-1)p^2$ (i.e., $\mathcal{O}(np^2)$) arithmetic operations, in this section, we apply a sketching map to the least-square problem described in Equation (2.17). Specifically, the *sketch-and-solve* paradigm maps the least-square problem into a lower dimension. Then, its solution is used as a proxy for the solution of the original problem. This approach can be very fast, and we only need one sketching for the block vector R_j, X_j and W_{jB}.

Let $\Theta \in \mathbb{R}^{t\times n}$ be a sketching for $\text{Span}\{R_j, X_j\} \in \mathbb{R}^{n\times 2p}$ ($p < t \ll n$) with distortion ε. From the B-Θ-orthonormalization process for eigenvectors $X_j{}^{\Theta} \in \mathbb{R}^{t\times p}$ described in Equation (2.21), we have $X_j{}^{\Theta T} W_{jB}{}^{\Theta} = I_p$. Define $\widetilde{\mathcal{P}}_j{}^{\Theta} = I_t - X_j{}^{\Theta} W_{jB}{}^{\Theta T} \in \mathbb{R}^{t\times t}$ and $\mathcal{P}_j{}^{\Theta} = I_n - X_j W_{jB}{}^{\Theta T}\Theta = I_n - X_j W_{jB}{}^{T}\Theta^T\Theta \in \mathbb{R}^{n\times n}$, we consider a lower-dimensional least-square problem as

$$\underset{R_j{}^{\Theta}\in\mathbb{R}^{t\times p}}{\text{argmin}} \|R_j{}^{\Theta}\| = \underset{R_j{}^{\Theta}\in\mathbb{R}^{t\times p}}{\text{argmin}} \|\widetilde{\mathcal{P}}_j{}^{\Theta} R_j{}^{\Theta}\| = \underset{W_{jB}{}^{\Theta T} R_j{}^{\Theta}\in\mathbb{R}^{p\times p}}{\text{argmin}} \|R_j{}^{\Theta} - X_j{}^{\Theta} W_{jB}{}^{\Theta T} R_j{}^{\Theta}\|. \tag{2.24}$$

Then the least-square problem related to residuals in original high-dimensional space could be formed as

$$\underset{R_j\in\mathbb{R}^{n\times p}}{\text{argmin}} \|R_j\| = \underset{R_j\in\mathbb{R}^{n\times p}}{\text{argmin}} \|\mathcal{P}_j{}^{\Theta} R_j\| = \underset{W_{jB}{}^{\Theta T} R_j{}^{\Theta}\in\mathbb{R}^{p\times p}}{\text{argmin}} \|R_j - X_j W_{jB}{}^{\Theta T} R_j{}^{\Theta}\|. \tag{2.25}$$

The projection processes described in Equation (2.24)-(2.25) can be viewed as a paired results that requires $(2n+2t-1)p^2$ (i.e., $\mathcal{O}(np^2)$) floating-point operations such that $W_{jB}{}^{\Theta T} R_j{}^{\Theta} = 0$. The Equation (2.24) in lower dimension is the sketched variant of the original least-square problem (2.17). Since Θ preserves geometry, we hope that the solution to the sketched problem (2.24) (like $W_{jB}{}^{\Theta T} R_j{}^{\Theta}$) can mimic or approximate the solution to the original problem (2.17) (like $W_{jB}{}^{T} R_j$). The theoretical bound for the distance between the deterministic least-square problem and the sketched one could be deduced similarly by the process described in [5, Section 2.1] [7, Lemma 5.6] and [21, Section 10.3].

The *sketch-and-solve* paradigm requires us to form matrix $X_j{}^{\Theta} = \Theta X_j$ at a cost around $\mathcal{O}(np\log(t))$ operations. We can solve the (dense) reduced problem with a direct method, using $\mathcal{O}(tp^2)$ operations. Assuming $t \sim p\log(p)/\varepsilon^2$, the total arithmetic cost is $\mathcal{O}(np\log(p/\varepsilon^2) + p^3\log(p)/\varepsilon^2)$.

In summary, we witness an improvement in computational cost over classical methods since $\log(p) \ll p \ll n/\log(p)$ and ε is constant. While we have to accept large errors if only one step of sketching described in Equation (2.24) is applied to the whole iteration processes of sLOBPCG. Alternatively, *iterative-sketching* attempts to remediate such poor accuracy by applying sketching recursively at each iteration. That is for each iteration j, we draw a fresh random subspace embedding $\Theta_j \in \mathbb{R}^{t\times n}$ (rather than a fixed one Θ) for $\text{Span}\{R_j, X_j\}$, with constant distortion for the least-square problem (2.25). In this setting, *iterative-sketching* costs more than the *sketch-and-solve* paradigm to achieve better accuracy because of the repeated sketches of the block vectors at each iteration. Here recall that this *iterative-sketching* is also applied to the sketched B-Θ-orthonormalization process described in Section 2.3.1.

2.3.3. sLOBPCG with the normal Rayleigh-Ritz procedure. If the normal Rayleigh-Ritz (RR) procedure is applied to sLOBPCG with the aforementioned Section 2.3.1-2.3.2, then the Gram matrix G_B in Equation (2.18) switches back to the dense form in Equation (2.4) because of $X_j^T Z_{jB} \neq 0$, $R_J^T W_{jB} \neq 0$, $X_j^T W_{jB} \neq I$, $R_j^T Z_{jB} \neq I$ and $P_J^T Q_{jB\,J} \neq I$. The Gram matrix G_A in sLOBPCG keeps the form in Equation (2.3). For this sLOBPCG with normal RR procedure applied on the trial subspace $\mathcal{W}_j$ defined in Equation (2.1), if we only consider the first p rows of the computed block eigenvector Y partitioned in Equation (2.7), we have

$$(X_j^T A X_j) Y_X = (X_j^T B X_j)(Y_X \Lambda_{j+1}) \tag{2.26}$$

with ${X_j}^{\Theta^T} {W_{jB}}^{\Theta} = (\Theta X_j)^T (\Theta B X_j) = I_p$ and $X_j \in \mathbb{R}^{n\times p}$, $Y_X, \Lambda_{j+1} \in \mathbb{R}^{p\times p}$. By repeatedly applying the normal Rayleigh-Ritz process to the trial subspace $\mathcal{W}_j \in \mathbb{R}^{n\times p_Y}$ $(p < p_Y \lll n)$, we aim to refine Equation (2.26) such that the first p eigenpairs of $\mathcal{W}_j^T A \mathcal{W}_j \in \mathbb{R}^{p_Y \times p_Y}$ are as close as possible to those of $A \in \mathbb{R}^{n\times n}$. In other words, these eigenpairs are well approximated through this normal RR process even with the B-Θ-orthonormal X_j. Note that Equation (2.26) still holds for LOBPCG recalled in Section 2.1 but with $X_j^T W_{jB} = X_j^T B X_j = I_p$, i.e., the columns of X_j are B-orthonormal.

In the case of sLOBPCG, the trial subspace defined in Equation (2.1) remains full-column rank. Initially, the columns of the block eigenvecor X_0 are B-orthonormal, as in LOBPCG. However, for subsequent iterations (i.e., $j > 1$), the columns of the block eigenvecor X_j, block residual R_j and block search direction P_j become B-Θ-orthonormal, as described in Equation (2.21)-(2.23), respectively.

Furthermore, from Equation (2.21)-(2.25), we observe that six sketching operations are required for each iteration step. However, if we assume $B = I$, only three sketching operations are required. That is the sketching for the block eigenvector, block residual and block search direction, which respectively writes as ${X_j}^{\Theta} = \Theta X_j$, ${R_j}^{\Theta} = \Theta R_j$, and ${P_j}^{\Theta} = \Theta P_j$. Alternatively, instead of computing all the three sketched block vectors, we can do one sketching to block residual and then explicitly update the other two sketched results as

$$P^{\Theta^{new}} = {R_j}^{\Theta} Y_R + {P_j}^{\Theta} Y_P, \tag{2.27}$$

$$X^{\Theta^{new}} = {X_j}^{\Theta} Y_X + P^{\Theta^{new}}, \tag{2.28}$$

to further reduce the cost caused by applying sketchings. Combining Equation (2.27)-(2.28) with Equation (2.8)-(2.11), we have the new block vectors required for the next iteration of the sLOBPCG algorithm. Note that the case for $B \neq I$ could be similarly derived and is therefore omitted here. The sLOBPCG algorithm with three (or six when $B \neq I$) explicitly sketchings for each iteration step is referred as sLOBPCG$_{\text{e-s}}$, where the "e-s" means applying explicitly sketching to the six involved block vectors discussed above. To facilitate the distinction, another sketched variant with one (or two when $B \neq I$) explicit sketching for residuals and other two (or four when $B \neq I$) implicit sketchings as described in Equation (2.27)-(2.28) is denoted as sLOBPCG alternatively.

Theoretically, these two sketched LOBPCG variants, differing in the frequency of applying sketchings at each iteration step, are mathematically equivalent. Our experiments, however, revealed that the sLOBPCG variant may be less stable than sLOBPCG$_{\text{e-s}}$ in terms of the final required iteration steps. On the other hand, the sLOBPCG variant is faster than sLOBPCG$_{\text{e-s}}$ in terms of implementation time. A similar observation was made in [4, Remark 2.10] for the randomized Gram-Schmidt process, which can involve one or two sketching operations per Gram-Schmidt step.

The associated pseudocode of sLOBPCG$_{\text{e-s}}$ and sLOBPCG are respectively presented in Algorithm 2 and Algorithm 3 with the dense Gram matrices in form (2.3) and (2.4) and with the soft-locking strategy discussed in Section 2.1.3.

According to Section 2.1.1-2.1.2 of LOBPCG and Section 2.3.1-2.3.3 of sLOBOCG variants, we know that the Cholesky decomposition (simplified as `chol`$(\cdot)$), the Gram-Schmidt (GS) process, like the classical GS (simplified as CGS$(\cdot)$), the modified GS (simplified as MGS$(\cdot)$), and the randomized GS (simplified as RGS$(\cdot)$) can be used for the orthonormalization processes of related block vectors. For easy comparison, we summarize the orthonormalization processes of LOBPCG and sLOBPCG variants as follows:

- LOBPCG:
  - B-orthogonalization of X_j, R_j, P_j in Section 2.1.1: `chol`$(\cdot)$, CGS$(\cdot)$, MGS$(\cdot)$;
  - B-orthogonalize R_j to X_j Section 2.1.2: CGS$(\cdot)$, MGS$(\cdot)$;

Algorithm 2 *sLOBPCG$_{e\text{-}s}$: sketched LOBPCG algorithm with Rayleigh-Ritz (RR) procedure and explicitly computed sketched vectors*

Require: As the **Require** described in the Algorithm 1, and set a sketch matrix $\Theta \in \mathbb{R}^{t\times n}$ ($t = \min(n,\ 2m\log(n)/\log(m),\ 3\times m)$, here the m is the length of the search space), the sketched inner vector products as $(X,Y)_\Theta = (X^\Theta, Y^\Theta)$, and define the B-Θ-orthonormalized X as $(X, BX)_\Theta = I$

Step 1-5 of the Algorithm 1
for $j = 1, 2, \ldots, MaxIterations$: **do**
  Step 7-11 of the Algorithm 1
  Compute the sketched active eigenvector ${X_j}^\Theta$, and ${W_{jB}}^\Theta$, and then B-Θ-orthonormalize X_j (s.t., $(X_j, W_{jB})_\Theta = I$) as process in Equation (2.21) in Section 2.3.1
  B-Θ-orthogonalize the preconditioned active residuals to ${X_j}^\Theta$ as Equation (2.25) in Section 2.3.2
  Compute Z_{jB_J} as $Z_{jB_J} = BR_{j_J}$, compute the sketched preconditioned active residuals ${R_{j_J}}^\Theta$ and ${Z_{jB_J}}^\Theta$, and then B-Θ-orthonormalize R_{j_J} (s.t., $(R_{j_J}, Z_{jB_J})_\Theta = I$) as process in Equation (2.22) in Section 2.3.1. Compute Z_{jA_J}: $Z_{jA_J} = AR_{j_J}$
  if $j > 1$ **then**
    Compute the sketched active residuals ${P_{j_J}}^\Theta$ and ${Q_{jB_J}}^\Theta$, and then B-Θ-orthonormalize P_{j_J} (s.t., $(P_{j_J}, Q_{jB_J})_\Theta = I$) as process in Equation (2.23) in Section 2.3.1
  end if
  / Perform the RR Procedure for the pencil $A - B\Lambda$ in the B-orthonormalized and B-Θ-orthonormalized subspace: $\mathcal{W}_j = Span\{X_j, R_{j_J}, P_{j_J}\} \in \mathbb{R}^{n\times p_Y}$ ($p_Y = p + p_{R_{j_J}} + p_{P_{j_J}} \lll n$) to compute/update the Ritz values Λ and the corresponding Ritz vectors X. */*
  Compute symmetric Gram matrices of size $\mathbb{R}^{p_{Y_J}\times p_{Y_J}}$ (p_{Y_J} equals to $p + p_{R_J} + p_{P_J}$ or $p + p_{R_J}$):
  if $j > 1$ **then**
    $G_A \in \mathbb{R}^{p_Y\times p_Y}$ in Equation (2.3) and $G_B \in \mathbb{R}^{p_Y\times p_Y}$ in Equation (2.4) with J for columns of R_{j_J} and P_{j_J}
  else
    With J for columns of R_{j_J}, $G_A = \begin{bmatrix} (X_j, W_{jA_J}) & (X_j, Z_{jA_J}) \\ (R_{j_J}, W_{jA_J}) & (R_{j_J}, Z_{jA_J}) \end{bmatrix} \in \mathbb{R}^{(p+p_{R_{j_J}})\times(p+p_{R_{j_J}})}$, $G_B = \begin{bmatrix} (X_j, W_{jB_J}) & (X_j, Z_{jB_J}) \\ (R_{j_J}, W_{jB_J}) & (R_{j_J}, Z_{jB_J}) \end{bmatrix} \in \mathbb{R}^{(p+p_{R_{j_J}})\times(p+p_{R_{j_J}})}$
  end if
  Solve the generalized eigenvalue problem: $G_A Y = G_B Y \Lambda_{j+1}$, where the first p eigenvalues in increasing order are in the diagonal matrix $\Lambda_{j+1} \in \mathbb{R}^{p\times p}$, and the G_B-orthonormalized eigenvectors are the columns of Y (i.e., $(Y, G_B\, Y) = I, (Y, G_A\, Y) = \Lambda_{j+1}$)
  / Update Ritz vectors and the recycling tall matrices: */*
  if $j > 1$ **then**
    Partition $Y \in \mathbb{R}^{p_{Y_J}\times p}$ as form (2.7) according to the number of columns in X_j, R_{j_J}, and P_{j_J}
    Update $P_{j+1} = R_{j_J}Y_R + P_{j_J}Y_P$; $Q_{j+1_A} = Z_{jA_J}Y_R + Q_{jA_J}Y_P$; $Q_{j+1_B} = Z_{jB_J}Y_R + Q_{jB_J}Y_P$
    Update $X_{j+1} = X_jY_X + P_{j+1}$; $W_{j+1_A} = W_{jA}Y_X + Q_{A_{j+1}}$; $W_{j+1_B} = W_{jB}Y_X + Q_{B_{j+1}}$
  else
    Partition $Y = \begin{bmatrix} Y_X \\ Y_R \end{bmatrix} \in \mathbb{R}^{(p+p_{R_{j_J}})\times p}$ according to the number of columns in X, R_J
    Update $P_{j+1} = R_{j_J}Y_R$; $Q_{j+1_A} = Z_{jA_J}Y_R$; $Q_{j+1_B} = Z_{jB_J}Y_R$
    Update $X_{j+1} = X_jY_X + P_{j+1}$; $W_{j+1_A} = W_{jA}Y_X + Q_{jA}$; $W_{j+1_B} = W_{jB}Y_X + Q_{jB}$
  end if
end for
return the approximations Λ_{j+1} and X_{j+1} to the smallest eigenvalues and corresponding eigenvectors

- sLOBPCG variants with normal Rayleigh-Ritz procedure:
  - B-Θ-orthogonalization X_j R_j, P_j in Section 2.3.1: $\texttt{chol}(\cdot)$, RGS$(\cdot)$;
  - B-Θ-orthogonalize R_j to X_j Section 2.3.2: RGS$(\cdot)$.

2.3.4. Comparison with LOBPCG in terms of the flops within a single iteration. For comparison with LOBPCG in terms of computational cost, Table 2.1 describes the difference between LOBPCG and

Algorithm 3 *sLOBPCG: Variant of sLOBPCG$_{e\text{-}s}$ with RR and implicitly computed sketched vectors*

Require: As the **Require** described in the Algorithm 2

1: Step 1-5 of the Algorithm 1
2: Compute ${X_1}^{\Theta} = \Theta X_1,\ {W_1}_B{}^{\Theta} = \Theta W_{1B} \in \mathbb{R}^{t\times p}$
3: Define the index set J of active iterates to be $\{1, ..., p\}$
4: **for** $j = 1, 2, \ldots, MaxIterations$: **do**
5: Step 7-11 of the Algorithm 1
6: B-Θ-orthonormalize X_j (s.t., $(X_j, W_{jB})_\Theta = I$) as process in Equation (2.21) in Section 2.3.1
7: Compute the sketched preconditioned active residuals ${R_{jJ}}^{\Theta} = \Theta R_{jJ}$, and then B-Θ-orthogonalize preconditioned active residuals R_{jJ} to subspace ${X_j}^{\Theta}$: compute ${R_{jJ}}^{\Theta} = {R_{jJ}}^{\Theta} - {X_j}^{\Theta}({W_{jB}}^{\Theta}, {R_{jJ}}^{\Theta})$, and update $R_{jJ} = R_{jJ} - X_j({W_{jB}}^{\Theta}, {R_{jJ}}^{\Theta})$
8: Compute Z_{jBJ} as $Z_{jBJ} = BR_{jJ}$, and sketch Z_{jBJ} as ${Z_{jBJ}}^{\Theta} = \Theta Z_{jBJ}$, and then B-Θ-orthonormalize R_{jJ} (s.t., $(R_{jJ}, Z_{jBJ})_\Theta = I$) as process in Equation (2.22) in Section 2.3.1. Update ${R_{jJ}}^{\Theta} = {R_{jJ}}^{\Theta}\, T^{-1}$, ${Z_{jBJ}}^{\Theta} = {Z_{jBJ}}^{\Theta}\, T^{-1}$ and compute Z_{jAJ}: $Z_{jAJ} = AR_{jJ}$
9: **if** $j > 1$ **then**
10: B-Θ-orthonormalize P_{jJ} (s.t., $(P_{jJ}, Q_{jBJ})_\Theta = I$) as process in Equation (2.23) in Section 2.3.1. Update ${P_{jJ}}^{\Theta} = {P_{jJ}}^{\Theta}\, T^{-1}$, ${Q_{jB}}^{\Theta} = {Q_{jB}}^{\Theta}\, T^{-1}$, ${Q_{jAJ}}^{\Theta} = {Q_{jAJ}}^{\Theta}\, T^{-1}$
11: **end if**
12: Step 10-15 of Algorithm 2: Perform the RR Procedure to solve a generalized eigenvalue problem
/ Update Ritz vectors and the recycling tall matrices: */*
13: **if** $j > 1$ **then**
14: Partition $Y \in \mathbb{R}^{p_{Y_J}\times p}$ as form (2.7) according to the number of columns in X_j, R_{jJ}, and P_{jJ}
15: Compute and store ${P_{j+1}}^{\Theta} = {R_{jJ}}^{\Theta} Y_R + {P_J}^{\Theta} Y_P$, ${Q_{j+1B}}^{\Theta} = {Z_{jBJ}}^{\Theta} Y_R + {Q_{jBJ}}^{\Theta} Y_P$ (for Step 8-10); ${X_{j+1}}^{\Theta} = {X_j}^{\Theta} Y_X + {P_{j+1}}^{\Theta}$, ${W_{j+1B}}^{\Theta} = {W_{jB}}^{\Theta} Y_X + {Q_{j+1B}}^{\Theta}$ (for Step 6)
16: Update $P_{j+1} = R_{jJ} Y_R + P_{jJ} Y_P$, $Q_{j+1A} = Z_{jAJ} Y_R + Q_{jAJ} Y_P$, $Q_{j+1B} = Z_{jBJ} Y_R + Q_{jBJ} Y_P$
17: Update $X_{j+1} = X_j Y_X + P_{j+1}$, $W_{j+1A} = W_{jA} Y_X + Q_{j+1A}$, $W_{j+1B} = W_{jB} Y_X + Q_{j+1B}$
18: **else**
19: Partition $Y = \begin{bmatrix} Y_X \\ Y_R \end{bmatrix} \in \mathbb{R}^{(p+p_{R_{jJ}})\times p}$ according to the number of columns in X_j and R_{jJ}
20: Compute and store ${P_{j+1}}^{\Theta} = {R_{jJ}}^{\Theta} Y_R$, ${Q_{j+1B}}^{\Theta} = {Z_{jBJ}}^{\Theta} Y_R$ (for Step 8-10); ${X_{j+1}}^{\Theta} = {X_j}^{\Theta} Y_X$, ${W_{j+1B}}^{\Theta} = {W_{jB}}^{\Theta} Y_X$ (for Step 6)
21: Update $P_{j+1} = R_{jJ} Y_R$; $Q_{j+1A} = Z_{jAJ} Y_R$; $Q_{j+1B} = Z_{jBJ} Y_R$
22: Update $X_{j+1} = X_j Y_X + P_{j+1}$, $W_{j+1A} = W_{jA} Y_X + Q_{j+1A}$, $W_{j+1B} = W_{jB} Y_X + Q_{j+1B}$
23: **end if**
24: **end for**
25: **return** the approximations Λ_{j+1} and X_{j+1} to the smallest eigenvalues and corresponding eigenvectors

sLOBPCG in terms of the floating-point operations and stored vectors within a single iteration step. From Table 2.1 we noticed that the sLOBPCG variant with normal Rayleigh-Ritz procedure requires more flops than LOBPCG within one iteration. The extra cost mainly comes from sketching, which includes the cost of constructing a sketching matrix and further applying it to the block vectors. For this, some block SRHT variants have been devised in [3] for parallel computing to further reduce the sketching cost. We emphasize that the main focus of the current work is to discuss the details of realizing sLOBPCG. However, its parallel implementation with this block SRHT technique [3] should be considered in the future. Additionally, this extra cost arises from the fact that we cannot leverage the sketched orthogonalization properties to simplify certain matrix-matrix multiplications when constructing the Gram matrix G_B as shown in Equation (2.4).

Furthermore, we observed that the computational flops cost for the LOBPCG and sLOBPCG variants is roughly the same for each iteration, provided we do not account for the inevitable cost introduced by sketching. Generally, one can expect that the benefits of sketching, do parts of implementations in a lower-dimension subspace rather than in the original high-dimension space, can offset this inescapable sketching cost. This expectation holds true when sketching is applied to algorithms with long-term recurrence, such as the GMRES-type methods considered in [4, 5, 13], where the sketched vectors (used once or twice) contribute to multiple matrix-matrix multiplications, least-squares problems, or projection tasks in each iteration. Additionally, we hope that the sketched orthogonalization property (as opposed to the

standard orthogonalization) can be "recyclable" for subsequent computations. However, these discussions are not always true for algorithms with short-term recurrence [34], like the conjugate gradient typed methods considered in this work. Based on these trade-offs and the total computational flops cost for each iteration, we conclude that applying dimension reduction sketching techniques is more advantageous for algorithms with long-term recurrence, rather than those with short-term recurrence, even if both types of randomized variants can converge.

TABLE 2.1
The difference between LOBPCG and sLOBPCG variants with normal Rayleigh-Ritz (RR) procedure in terms of the floating-point operations (or flops) and stored and updated vectors within a single iteration step.

LOBPCG + RR	floating-point operations ($\perp$ + `chol`/CGS/MGS)		stored vectors	
	$R \perp W_{jB}$:	$(4n-1)p^2$		
	do `chol`/CGS/MGS as :			
	`chol`$((X_j, W_{jB}))$:	$(4n-1)p^2$		
	`chol`$((R_J, Z_{jB_J}))$:	$(4n-1)p^2$		
	`chol`$((P_J, Q_{jB_J}))$:	$(4n-1)p^2$	Update :	X, P
	$R_J^T W_{jB} = 0$:	0		W_{jA}, Q_{jA}
	$X_j^T W_{jA} = \Lambda_j$:	0		W_{jB}, Q_{jB}
	$X_j^T W_{jB} = I$:	0		
	$R_J^T Z_{jB_J} = I$:	0		
	$P_J^T Q_{jB_J} = I$:	0		
	Whole flops:	$(16n-3)p^2 = \mathcal{O}(np^2)$		
sLOBPCG + RR	**floating-point operations ($\perp_\Theta$ + `chol`/RGS)**		**stored vectors**	
	${R^\Theta}$, ${Z_{jB}}^\Theta$:	$2np\log_2(t+1)$ [7]		
	$R \perp_\Theta W_{jB}$:	$(2n+2t-1)p^2$		
	do `chol`/RGS as :			
	`chol`$((X_j^\Theta, W_{jB}^\Theta))$:	$(2n+2t-1)p^2$		
	`chol`$((R_J^\Theta, Z_{jB_J}^\Theta))$:	$(2n+2t-1)p^2$		
	`chol`$((P_J^\Theta, Q_{jB_J}^\Theta))$:	$(2n+2t-1)p^2$	Update :	X, P
	$R_J^T W_{jB} \neq 0$:	$(2n-1)p^2$		W_{jA}, Q_{jA}
	$X_j^T W_{jA} \neq \Lambda_j$:	$(2n-1)p^2$		W_{jB}, Q_{jB}
	$X_j^T W_{jB} \neq I$:	$(2n-1)p^2$	Update :	$X^\Theta, {W_{jB}}^\Theta$
	$R_J^T Z_{jB_J} \neq I$:	$(2n-1)p^2$		$P^\Theta, {Q_{jB}}^\Theta$
	$P_J^T Q_{jB_J} \neq I$:	$(2n-1)p^2$		
	Whole flops:	$(18n+8t-9)p^2 + \mathcal{O}(np\log_2(t+1))$ $= \mathcal{O}(np^2 + np\log_2(t+1))$		

2.3.5. Convergence analysis. Currently, there is no established theory to accurately predict the convergence rate of the LOBPCG algorithm [17], which similarly makes predicting the convergence speed of the sLOBPCG algorithm equally challenging. Given the LOBPCG variants belongs to the category of gradient-based methods for the eigenvalue problem, thus, we can use the known and well-developed convergence theory of the latter methods to approximate convergence rate of the former methods. For example, see the results by Neymeyr [24–26] for the theoretical convergence rate of the block preconditioned inverse iterations for computing single and multiple eigenpairs. In short, we do not prove any new theoretical convergence rate results for the sLOBPCG algorithm, but as the remedy Knyazev did for LOBPCG [17, Section 7 and 8]: the numerical comparisons are used to illustrate that the sLOBPCG is as efficient as LOBPCG.

Assume the convergence threshold is denoted as a given parameter ϵ, the stopping criterion for the eigenpairs (Λ_j, X_j) (with $\Lambda_j = \text{Diag}(\lambda_j^{(1)}, \ldots, \lambda_j^{(p)}) \in \mathbb{R}^{p\times p}, X_j = [x_j^{(1)}, \ldots, x_j^{(p)}] \in \mathbb{R}^{n\times p}$) of the eigenvalue problem (1.1) considered in this work is based on the Eucledian norm of the columns of block residual

$$\eta(\lambda_j^{(i)}, x_j^{(i)}) = \|R_j(:,i)\| = \|Ax_j^{(i)} - x_j^{(i)}\lambda_j^{(i)}\| \leq \epsilon \text{ for all } i = 1, 2 \ldots, p \tag{2.29}$$

or its scaled version

$$\eta_{(\lambda,x)}(\lambda_j^{(i)}, x_j^{(i)}) = \frac{\|Ax_j^{(i)} - x_j^{(i)}\lambda_j^{(i)}\|}{\|x_j^{(i)}\lambda_j^{(i)}\|} \leq \epsilon. \tag{2.30}$$

The corresponding Estimated eigenvalue errors is formed as

$$\eta(\lambda_j^{(i)}) = |\Lambda_k(:,i) - \Lambda_j(:,i)| = |\lambda_k^{(i)} - \lambda_j^{(i)}| \text{ with } k = 1, 2, \ldots, j-1. \tag{2.31}$$

2.3.6. Further remarks on sLOBPCG with the sketched Rayleigh-Ritz procedure. Comparing to Section 2.3.3, in this section, we discuss the situation of applying the sketched Rayleigh-Ritz (sRR) projection defined in Definition 5 to Knyazev's LOBPCG method.

DEFINITION 5 (Sketched Rayleigh-Ritz projection). *Consider a trial subspace $\mathcal{W}$ of $\mathbb{R}^n$. Given two general nonsingular matrices $A, B \in \mathbb{R}^{n\times n}$, a sketching matrix $\Theta \in \mathbb{R}^{t\times n}(t \lll n)$, a scale $\lambda \in \mathbb{R}$, and a vector $y \in \mathcal{W}$, we see that (λ, y) is a Rayleigh-Ritz pair of the pencil $\Theta^T\Theta(A - \lambda B)$, a low-rank approximation of $A - \lambda B$, with respect to the space $\mathcal{W}$ if and only if*

$$Ay - \lambda By \perp_\Theta \mathcal{W}$$

or equivalently,

$$\forall w \in \text{Range}(\mathcal{W}) \quad (w^\Theta)^T (Ay - \lambda By)^\Theta = 0 \text{ or } w^T \Theta^T\Theta(Ay - \lambda By) = 0.$$

The vector y is a Rayleigh-Ritz vector associated with the Rayleigh-Ritz value λ.

Based on Definition 5 and the processes described in Section 2.3.1-2.3.3, the generalized eigenvalue problem of sLOBPCG with sRR and trial subspace $\mathcal{W}_j$ in Equation (2.1) can be written as

$$G_A{}^{sRR}Y = G_B{}^{sRR}Y\Lambda_{j+1}$$

with

$$G_A{}^{sRR} = \begin{bmatrix} \Lambda_j & (X_j{}^\Theta, Z_{jA}{}^\Theta) & (X_j{}^\Theta, Q_{jA}{}^\Theta) \\ (R_j{}^\Theta, W_{jA}{}^\Theta) & (R_j{}^\Theta, Z_{jA}{}^\Theta) & (R_j{}^\Theta, Q_{jA}{}^\Theta) \\ (P_j{}^\Theta, W_{jA}{}^\Theta) & (P_j{}^\Theta, Z_{jA}{}^\Theta) & (P_j{}^\Theta, Q_{jA}{}^\Theta) \end{bmatrix} \in \mathbb{R}^{p_Y\times p_Y},$$

$$G_B{}^{sRR} = \begin{bmatrix} I & 0 & (X_j{}^\Theta, Q_{jB}{}^\Theta) \\ 0 & I & (R_j{}^\Theta, Q_{jB}{}^\Theta) \\ (P_j{}^\Theta, W_{jB}{}^\Theta) & (P_j{}^\Theta, Z_{jB}{}^\Theta) & I \end{bmatrix} \in \mathbb{R}^{p_Y\times p_Y},$$

and $Y \in \mathbb{R}^{p_{Y_J}\times p}$ with the form shown in Equation (2.7). Then the new eigenvectors for next iteration are updated as $(X^\Theta)^{new} = (\Theta X)^{new} = \Theta[X_j, R_j, P_j]Y$ and $X^{new} = [X_j, R_j, P_j]Y$.

If one only consider the first p rows of Y partitioned in Equation (2.7), we have

$$(X_j^T\Theta^T\Theta A X_j)Y_X = Y_X\Lambda_j$$

for the sLOBPCG with sRR. Thus repeat above sRR process and update equation with information of the first p columns, it will be some eigenvalues of $\mathcal{W}_j^T\Theta^T\Theta A\mathcal{W}_j \in \mathbb{R}^{p_Y\times p_Y}$ to approximate some eigenvalues of $\Theta^T\Theta A \in \mathbb{R}^{n\times n}$ rather than the original matrix $A \in \mathbb{R}^{n\times n}$. Even $\Theta^T\Theta A$ could be viewed as a low-rank approximation of A, their similarity closely depends on the random sketching.

To distinguish the previously discussed sLOBPCG method with the normal Rayleigh-Ritz (RR) procedure, we use the abbreviation sLOBPCG+sRR to represent the sLOBPCG method that incorporates a sketched RR procedure, as discussed in this section. Numerical tests reveal that in the early stages of iteration, the convergence curves of the residual norms for sLOBPCG+sRR overlap with those of the deterministic LOBPCG and randomized sLOBPCG methods. However, once the accuracy reaches a certain threshold, the LOBPCG and sLOBPCG methods continue to improve with additional iterations. In contrast, the sLOBPCG+sRR method experiences stagnation. This stagnation prevents sLOBPCG+sRR from achieving higher accuracy and ultimately causes it to fail in approximating the eigenpairs of the original problem to the desired precision. Due to these observations, we exclude the results of sLOBPCG+sRR from the subsequent numerical sections. When referring to the use of sLOBPCG variants in the following discussions, it specifically indicates the sLOBPCG method employing the normal RR procedure.

3. Numerical experiments. In the following sections, we present various numerical aspects of the novel sLOBPCG algorithm. For simplicity, we exclusively consider the normal eigenvalue problem as exemplified in the numerical sections of [17], that is Equation (1.1) with matrix $B = I$. To facilitate comparisons, we also include results obtained using the deterministic LOBPCG method [17]. We evaluate the performance of these algorithms based on two key metrics: the number of iteration steps ($\#iter$) and the CPU time ($\#time(s)$) required for convergence. To illustrate the potential benefit of sLOBPCG when compared to LOBPCG, in some experiments we also plot the convergence history of the eigenpairs errors along the iteration steps. This includes the Eucledian norm of the columns of the block residual $\eta(\lambda_j^{(i)}, x_j^{(i)})$ described in Equation (2.29) or its scaled one $\eta_{(\lambda,x)}(\lambda_j^{(i)}, x_j^{(i)})$ in Equation (2.30), and the estimated eigenvalue errors $\eta(\lambda_j^{(i)})$ shown in Equation (2.31) with $j = 1, 2, \ldots, m_d$, and m_d is the final value of the consumed $\#iter$ of the involved block methods. Moreover, we may also plot out the convergence history of the approximated eigenvalues $[\lambda_j^{(1)}, \ldots, \lambda_j^{(p)}]$, and the condition number of the trail space $\mathcal{W}_j$ in Eucledian norm along the iteration step. For simplicity, we set Cond.-W $= \texttt{cond}(\mathcal{W}_j, 2)$ where $\texttt{cond}(\cdot)$ refers to a function to compute the condition number.

The default experimental settings are as follows: the block initial eigenvector $X_0 = \texttt{randn}(n, p) = [x^{(1)}, x^{(2)}, \ldots, x^{(p)}] \in \mathbb{R}^{n\times p}$ is constructed with p randomly generated linearly independent vectors, and p is the number of clustered eipenparis to be computed. The sketching matrix is an SRHT matrix $\Theta \in \mathbb{R}^{t\times p}$ with sketching dimension $t = 3 \times t_{\mathcal{W}}$, where $t_{\mathcal{W}} = 3 \times p$ refers to the maximal length of the trail subspace $\mathcal{W}_j$ with soft-locking strategy illustrated in Section 2.1.3. In this case, the sketching dimension t depends solely on the block size p and is independent of the size of the coefficient matrix n. Note that other setting of the sketching dimension works as well, such as $t = 2 \times t_{\mathcal{W}} \times \log(n)/\log(t_{\mathcal{W}})$ used in [4] for the randomized Gram-Schmidt (RGS) process. For this case, the sketching dimension t is related to both p and $\log(n)$. Unless otherwise noted, the number of clustered eigenpairs is set to $p = 10$, the maximum iterations $maxIters$ is 50000, and the convergence threshold is $\epsilon = 10^{-4}$. Generally, we stop the algorithm when the residual norm satisfying $\max\limits_{i=1,\ldots,p} \eta(\lambda_j^{(i)}, x_j^{(i)}) \leq \epsilon$ or when $\#iter$ reaches $maxIters$. We may also use scaled residual norm $\max\limits_{i=1,\ldots,p} \eta_{(\lambda,x)}(\lambda_j^{(i)}, x_j^{(i)}) \leq \epsilon$ as the stopping criterion for some cases.

Based on the B-orthogonalization strategies for the LOBPCG and sLOBPCG variants discussed at the end of Section 2.3.3, we observe that there are multiple approaches for achieving B-orthogonalization of the columns of X_j, R_j, P_j, as well as for implementing the B-orthogonalization of R_j with respect to X_j. Numerical results show that the performance of involved solvers employing these different strategies is similar to each other. Below, we summarize the default B-orthonormalization strategies used in this section:

- LOBPCG:
  - B-orthogonalization of X_j, R_j, P_j in Section 2.1.1: $\texttt{chol}(\cdot)$;
  - B-orthogonalize R_j to X_j Section 2.1.2: CGS$(\cdot)$;
- sLOBPCG variants with normal Rayleigh-Ritz procedure:
  - B-Θ-orthogonalization X_j, R_j, P_j in Section 2.3.1: $\texttt{chol}(\cdot)$;
  - B-Θ-orthogonalize R_j to X_j Section 2.3.2: RGS(20reorth)$(\cdot)$.

The experiments have been carried out on the CLEPS [2] system by MATLAB (R2021b) with hardware setting as 2x Cacade Lake Intel Xeon 5218 16 cores, 2.4GHz, 6 GB RAM. Numerical experiments are carried out on a set of symmetric positive definite (denoted as SPD) and symmetric but not positive definite (denoted as SYS) matrices downloaded from the University of Florida SuiteSparse Matrix Collection [9]. Following the test examples used in [17], we also consider a self-defined SYS matrix of size $n = 10000$; we denote it as Matrix$_1$. It is composed by the addition of two sparse SYS matrices generated by two MATLAB built-in functions. To be specific, $\texttt{Matrix}_1 = \texttt{spdiags}([1 : n]', 0, n, n) + \texttt{sprandsym}(n, .1)$, where $\texttt{spdiags}(\cdot)$ refers to a sparse matrix formed from diagonals, and the $\texttt{sprandsym}(\cdot)$ means a sparse random symmetric matrix. The main features of these SPD and SYS are respectively described in Table 3.1 and Table 3.2, where notation "-" refers to the information is unavailable from the SuiteSparse Matrix Collection website. As a supplement to testing examples from industry, we also consider a sequence of sparse SYS Hamiltonian matrices [12] derived from molecular simulations in quantum chemistry. Information of these large sparse SYS Hamiltonian matrices is presented in Table 3.8.

[2] https://paris-cluster-2019.gitlabpages.inria.fr/cleps/cleps-userguide/index.html

TABLE 3.1
Main characteristics of the symmetric positive definite (denoted as SPD here) matrices

Name	n	Nonzero	Nonzero / n^2	Origin*	Cond. number
af_shell3	504,855	17,562,051	6.8904e-05	Sub. Stru. Prob.	-
apache1	80,800	542,184	8.3047e-05	Stru. Prob.	-
bodyy4	17,546	121,550	3.9482e-04	Stru. Prob.	8.06e+02
bodyy5	18,589	128,853	3.7289e-04	Stru. Prob.	7.87e+03
bodyy6	19,366	134,208	3.5785e-04	Stru. Prob.	769e+04
boneS01	127,224	5,516,602	3.4083e-04	Model Redu. Prob.	-
bundle1	10,581	770,811	0.0069	Comp. Grap./Vis. Prob.	1.00e+03
cant	62,451	4,007,383	0.0010	2D/3D Prob.	-
cbuckle	13,681	676,515	0.0036	Stru. Prob.	3.29e+07
cfd1	70,656	1,825,580	3.6568e-04	CFD Prob.	-
cfd2	123,440	3,085,406	2.0249e-04	CFD Prob.	-
crystm02	13,965	322,905	0.0017	Mat. Prob.	2.504738e+02
crystm03	24,696	583,770	9.5717e-04	Mat. Prob.	2.640325e+02
Dubcova1	16,129	253,009	9.7257e-04	2D/3D Prob.	9.97e+02
Dubcova2	65,025	1,030,225	2.4365e-04	2D/3D Prob.	-
Dubcova3	146,689	3,636,643	1.6901e-04	2D/3D Prob.	-
finan512	74,752	596,992	1.0684e-04	Econ. Prob.	-
fv1	9,604	85,264	9.2440e-04	2D/3D Prob.	8.81e+00
fv2	9,801	87,025	9.0595e-04	2D/3D Prob.	8.81e+00
fv3	9,801	87,025	9.0595e-04	2D/3D Prob.	2.03e+03
G2_circuit	150,102	726,674	3.2253e-05	Cir. Sim. Prob.	-
Kuu	7,102	340,200	0.0067	Stru. Prob.	1.57e+04
minsurfo	40,806	203,622	1.2229e-04	Opti. Prob.	-
msdoor	415,863	19,173,163	1.1086e-04	Stru. Prob.	-
nd6k	18,000	6,897,316	0.0213	2D/3D Prob.	1.573206e+07
nd12k	36,000	14,220,946	0.0110	2D/3D Prob.	1.330782e+07
nd24k	72,000	28,715,634	0.0055	2D/3D Prob.	-
obstclae	40,000	197,608	1.2350e-04	Opti. Prob.	4.10e+01
Pres_Poisson	14,822	715,804	0.0033	CFD Prob.	2.04e+06
pdb1HYS	36,417	4,344,765	0.0033	Weig. Undire. Grag.	3.53e+11
qa8fm	66,127	1,660,579	3.7975e-04	Acos. Prob.	-
shallow_water1	81,920	327,680	4.8828e-05	CFD Prob.	-
shallow_water2	81,920	327,680	4.8828e-05	CFD Prob.	-
ted_B	10,605	144,579	0.0013	Ther. Prob.	1.89e+07
ted_B_unscaled	10,605	144,579	0.0013	Ther. Prob.	1.27e+11
thermal1	82,654	574,458	8.4087e-05	Ther. Prob.	-
torsion1	40,000	197,608	1.2350e-04	Dup. Opti. Prob.	4.10e+01
wathen100	30,401	471,601	5.1027e-04	Rand. 2D/3D Prob.	5.82e+03
wathen120	36,441	565,761	4.2604e-04	Rand. 2D/3D Prob.	2.58e+03

*Structural Problem, Computer Graphics/Vision Problem, Optimization Problem, Model Reduction Problem, Computational Fluid Dynamics Problem, 2D/3D Problem, Duplicate Optimization Problem, Acoustics Problem, Random 2D/3D Problem, Subsequent Structural Problem, Circuit Simulation Problem, Weighted Undirected Graph, Economic Problem, Materials Problem, and Thermal Problem are simplified as Stru. Prob., Comp. Grap./Vis. Prob., Opti. Prob., Model Redu. Prob., CFD Prob., 2D/3D Prob., Dup. Opti. Prob., Acos. Prob., Rand. 2D/3D Prob., Sub. Stru. Prob., Cir. Sim. Prob., Weig. Undire. Grag., Econ. Prob., Mat. Prob., and Ther. Prob., respectively.

3.1. Influence of the value of the sketching dimension. In this section, we investigate how the value of sketching dimension t in the SRHT matrix Θ affects the performance and robustness of the proposed sLOBPCG variants. Except for the default numerical setting described above (i.e., $3\times t_{\mathcal{W}}$ with $t_{\mathcal{W}} = 3\times p$, a commonly sketching dimension bound used in RandNLA), the testing sketching dimension t for sLOBPCG

TABLE 3.2
Main characteristics of the symmetric but not positive definite (denoted as SYS here) matrices

Name	n	Nonzero	Nonzero / n^2	Origin*	Cond. number
3dtube	45,330	3,213,618	0.0016	CFD Prob.	-
af_shell1	504,855	17,562,051	6.8904e-05	Stru. Prob. Seq.	-
barth5	15,606	61,484	2.5245e-04	Dup. Stru. Prob.	5.85e+10
brainpc2	27,607	179,395	2.3538e-04	Opti. Prob.	4.46e+04
bratu3d	27,792	173,796	2.2501e-04	2D/3D Prob.	5.92e+02
copter1	17,222	211,064	7.1162e-04	CFD Prob.	9.63e+03
c-55	32,780	403,450	3.7547e-04	Opti. Prob.	4.74e+08
darcy003	389,874	2,097,566	1.3800e-05	2D/3D Prob.	-
d_pretok	182,730	1,641,672	4.9166e-05	2D/3D Prob.	-
ecology1	1,000,000	4,996,000	4.9960e-06	2D/3D Prob.	-
F1	343,791	26,837,113	2.2706e-04	Stru. Prob.	-
fcondp2	201,822	11,294,316	2.7728e-04	Stru. Prob.	-
gearbox	153,746	9,080,404	3.8415e-04	Stru. Prob.	-
gupta1	31,802	2,164,210	0.0021	Opti. Prob.	Inf
helm2d03	392,257	2,741,935	1.7820e-05	2D/3D Prob.	-
helm3d01	32,226	428,444	4.1255e-04	2D/3D Prob.	2.45e+05
ins2	309,412	2,751,484	2.8740e-05	Opti. Prob.	-
k1_san	67,759	559,774	1.2192e-04	2D/3D Prob.	-
kkt_power	2,063,494	12,771,361	2.9994e-06	Opti. Prob.	-
Lin	256,000	1,766,400	2.6953e-05	Stru. Prob.	-
lp1	534,388	1,643,420	5.7549e-06	Opti. Prob.	-
Matrix_1	10,000	9,526,450	0.0953	self-defined	6.34e+04
mario001	38,434	204,912	1.3872e-04	2D/3D Prob.	1.87e+04
mario002	389,874	2,097,566	1.3800e-05	2D/3D Prob.	-
nemeth01	9,506	725,054	0.0080	T/QC Prob.	1.40e+02
nlpkkt200	16,240,000	440,225,632	1.6692e-06	Opti. Prob.	-
onera_dual	85,567	419,201	5.7255e-05	Stru. Prob.	-
pct20stif	52,329	2,698,463	9.8544e-04	Stru. Prob.	-
pkustk03	63,336	3,130,416	7.8037e-04	Stru. Prob.	-
qa8fk	66,127	1,660,579	3.7975e-04	Acou. Prob.	-
rajat06	10,922	46,983	3.9386e-04	Cirt Sim Prob.	6.35e+02
rajat07	14,842	63,913	2.9014e-04	Cirt Sim Prob.	7.89e+02
rajat09	24,482	105,573	1.7614e-04	Cirt Sim Prob.	1.23e+03
rajat10	30,202	130,303	1.4285e-04	Cirt Sim Prob.	1.31e+03
saylr4	3,564	22,316	0.0018	CFD Prob.	6.86e+06
struct3	53,570	1,173,694	4.0899e-04	Stru. Prob.	-
tandem_vtx	18,454	253,350	7.4394e-04	Stru. Prob.	1.68e+05
tuma1	22,967	87,760	1.6638e-04	2D/3D Prob.	3.07e+03
tuma2	12,992	49,365	2.9246e-04	2D/3D Prob.	1.70e+03
turon_m	189,924	1,690,876	4.6876e-05	2D/3D Prob.	-

*Abbreviation T/QC Prob., Cirt Sim Prob., Stru. Prob. Seq., Dup. Stru. Prob., refers to Theoretical/Quantum Chemistry Problem, Circuit Simulation Problem, Structural Problem Sequence, Duplicate Structural Problem, respectively.

is also set as $t = t_{\mathcal{W}}, l, 2\times l, 3\times l, 5\times l$ with $l = 2\times t_{\mathcal{W}} \times \log(n)/\log(t_{\mathcal{W}})$ (the sketching dimension bound defined in [4]). With these numerical settings, for illustration purpose, the numerical results for solving the eigenpairs of the self-defined Matrix_1 and an SPD matrix wathen120 by LOBPCG and sLOBPCG variants with different t are depicted in Table 3.3. Within Table 3.3, notation "$*$" indicates that the convergence based on the residual norm $\max\limits_{i=1,\ldots,p} \eta(\lambda_j^{(i)}, x_j^{(i)}) \leq \epsilon$ was not met.

For this example, no significant impact on performance is observed with an increase in the sketching

TABLE 3.3
Numerical results of LOBPCG variants with different sketching dimension t in terms of $\#iter$ and $\#time(s)$ for the self-defined symmetric Matrix$_1$ with the $p = 10$ eigenpairs to be computed. Here $l = 2 \times t_{\mathcal{W}} \times log(n)/log(t_{\mathcal{W}})$, $t_{\mathcal{W}} = 3 \times p$, $\epsilon = 10^{-4}$, and the "$*$" indicates that the algorithm diverges.

# t / t	Matrix (SPD / SYS)	Method	$\#iter$	$\#time(s)$
p : 10 / 10	wathen120 / Matrix$_1$	sLOBPCG$_{\text{e-s}}$	$*$ / $*$	$*$ / $*$
		sLOBPCG	$*$ / $*$	$*$ / $*$
$t_{\mathcal{W}}$: 30 / 30	wathen120 / Matrix$_1$	LOBPCG	339 / 1226	15.3545 / 45.4359
		sLOBPCG$_{\text{e-s}}$	**325** / 1226	36.5405 / 66.8635
		sLOBPCG	339 / **684**	22.3644 / 33.4502
$3 \times t_{\mathcal{W}}$: 90 / 90	wathen120 / Matrix$_1$	LOBPCG	339 / 1226	16.893 / 51.2093
		sLOBPCG$_{\text{e-s}}$	343 / **707**	38.6099 / 49.1428
		sLOBPCG	**333** / 814	22.629 / 38.6313
l : 186 / 163	wathen120 / Matrix$_1$	LOBPCG	339 / 1226	15.9744 / 47.001
		sLOBPCG$_{\text{e-s}}$	442 / 589	43.9464 / 44.7617
		sLOBPCG	**284** / **549**	20.749 / 35.1521
$2 \times l$: 372 / 326	wathen120 / Matrix$_1$	LOBPCG	339 / 1226	15.7819 / 45.7408
		sLOBPCG$_{\text{e-s}}$	331 / **615**	35.7045 / 45.0125
		sLOBPCG	**331** / 692	22.3746 / 34.7217
$3 \times l$: 558 / 489	wathen120 / Matrix$_1$	LOBPCG	339 / 1226	15.836 / 48.9274
		sLOBPCG$_{\text{e-s}}$	**294** / **920**	35.447 / 56.6996
		sLOBPCG	427 / 1079	25.4063 / 46.2687
$5 \times l$: 930 / 815	wathen120 / Matrix$_1$	LOBPCG	339 / 1226	15.4633 / 45.2971
		sLOBPCG$_{\text{e-s}}$	341 / **620**	37.4333 / 44.6228
		sLOBPCG	**335** / 709	23.4066 / 36.7186

dimension t, as long as it exceeds the required lower bound for the sketching dimension (i.e., $t \geq \min(t_{\mathcal{W}}, l)$). With a reasonable sketching dimension, it can be observed that the sketched LOBPCG variants, including sLOBPCG$_{\text{e-s}}$ and sLOBPCG, are capable of computing the required eigenpairs as efficient as the deterministic LOBPCG algorithm. Moreover, extensive numerical results demonstrate a high probability that the sLOBPCG variants (significantly) reduce the computational cost of $\#iter$, which in turn leads to a further reduction in the overall computational cost $\#time(s)$. It is also noticed that the performance of sLOBPCG$_{\text{e-s}}$ and sLOBPCG is similar to each other in terms of $\#iter$, while sLOBPCG$_{\text{e-s}}$ requires more $\#time(s)$. This is because each iteration of sLOBPCG$_{\text{e-s}}$ requires three sketchings, whereas sLOBPCG only requires one. Additionally, the $\#time(s)$ cost associated with performing sketching can be non-negligible.

As discussed in Section 2.2, how to reduce the cost of constructing a sketching matrix and applying it to a matrix or vector is an important but also challenging topic in RandNLA. In the remainder of this work, the sketching dimension is set to $t = 3 \times t_{\mathcal{W}}$ by default. This value depends solely on the block size p and is independent from the size of the coefficient matrix n.

3.2. Influence of the number of the block size. In this section, we illustrate the interplay between the number of eigenpairs to be computed and the performance of the LOBPCG variants; we vary $p = 1, 4, 6, 10, 20, 30, 50, 100$. The performances in terms of $\#iter$ and $\#time(s)$ are reported in Table 3.4 to illustrate the numerical results of solving the SPD matrix wathen120 and the self-defined SYS Matrix$_1$ with $t = 3 \times t_{\mathcal{W}}$ ($t_{\mathcal{W}} = 3 \times p$) and $\epsilon = 10^{-4}$. It can be observed that the $\#time(s)$ for all block solvers increase along with the growth of block size p. For explanation we retrospect that all the block solvers are required to solve a generalized eigenvalue problem $G_A Y = G_B Y \Lambda$ with Gram matrices $G_A, G_B \in \mathbb{R}^{p_Y \times p_Y}$. The $p_Y = 3 \times p$ if no partial convergence occurs and satisfying $p \leq p_Y \leq 3 \times p$ along iterations. Because of this, we summary that the cost of solving this generalized eigenvalue problem will gradually dominate the total $\#time(s)$ cost with the growth of block size p. When comparing the results of

solvers with a fixed p, the observations align closely with those in the previous section, that is the sLOBPCG variants demonstrate efficiency comparable to LOBPCG. And sLOBPCG$_{\text{e-s}}$ incurs a higher $\#time(s)$ due to the additional cost of applying sketching to two other block vectors.

A reasonable setting of the block size p should be based on the number of clustered eigenpairs, which requires priority information about the spectral distribution that is generally not known in advance. Therefore, in the remainder of this numerical section, we set the block size as $p = 10$ by default.

TABLE 3.4
Numerical results of LOBPCG variants with different block size p in terms of $\#iter$ and $\#time(s)$ for two matrices with different eigenpairs to be computed. Here $t = 3 \times t_{\mathcal{W}}$, $t_{\mathcal{W}} = 3 \times p$ and $\epsilon = 10^{-4}$.

# p / t	Matrix (SPD / SYS)	Method	$\#iter$	$\#time(s)$
1 / 9	wathen120 / Matrix$_1$	LOBPCG	591 / 702	6.1978 / 14.5054
		sLOBPCG$_{\text{e-s}}$	**558** / 703	8.2459 / 14.5169
		sLOBPCG	564 / **701**	5.2549 / 13.1285
4 / 36	wathen120 / Matrix$_1$	LOBPCG	287 / 882	7.0637 / 23.2731
		sLOBPCG$_{\text{e-s}}$	**252** / 909	14.2177 / 32.986
		sLOBPCG	253 / **805**	8.5729 / 24.7395
6 / 54	wathen120 / Matrix$_1$	LOBPCG	**324** / 1772	9.6936 / 51.5895
		sLOBPCG$_{\text{e-s}}$	354 / 984	23.3955 / 43.4237
		sLOBPCG	350 / **704**	13.5244 / 29.2783
10 / 90	wathen120 / Matrix$_1$	LOBPCG	**339** / 1226	15.6759 / 45.6136
		sLOBPCG$_{\text{e-s}}$	359 / 1075	40.6751 / 60.5165
		sLOBPCG	340 / **750**	23.4286 / 36.7012
20 / 180	wathen120 / Matrix$_1$	LOBPCG	369 / 495	28.6103 / 44.7397
		sLOBPCG$_{\text{e-s}}$	349 / **389**	67.3797 / 60.1008
		sLOBPCG	**302** / 600	36.3695 / 61.729
30 / 270	wathen120 / Matrix$_1$	LOBPCG	435 / 798	44.797 / 65.2185
		sLOBPCG$_{\text{e-s}}$	**426** / **787**	111.8627 / 110.1823
		sLOBPCG	482 / 860	64.1862 / 87.2924
50 / 450	wathen120 / Matrix$_1$	LOBPCG	808 / **510**	100.6285 / 73.899
		sLOBPCG$_{\text{e-s}}$	831 / 1033	271.5738 / 189.5917
		sLOBPCG	**608** / 1023	118.717 / 135.7128
100 / 900	wathen120 / Matrix$_1$	LOBPCG	640 / 477	187.4984 / 112.8754
		sLOBPCG$_{\text{e-s}}$	575 / 452	460.4572 / 214.1983
		sLOBPCG	**543** / **390**	266.9136 / 169.0814

3.3. Strategy for the orthonormalization of the columns of the block vector. Based on the orthonormalization strategies for LOBPCG and sLOBPCG variants outlined at end of Section 2.3.3, we summarize the various B-orthonormalization strategies to be compared in this section as follows.

- LOBPCG:
  - B-orthogonalization of X_j, R_j, P_j in Section 2.1.1: `chol`$(\cdot)$, MGS$(\cdot)$;
  - B-orthogonalize R_j to X_j Section 2.1.2: CGS$(\cdot)$, MGS$(\cdot)$;
- sLOBPCG variants with normal Rayleigh-Ritz procedure:
  - B-Θ-orthogonalization X_j, R_j, P_j in Section 2.3.1: `chol`$(\cdot)$, RGS$(\cdot)$;
  - B-Θ-orthogonalize R_j to X_j Section 2.3.2: RGS$(\cdot)$.

Table 3.5 presents the results of using these different orthonormalization strategies for solving 10 eigenpairs of an SPD and and an SYS examples selected from Table 3.1 and Table 3.2, respectively. From Table 3.5, we noticed that re-orthogonalization processes increase $\#time(s)$, but there is no significant impact on $\#iter$, particularly when the testing matrix is well-conditioned, such as the `wathen120` matrix.

TABLE 3.5

Numerical results of applying different strategies for implementing the B-orthogonalization of the block residual R to the block eigenvector X (i.e., $R \perp X$) in terms of $\#iter$ and $\#time(s)$ for an SPD and an SYS matrix with the $p = 10$ eigenpairs to be computed. Here $t = 3 \times t_{\mathcal{W}}$ and $\epsilon = 10^{-4}$.

# $R \perp X$	# B-orthonormal X, R, P	Matrix	Method	$\#iter$	$\#time(s)$
CGS	chol(·)	wathen120	LOBPCG	339	16.0072
MGS(2reorth)	chol(·)	wathen120	LOBPCG	339	21.6917
MGS(5reorth)	chol(·)	wathen120	LOBPCG	339	23.8801
CGS	chol(·)	c-55	LOBPCG	2403	87.2743
MGS(2reorth)	chol(·)	c-55	LOBPCG	2141	100.0005
MGS(5reorth)	chol(·)	c-55	LOBPCG	**1417**	95.78591
CGS	chol(·)	wathen120	LOBPCG	339	17.2895
CGS	MGS(2reorth)	wathen120	LOBPCG	339	22.6262
CGS	MGS(5reorth)	wathen120	LOBPCG	339	21.9575
CGS	chol(·)	c-55	LOBPCG	2403	87.779
CGS	MGS(2reorth)	c-55	LOBPCG	2014	104.7119
CGS	MGS(5reorth)	c-55	LOBPCG	**1878**	120.9614
RGS(5reorth)	chol(·)	wathen120	sLOBPCG$_{\text{e-s}}$	342	37.1184
			sLOBPCG	**309**	20.8371
RGS(10reorth)	chol(·)	wathen120	sLOBPCG$_{\text{e-s}}$	**306**	35.8964
			sLOBPCG	341	21.6057
RGS(20reorth)	chol(·)	wathen120	sLOBPCG$_{\text{e-s}}$	**328**	36.4628
			sLOBPCG	339	22.4108
RGS(5reorth)	chol(·)	c-55	sLOBPCG$_{\text{e-s}}$	1463	153.2691
			sLOBPCG	1619	104.6198
RGS(10reorth)	chol(·)	c-55	sLOBPCG$_{\text{e-s}}$	1441	148.5389
			sLOBPCG	1788	120.8447
RGS(20reorth)	chol(·)	c-55	sLOBPCG$_{\text{e-s}}$	**1293**	149.5971
			sLOBPCG	1408	97.4648
RGS(20reorth)	RGS(5reorth)	wathen120	sLOBPCG$_{\text{e-s}}$	442	44.4341
			sLOBPCG	**311**	22.054
RGS(20reorth)	RGS(10reorth)	wathen120	sLOBPCG$_{\text{e-s}}$	429	44.3073
			sLOBPCG	**338**	24.2958
RGS(20reorth)	RGS(20reorth)	wathen120	sLOBPCG$_{\text{e-s}}$	**301**	36.7359
			sLOBPCG	327	22.675
RGS(20reorth)	RGS(5reorth)	c-55	sLOBPCG$_{\text{e-s}}$	3789	285.6189
			sLOBPCG	3182	134.4775
RGS(20reorth)	RGS(10reorth)	c-55	sLOBPCG$_{\text{e-s}}$	1987	182.9228
			sLOBPCG	2808	124.9048
RGS(20reorth)	RGS(20reorth)	c-55	sLOBPCG$_{\text{e-s}}$	1647	164.2125
			sLOBPCG	**1560**	101.1567

3.4. Influence of the preconditioner. Based on the preconditioning techniques for eigenvalue problems discussed in [17, Section 2], the preconditioner for solving linear systems is considered in this section to accelerate the convergence rate of the involved block methods. To be specific, the preconditioner denoted as M is an approximation of A^{-1}. This corresponds to the *shift-and-invert* preconditioner $(A - \sigma B)^{-1}$ [32, Section 8.1] with zero shift $\sigma = 0$, which is usually a reasonable choice if one is interested in computing the smallest eigenvalues of a symmetric eigenvalue problem [20, Section 4.5.1] by LOBPCG [19]. For the SPD matrix, we consider an incomplete Cholesky factorization (denoted as `ichol`$(\cdot)$) as preconditioner for the preconditioned LOBPCG and sLOBPCG variants. For the SYS matrix, we utilize two different preconditioning strategies. The first involves the `Jacobi` preconditioner, which is constructed by the diagonal elements of the matrices. Alternatively, we also apply the incomplete LU factorization, denoted as `ilu`$(\cdot)$, as a preconditioner.

Numerical results for both block solvers, with and without preconditioning, for solving SPD and SYS matrices are presented in Table 3.6 and Table 3.7, respectively. Within these two tables, the superscript $^{*(i/p)}$ indicates that i eigenpairs ($0 < i \leq p$) out of the whole p have not met the stopping criteria within the maximum number of iteration steps. The results indicate that a well-defined preconditioner can enhance performance by reducing $\#iter$ and $\#time(s)$. However, we also noticed that an improper preconditioner leads to the inverse situation, as exemplified by the results for `nd24k` (`ichol`$(\cdot)$) and `ecology1` (`Jacobi`). Given that the definition of an appropriate preconditioner for an unknown eigenvalue system still remains open [16, 32], for the remainder of this section, we assume that no preconditioner is applied if the involved solvers can solve the problem effectively without preconditioning.

TABLE 3.6
Apply preconditioning for the involved block solvers for solving some SPD matrices listed in Table 3.1 with $p = 10$ eigenpairs to be computed, $t = 3 \times t_{\mathcal{W}} = 9 \times p = 90$ and $\epsilon = 10^{-4}$. The superscript $^{*(i/p)}$ indicates that i eigenpairs ($0 < i \leq p$) out of the whole p have not met the stopping criteria within the maximum number of iteration steps.

Matrix (Prec.)	$\#iter$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG	$\#time(s)$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG
af_shell3 (no Prec.)	$50000^{*(p/p)}$ / **6082** / 9053	57933.9212 / 6799.1055 / 7114.8465
cbuckle (no Prec.)	$50000^{*(p/p)}$ / $50000^{*(p/p)}$ / $50000^{*(p/p)}$	1270.5322 / 2802.4505 / 1769.5211
G2_circuit (no Prec.)	$50000^{*(1/p)}$ / 33129 / **31790**	4979.3799 / 7583.8969 / 4010.305
nd24k (no Prec.)	**6137** / 7201 / 8353	1349.0702 / 2287.045 / 1739.8822
af_shell3 (`ichol`$(\cdot)$)	12979 / **5962** / 15002	6564.0368 / 6181.2708 / 8612.092
cbuckle (`ichol`$(\cdot)$)	3517 / **2518** / 3134	62.8518 / 106.8719 / 73.6396
G2_circuit (`ichol`$(\cdot)$)	8595 / 7838 / **7288**	1841.5461 / 3387.5602 / 1825.1235
nd24k (`ichol`$(\cdot)$)	10903 / 7910 / **6383**	1780.6359 / 2307.3737 / 1446.8403

TABLE 3.7
Apply preconditioning for the involved block solvers for solving some SYS matrices listed in Table 3.2 with $p = 10$ eigenpairs to be computed, $t = 3 \times t_{\mathcal{W}} = 9 \times p = 90$ and $\epsilon = 10^{-4}$.

Matrix (Prec.)	$\#iter$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG	$\#time(s)$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG
Matrix$_1$ (no Prec.)	1226 / 1237 / **1079**	44.5212 / 66.9526 / 45.2689
ecology1 (no Prec.)	2043 / 1919 / **1907**	3467.608 / 5365.1941 / 3763.6458
Matrix$_1$ (`Jacobi`)	23 / 23 / 23	5.4774 / 2.2788 / 2.5094
ecology1 (`ilu`$(\cdot)$)	909 / 987 / **797**	1734.3426 / 2383.1438 / 1632.5837
ecology1 (`Jacobi`)	**4145** / 14219 / 6470	7487.4844 / 29836.7152 / 8387.2694

3.5. Experiments on a massive large sparce SYS Hamiltonian matrices. In this section, we solve the eigenvalue problem that arises in molecular simulations. Specifically, we provide a sequence of sparse matrices, known as Hamiltonian matrices [12], which are symmetric and are derived from processes describing the electronic properties of molecular systems encountered in quantum chemistry. We use the LOBPCG and its sketched variants to solve the clustered eigenvalues of the Hamiltonian matrices. Information about these large sparse symmetric Hamiltonian matrices is listed in Table 3.8. Numerical

results are reported in Table 3.9, which follow similar observations concluded previously.

TABLE 3.8
Main characteristics of the SYS Hamiltonian matrices from quantum chemistry

Name_Matrix	n	Nonzero	Nonzero/n^2
Hamiltonian_matrix_h2o	16,384	860,159	0.0032
Hamiltonian_matrix_h2oo2	16,384	761,855	0.0028
Hamiltonian_matrix_h2oQPKG	16,384	860,159	0.0032
Hamiltonian_matrix_h6	4096	342,015	0.0204
Hamiltonian_matrix_h10	1,048,576	744,226,815	6.7687e-04

TABLE 3.9
Numerical results of LOBPCG variants large sparce SYS Hamiltonian matrices listed in Table 3.8 with $p = 10$ eigenpairs to be computed, $t = 3 \times t_{\mathcal{W}} = 9 \times p = 90$, and $\epsilon = 10^{-4}$.

Matrix	$\#iter$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG	$\#time(s)$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG
h2o	868 / 634 / **619**	12.3161 / 18.5988 / 12.2061
h2oo2	393 / 367 / **349**	7.6044 / 14.1727 / 8.9917
h2oQPKG	825 / **628** / 668	11.5608 / 19.5286 / 12.5028
h6	136 / **92** / 135	3.6372 / 1.5948 / 1.5195
h10	**96** / 108 / 97	1138.2359 / 1356.1257 / 1199.8351

3.6. Experiments on a massive large sparse SPD and SYS matrices. In this section, a more exhaustive set of numerical results for solving the massive SPD matrices listed in Table 3.1 and the SYS matrices in Table 3.2 are reported in Table 3.10 and Table 3.11, respectively. The superscript $^{*(i/p)}$ indicates that there are i eigenpairs ($0 < i \leq p$) out of the whole p are not converged within the maximal iteration $maxIters$. From the results of `af_shell3` and `G2_circuit` showing in Table 3.10, we noticed that the LOBPCG method fails to solve the problem, but the randomized variants of sLOBPCG can. This exhibits the diverse advantages of using randomization for the deterministic algorithm. For the additional results of the SPD examples and SYS cases presented in Table 3.10 and Table 3.11, the observations align closely with those noted in the previous sections. The convergence histories for some of the listed matrices are depicted in Figure 3.1 and Figure 3.2. We show the convergence history of the errors of approximated eigenpairs (including the approximations of eigenvector and eigenvalue). From this, we observed that the convergence rate for eigenvalues is faster than that for eigenvectors. Additionally, the final achieved accuracy for eigenvalues is higher than that for eigenvectors.

Refer to Appendix C for the corresponding numerical results with $t = 2 \times t_{\mathcal{W}} \times \log(n)/\log(t_{\mathcal{W}})$, which is the sketching dimension used in [4] for the randomized Gram-Schmidt process.

4. Concluding remarks. The LOBPCG algorithm stands as one of the state-of-the-art eigensolver. In this work, we successfully introduced the randomized variant of the LOBPCG algorithm with dimension reduction techniques, denoted as sLOBPCG, for solving a group of eigenpairs of generalized eigenvalue problems with symmetric coefficient matrices. Numerical results demonstrate that the sLOBPCG variant exhibits comparable efficiency to the original LOBPCG method and, in certain instances, surpasses its performance. Based on the trade-off between the cost of sketching (includes constructing and applying sketching to vectors) and its benefits from implementing parts of operations in a reduced dimension, we summary that it is more attractive to apply the sketching to algorithms with long-term recurrence rather than the ones with short-term recurrence, even if their randomized variants can converge. In the future, the block SRHT technique proposed in [3] should be considered to further reduce the sketching cost in this work.

TABLE 3.10

Numerical results of LOBPCG variants with `chol` *for the B-/B-Θ-orthonormalization in terms of both $\#iter$ and $\#time(s)$ for all SPD matrices listed in Table 3.1 with the $p = 10$ eigenpairs to be computed, $t = 3\times t_{\mathcal{W}} = 9\times p = 90$ and $\epsilon = 10^{-4}$. The stopping criterion is the residual norm satisfying $\max_{i=1,\ldots,p} \eta(\lambda_j^i, x_j^{(i)}) \leq \epsilon$.*

Matrix	$\#iter$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG	$\#time(s)$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG
af_shell3	$50000^{*(p/p)}$ / **6082** / 9053	57933.9212 / 6799.1055 / 7114.8465
apache1	**8377** / 13776 / 15494	725.4148 / 2144.3237 / 1125.8208
bodyy4	3441 / 4091 / **2854**	59.2036 / 198.0735 / 87.1804
bodyy5	10461 / **8798** / 8879	277.8127 / 612.168 / 365.6974
bodyy6	**33744** / 36912 / 38371	1052.9205 / 2497.139 / 1513.675
boneS01	1807 / 2194 / **1577**	303.1483 / 542.8065 / 318.026
cant	14364 / 17469 / **9605**	1436.0213 / 2782.1994 / 1310.5358
cfd1	838 / 875 / **807**	73.7969 / 174.7589 / 99.9543
cfd2	1816 / 1945 / **1759**	454.1249 / 775.8414 / 517.9351
crystm02	1 / 1 / 1	0.24031 / 0.21692 / 0.26383
crystm03	1 / 1 / 1	0.21114 / 0.15974 / 0.61214
cbuckle	$50000^{*(p/p)}$ / $50000^{*(p/p)}$ / $50000^{*(p/p)}$	1270.5322 / 2802.4505 / 1769.5211
Dubcova1	184 / 189 / 184	7.3572 / 11.042 / 7.6365
Dubcova2	289 / 459 / **275**	30.765 / 67.5034 / 35.1432
Dubcova3	258 / 292 / **227**	53.7701 / 119.4949 / 67.1424
finan512	254 / 252 / **252**	29.6269 / 66.274 / 40.9166
fv1	265 / **177** / 215	7.6579 / 10.4277 / 8.1682
fv2	**140** / 179 / 171	6.0443 / 10.9361 / 7.3051
fv3	**140** / 179 / 172	5.432 / 9.9826 / 6.8936
G2_circuit	$50000^{*(1/p)}$ / 33129 / **31790**	4979.3799 / 7583.8969 / 4010.305
Kuu	548 / 530 / **472**	21.0454 / 25.7052 / 19.8174
minsurfo	1013 / 729 / **586**	75.2789 / 120.6771 / 72.2686
nd6k	2712 / 2879 / **2447**	231.7835 / 372.4913 / 249.8763
nd12k	4550 / **3503** / 3540	561.8281 / 724.5914 / 519.6452
nd24k	**6137** / 7201 / 8353	1349.0702 / 2287.045 / 1739.8822
obstclae	556 / **444** / 446	37.4566 / 61.6425 / 41.746
Pres_Poisson	**2362** / 2686 / 2363	134.0578 / 197.2994 / 145.6456
pdb1HYS	**1869** / 6183 / 3116	188.596 / 765.8501 / 315.1882
qa8fm	3 / 3 / 3	0.44156 / 0.7278 / 0.60673
ted_B	10 / 10 / 10	0.63915 / 0.85769 / 0.72063
ted_B_unscaled	2 / 2 / 2	0.14342 / 0.1491 / 0.16716
thermal1	658 / **518** / 585	83.4554 / 150.7551 / 104.6225
torsion1	556 / 483 / **444**	43.7442 / 71.6977 / 48.3775
wathen100	779 / 709 / **556**	42.752 / 65.6416 / 37.9989
wathen120	339 / 427 / **305**	28.4786 / 61.8549 / 33.9339
bundle1	16309 / **14296** / 16821	401.0694 / 595.0596 / 418.2936
shallow_water1	713 / **664** / 731	50.2168 / 101.6753 / 56.5922
shallow_water2	997 / **913** / 1185	122.8057 / 171.1286 / 117.3148

*Note that the scaled residual norm $\max_{i=1,\ldots,p} \eta_{(\lambda,x)}(\lambda_j^{(i)}, x_j^{(i)}) \leq \epsilon$ is considered as stopping criterion for matrices `bundle1`, `shallow_water1` and `shallow_water2`.

TABLE 3.11

Numerical results of LOBPCG variants with `chol` *for the B-/B-Θ-orthonormalization in terms of both $\#iter$ and $\#time(s)$ for all SYS matrices listed in Table 3.2 with $p = 10$ eigenpairs to be computed, $t = 3 \times t_{\mathcal{W}} = 9 \times p = 90$, and $\epsilon = 10^{-4}$. The stopping criterion is the residual norm satisfying $\max_{i=1,\ldots,p} \eta(\lambda_j^{(i)}, x_j^{(i)}) \leq \epsilon$.*

Matrix	$\#iter$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG	$\#time(s)$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG
af_shell1	50000$^{*(1/p)}$ / **45669** / 49890	46236.8526 / 58860.5633 / 45243.2796
3dtube	**521** / 1715 / 528	24.0123 / 111.8016 / 26.0017
barth5	**263** / 296 / 299	4.6726 / 10.9263 / 7.021
bratu3d	312 / 327 / **248**	6.5796 / 16.2949 / 7.8658
copter1	**309** / 358 / 358	5.1692 / 16.5143 / 8.8232
c-55	2403 / **1330** / 1523	83.6103 / 157.9857 / 101.2861
darcy003	574 / 573 / **572**	224.7753 / 410.7608 / 254.6734
d_pretok	16536 / 16137 / **16041**	3492.2297 / 6793.8573 / 4194.3284
ecology1	2043 / 1919 / **1907**	3467.608 / 5365.1941 / 3763.6458
fcondp2	287 / **224** / 294	67.205 / 104.6558 / 83.3067
gearbox	**137** / 140 / 178	36.8404 / 70.1701 / 51.4123
gupta1	68 / 68 / 68	2.4068 / 4.5253 / 2.9053
helm3d01	84 / 84 / 84	3.0172 / 5.7852 / 3.7592
helm2d03	**627** / 639 / 711	304.2885 / 535.891 / 383.51
ins2	21 / 22 / 21	7.1605 / 14.6576 / 8.9926
k1_san	3079 / 929 / **759**	134.2448 / 147.3192 / 72.9402
Lin	938 / 734 / **524**	191.2753 / 301.9077 / 159.9025
lp1	47 / 47 / 47	32.9782 / 69.6037 / 43.0342
Matrix$_1$	1226 / 1237 / **1079**	44.5212 / 66.9526 / 45.2689
Matrix$_2$	885 / 2631 / **807**	437.6357 / 912.9141 / 450.2785
mario001	**255** / 281 / 281	10.707 / 28.9492 / 17.4309
mario002	**574** / 622 / 576	214.423 / 412.6486 / 256.2713
nemeth01	1175 / 2057 / **821**	18.4736 / 56.3902 / 20.8356
nlpkkt200	2849 / 6883 / **2315**	74011.3036 / 189761.5373 / 76346.272
onera_dual	**203** / 216 / 217	19.8809 / 42.4877 / 26.7851
pct20stif	438 / 312 / **312**	16.8809 / 28.1619 / 16.332
pkustk03	406 / **368** / 394	18.3052 / 32.8197 / 20.0831
qa8fk	**95** / 107 / 264	8.0801 / 21.2285 / 16.8266
rajat06	376 / 330 / **296**	4.006 / 8.5767 / 4.347
rajat07	488 / 488 / **413**	5.9975 / 12.7365 / 6.4456
rajat09	**391** / 530 / 449	7.9094 / 23.5132 / 11.9886
rajat10	818 / **556** / 695	16.7092 / 27.2946 / 18.4342
saylr4	712 / 728 / 712	3.9722 / 6.7964 / 4.6907
struct3	**110** / 141 / 121	7.0571 / 16.1062 / 9.5507
tandem_vtx	319 / **298** / 308	5.0731 / 13.9556 / 7.1813
tuma1	260 / 236 / **219**	6.5727 / 15.4335 / 8.9799
tuma2	237 / 228 / **199**	3.6879 / 8.2879 / 5.0575
turon_m	**737** / 1524 / 860	184.5999 / 633.8368 / 223.5552
brainpc2	799 / 802 / 799	39.282 / 68.9912 / 44.6828
F1	45368 / **25909** / 50000$^{*(2/p)}$	17435.1166 / 23871.0552 / 22174.8263
kkt_power	4216 / 4924 / **4034**	20354.693 / 30367.5635 / 21193.3167

*Note that the scaled residual norm $\max_{i=1,\ldots,p} \eta_{(\lambda,x)}(\lambda_j^{(i)}, x_j^{(i)}) \leq \epsilon$ is considered as stopping criterion for matrices `brainpc2`, `F1` and `kkt_power`.

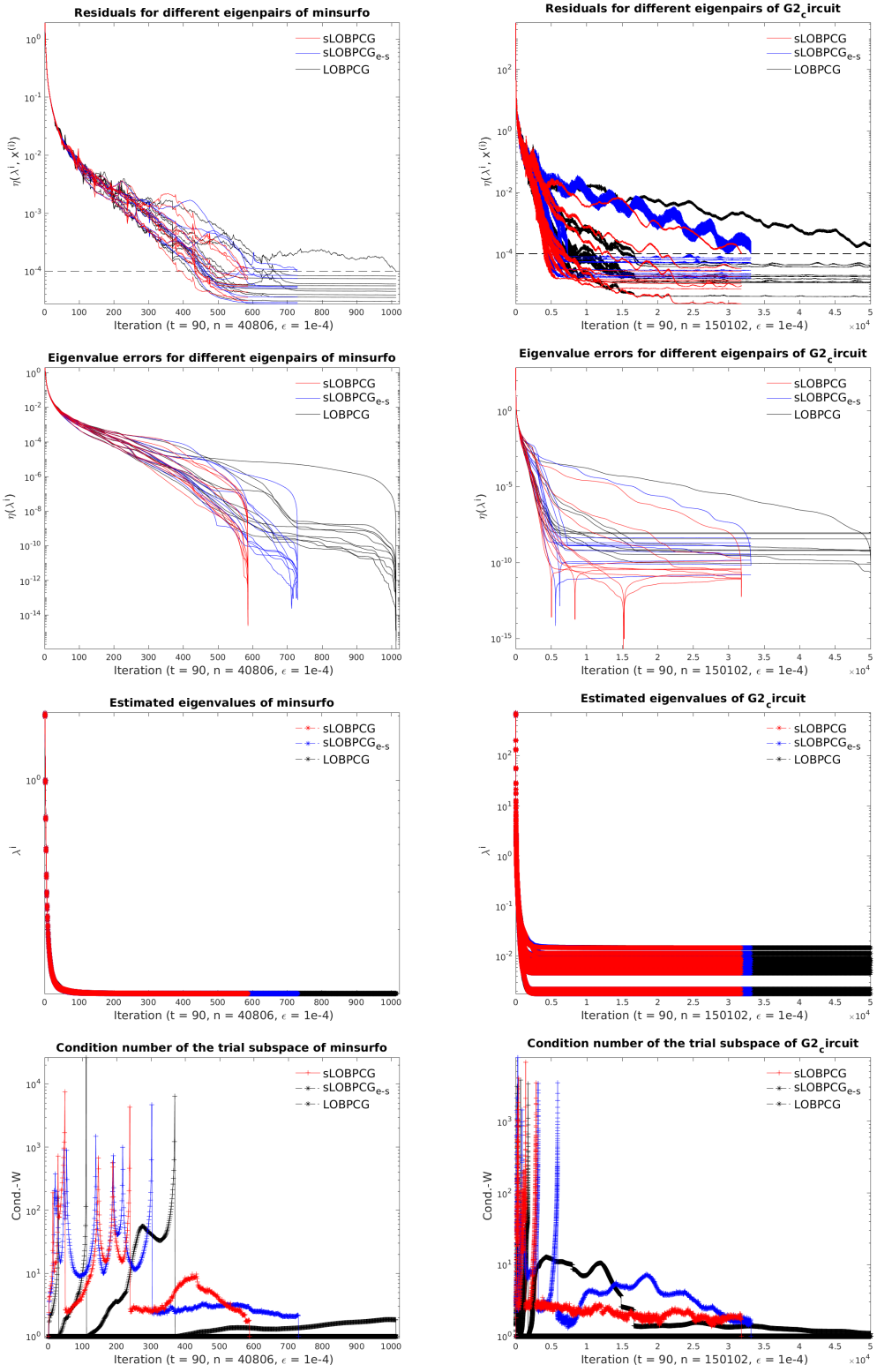

FIG. 3.1. *History of two SPD matrices listed in Table 3.1 with* $p = 10$ *eigenpairs to be computed,* $t = 3 \times t_{\mathcal{W}} = 90$, *and* $\epsilon = 10^{-4}$. *The* `chol` *is used for realizing the* B*-/*B*-*Θ*-orthonormalization.*

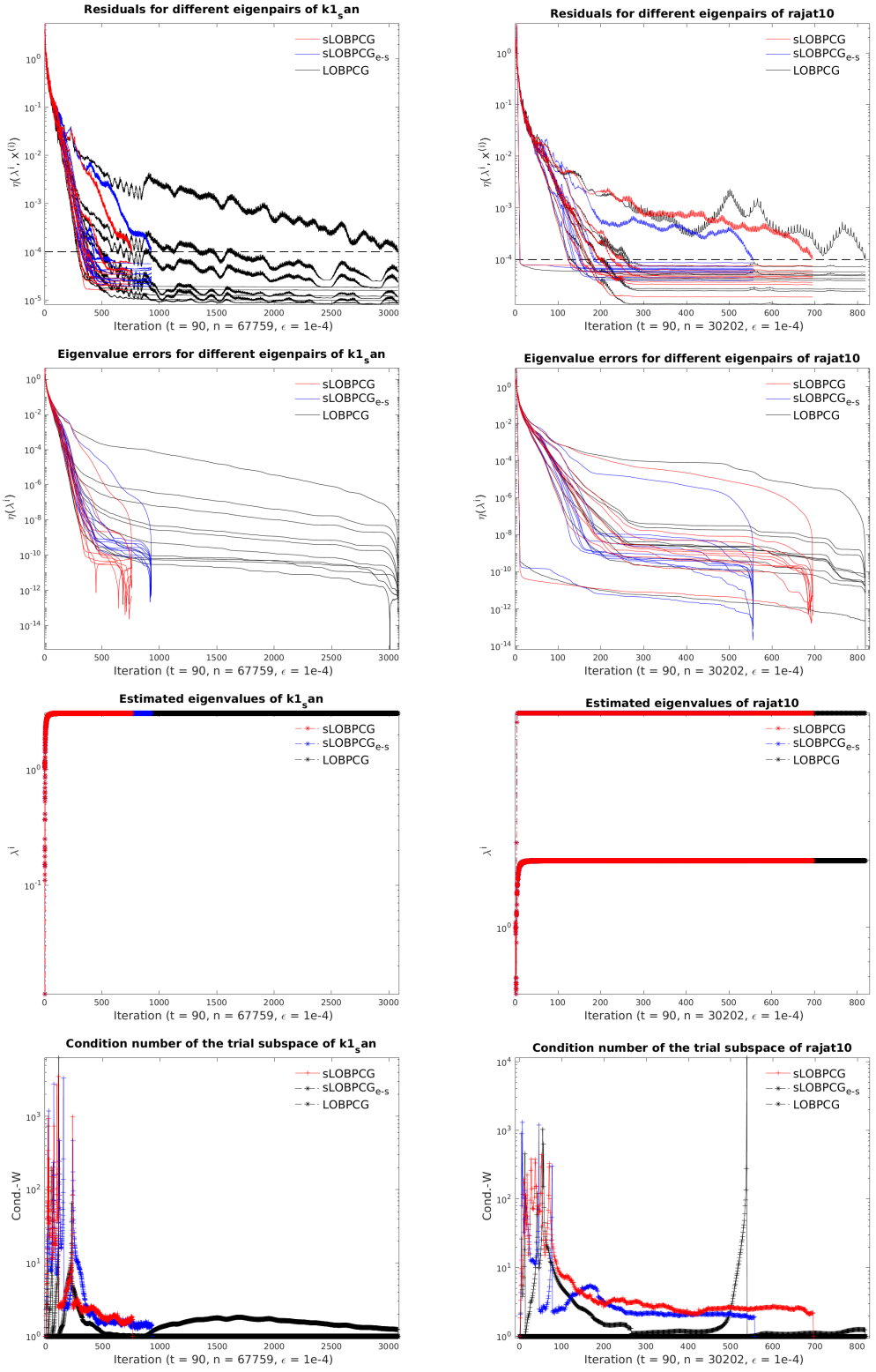

FIG. 3.2. *History of two SYS matrices listed in Table 3.2 with $p = 10$ eigenpairs to be computed, $t = 3 \times t_{\mathcal{W}} = 90$, and $\epsilon = 10^{-4}$. The* `chol` *is used for realizing the B-/B-Θ-orthonormalization.*

Acknowledgments. Part of this work was completed during the author's postdoctoral research with the Alpines team at Centre Inria de Paris, Laboratoire Jacques-Louis Lions (LJLL), France, under the supervision of Laura Grigori (EPFL and PSI, Switzerland). The author would like to thank her former colleague Siwar Badreddine (Qubit Pharmaceuticals, France) for providing the dataset of Hamiltonian examples. She also extends her gratitude to Agnieszka Miedlar (Virginia Tech, Blacksburg, USA) for the insightful discussions on LOBPCG during a workshop in Roscoff, France.

Experiments presented in this work were carried out using the CLEPS experimental testbed, supported by Centre Inria de Paris (refer to https://paris-cluster-2019.gitlabpages.inria.fr/cleps/cleps-userguide/index.html).

REFERENCES

[1] N. Ailon and B. Chazelle. The Fast Johnson–Lindenstrauss Transform and Approximate Nearest Neighbors. *SIAM J. Sci. Comput.*, 39(1):302–322, 2009. https://doi.org/10.1137/060673096.

[2] N. Ailon and E. Liberty. Fast Dimension Reduction Using Rademacher Series on Dual BCH Codes. *Discrete Comput Geom*, 42:615–630, 2009. https://doi.org/10.1007/s00454-008-9110-x.

[3] O. Balabanov, M. Beaupère, L. Grigori, and V. Lederer. Block subsampled randomized hadamard transform for low-rank approximation on distributed architectures. *arXiv*, 2022. https://doi.org/10.48550/arXiv.2210.11295.

[4] O. Balabanov and L. Grigori. Randomized Gram-Schmidt Process with Application to GMRES. *SIAM J. Sci. Comput.*, 44(3):A1450–A1474, 2022. https://doi.org/10.1137/20M138870X.

[5] O. Balabanov and L. Grigori. Randomized block Gram-Schmidt process for the solution of linear systems and eigenvalue problems. *arXiv*, 2023. https://doi.org/10.48550/arXiv.2111.14641.

[6] M. Beaupère. Parallel algorithms for computing low-rank decompositions of matrices and tensors. Ph.D thesis, Sorbonnes Université, Inria, 2023. https://www.theses.fr/2023SORUS108.

[7] C. Boutsidis and A. Gittens. Improved Matrix Algorithms via the Subsampled Randomized Hadamard Transform. *SIAM J. Matrix Anal. Appl.*, 34(3):1301–1340, 2013. https://doi.org/10.1137/120874540.

[8] S. Dasgupta and A. Gupta. An Elementary Proof of a Theorem of Johnsonand Lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, 2003. https://doi.org/10.1002/rsa.10073.

[9] T. A. Davis and Y. Hu. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Softw.*, 38:1:1–1:25, 2011.

[10] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Rev.*, 53(2):217–288, 2011. https://doi.org/10.1137/090771806.

[11] U. Hetmaniuk and R. Lehoucq. Basis selection in LOBPCG. *J. Comp. Phys.*, 218(1):324–332, 2006. https://doi.org/10.1016/j.jcp.2006.02.007.

[12] S. Holtz, T. Rohwedder, and R. Schneider. The Alternating Linear Scheme for Tensor Optimization in the Tensor Train Format. *SIAM J. Sci. Comput.*, 34(2):A683–A713, 2012. https://doi.org/10.1137/100818893.

[13] Y. Jang, L. Grigori, E. Martin, and C. Content. Randomized flexible GMRES with Singular Vectors Based Deflated Restarting. *J. Comp. Phys.*, 2022.

[14] W. B. Johnson, J. Lindenstrauss, and G. Schechtman. Extensions of lipschitz maps into Banach spaces. *Israel J. Math.*, 54:129–138, 1986. https://doi.org/10.1007/BF02764938.

[15] A. V. Knyazev. Convergence rate estimates for iterative methods for symmetric eigenvalue problems and its implementation in a subspace. *Internat. Ser. Numer. Math.*, 96:143–154, 1991.

[16] A. V. Knyazev. Preconditioned Eigensolvers - An Oxymoron? *Electron. Trans. Numer. Anal.*, 7:104–123, 1998.

[17] A. V. Knyazev. Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method. *SIAM J. Sci. Comput.*, 23(2):517–541, 2001. https://doi.org/10.1137/S1064827500366124.

[18] 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(5):2224–2239, 2007. https://doi.org/10.1137/060661624.

[19] A. V. Knyazev and K. Neymeyr. Efficient solution of symmetric eigenvalue problems using multigrid preconditioners in the locally optimal block conjugate gradient method. *Electron. Trans. Numer. Anal.*, 15:38–55, 2003.

[20] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* SIAM, Philadelphia, 1987.

[21] P.-G. Martinsson and J. A. Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, 2020. https://doi.org/10.1017/S0962492920000021.

[22] R. B. Morgan. GMRES with deflated restarting. *SIAM J. Sci. Comput.*, 24(1):20–37, 2002.

[23] Y. Nakatsukasa and J. A. Tropp. Fast & Accurate Randomized Algorithms for Linear Systems and Eigenvalue Problems. *arXiv*, 2021. https://doi.org/10.48550/arXiv.2111.00113.

[24] K. Neymeyr. A geometric theory for preconditioned inverse iteration I: Extrema of the Rayleigh quotient. *Linear Algebra Appl.*, 322(1):61–85, 2001. https://doi.org/10.1016/S0024-3795(00)00239-1.

[25] K. Neymeyr. A geometric theory for preconditioned inverse iteration II: Convergence estimates. *Linear Algebra Appl.*, 322(1):87–104, 2001. https://doi.org/10.1016/S0024-3795(00)00236-6.

[26] K. Neymeyr. A Geometric Theory for Preconditioned Inverse Iteration Applied to a Subspace. *Mathematics of Computation*, 71(237):197–216, 2002. http://www.jstor.org/stable/2698867.

[27] K. Neymeyr and A. L. Skorokhodov. Preconditioned gradient-type iterative methods in a subspace for partial generalized symmetric eigenvalue problems. *SIAM J. Numer. Anal.*, 31(4):1226–1239, 1994. https://doi.org/10.1137/0731064.

[28] T. Nottoli, I. Giannì, A. Levitt, and F. Lipparini. A robust, open-source implementationof the locally optimal block preconditioned conjugate gradient for large eigenvalue problems inquantum chemistry. [Research Report] hal-04094087, Theoretical Chemistry Accounts: Theory, Computation, and Modeling, 2023.

[29] E. Ovtchinnikov. Cluster robustness of preconditioned gradient subspace iteration eigensolvers. *Linear Algebra Appl.*,

415(1):140–166, 2006. https://doi.org/10.1016/j.laa.2005.06.039.

[30] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM J. Sci. Comput.*, 28(5):1651–1674, 2006.

[31] Y. Saad. *Iterative Methods for Sparse Linear Systems, 2nd ed.* SIAM, Philadelphia, 2003. https://epubs.siam.org/doi/book/10.1137/1.9780898718003.

[32] Y. Saad. *Numerical Methods for Large Eigenvalue Problems, 2nd ed.* SIAM, Philadelphia, 2011. https://doi.org/10.1137/1.9781611970739.

[33] Y. Saad and M. H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Computing*, 7(3):856–869, 1986. https://doi.org/10.1137/0907058.

[34] E. Timsit, L. Grigori, and O. Balabanov. Randomized Orthogonal Projection Methods for Krylov Subspace Solvers. *arXiv*, 2023. https://doi.org/10.48550/arXiv.2302.07466.

[35] J. A. Tropp. Improved analysis of the subsampled randomized Hadamard transform. *Advances in Adaptive Data Analysis*, 03(01n02):115–126, 2011. https://doi.org/10.1142/S1793536911000787.

[36] D. P. Woodruf. Sketching as a Tool for Numerical Linear Algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1-2):1–157, 2014. https://doi.org/10.1561/0400000060.

[37] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert. A fast randomized algorithm for the approximation of matrices. *Appl. Comput. Harmon. Anal.*, 25(3):335–366, 2008. https://doi.org/10.1016/j.acha.2007.12.002.

[38] C. Yang and C. Musco. Efficient block approximate matrix multiplication. In *31st Annual European Symposium on Algorithms (ESA 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

Appendix A. A robust version of LOBPCG [11, 28] (with $B = I_n$). Afther wide application of Knyazev's LOBPCG algorithm [17,18], some researchers noticed that an inappropriate choice of the basis of the trial subspace can lead to ill-conditioned Gram matrices in the RR processes that can delay convergence or produce inaccurate eigenpairs. For remedy, [11,28] proposed a robust version of LOBPCG with different ways of the basis selection for the case of standard eigenvalue problem (i.e., $B = I_n$ in Equation (1.1)). For the sake of distinguish, we denote this robust version as rLOBPCG and describe its pseudocode in Algorithm 4.

In rLOBPCG, the trial subspace is

$$\mathcal{W}_j = \text{Span}\{X_j, R_{j\,J}, P_{j\,J}\} \in \mathbb{R}^{n\times p_{Y_J}} \ (p_{Y_J} = p_{X_j} + p_{R_{j\,J}} + p_{P_{j\,J}} \lll n) \tag{A.1}$$

with solft-locking, the computed block eigenvector of the Gram matrices is Y in the form (2.7) according to the number of columns in X_j, $R_{j\,J}$, and $P_{j\,J}$, and the way to update the eigenvector is

$$X^{new} = X_j Y_X + R_{j\,J} Y_R + P_{j\,J} Y_P = \mathcal{W}_j Y.$$

Compared to Knyazev's LOBPCG algorithm, there are two things listed below have changed within this rLOBPCG algorithm.

The way to update the block search direction. With the above information about the trial subspace and the computed block eigenvector, rLOBPCG updates the new block search direction at the end of j-th iteration as

$$P^{new} = X^{new} - X_j = X_j(Y_X - I) + R_{j\,J} Y_R + P_{j\,J} Y_P = \mathcal{W}_j \tilde{Y}, \tag{A.2}$$

with $\tilde{Y} = [Y_X - I; Y_R; Y_P] \in \mathbb{R}^{p_{Y_J}\times p}$. We noticed that this way of update P^{new} in rLOBPCG can be viewed as a special case for LOBPCG when the Y_X in the right-hand side of (2.11) satisfy $Y_X = I$. Note that the P^{new} will gradually converge to zero since the new approximated eigenvector X^{new} and its previous value X_j will get closer to each other along the iteration processes, so special measures need to be taken to overcome the potential numerical instability. That is to let the new computed P^{new} in Equation (A.2) be orthogonal to X^{new}, which is realized implicitly by applying the orthonormalization to the computed block eigenvector Y of the Gram matrices as

$$Y^P = \tilde{Y} - YY^T\tilde{Y} \tag{A.3}$$

such that $Y^T Y^P = 0$. Then, change this new computed Y^P into the $\tilde{Y}$ in (A.2) to update new search direction as $P^{new} = \mathcal{W}_j Y^P$ such that $X^{newT} P^{new} = 0$.

The B-orthonormalization of the block residual to a subspace spanned by the block eigenvector and block search direction. Compared to the B-orthogonalization of the block residual R_j to the block eigenvector X_j described in Section 2.1.2, the rLOBPCG implements the B-orthogonalization of R_j to Span($[X_j, P_{j\,J}]$) (a subspace spanned by the columns of block eigenvector and block search direction) as

$$R_{j\,J} = R_{j\,J} - [X_j, P_{j\,J}]([X_j, P_{j\,J}], R_{j\,J})$$

such that $[X_j, P_{j\,J}]^T R_{j\,J} = 0$. Because of this orthogonality and ${X_j}^T P_j = 0$, the Gram matrix G_B shown in Equation (2.16) could be simplified as $G_B = I_{p_{Y_J}}$ when $B = I_n$.

Due to the aforementioned modifications, the trial subspace $\mathcal{W}j \in \mathbb{R}^{n\times pY_J}$ of rLOBPCG shown in (A.1) owns p_{Y_J} orthonormal columns. In the contrast, for Knyazev's LOBPCG, only each block vector possesses orthonormal columns, and the entire set of columns in $\mathcal{W}_j$ is linearly independent.

Based on Section 2.3.1-2.3.3, it is easy to deduce the corresponding sketched version of the rLOBPCG method. Thus we omit the details but refer the interested reader to Algorithm 5 of Appendix B for its pseudocode under the case with $B = I_n$.

Algorithm 4 *rLOBPCG: A roubust version of LOBPCG (with $B = I_n$):*

Require: As the **Require** described in the Algorithm 1

1: Step 1-5 of Algorithm 1
2: **for** $j = 1, 2, \ldots, MaxIterations$: **do**
3: Compute the active residuals: $R_J = W_{jA} - W_{jB}\Lambda_j$ (we have $R_{j\,J} \perp X_j$ because of Step 4)
4: full_$X_j(:, J) = X_j$, full_$AX_j(:, J) = W_{jA} = AX_j = W_{jA}$, full_$BX_j(:, J) = W_{jB} = BX_j$, full_$R_j(:, J) = R_j$, full_$P_j(:, J) = P_j$, full_$AP_j(:, J) = Q_{jA} = AP_j$, full_$BP_j(:, J) = Q_{jB} = BP_j$, full_$\Lambda_j(:, J) = \Lambda_j$
5: Step 8 of Algorithm 1 for updating index set J
6: Compute the active vectors with the update index set J (here length$(J) \le p$): $X_j =$ full_$X_j(:, J)$; $\Lambda_j =$ full_$\Lambda_j(:, J)$; $R_{j\,J} =$ full_$R_j(:, J)$; $P_{j\,J} =$ full_$P_j(:, J)$; $Q_{jA\,J} =$ full_$AP_j(:, J)$; $Q_{jB\,J} =$ full_$BP_j(:, J)$
7: Step 10-11 of Algorithm 1
8: B-orthogonalize (s.t., [full_$X_j, P_{j\,J}$]$\perp_B R_{j\,J}$) the preconditioned active residuals to [full_$X_j, P_{j\,J}$]: $R_{j\,J} = R_{j\,J} -$ [full_$X_j, P_{j\,J}$]$((B$[full_$X_j, P_{j\,J}$]$)^T R_{j\,J})$
9: Compute $Z_{jB\,J}$ as $Z_{jB\,J} = BR_{j\,J}$, and B-orthonormalize $R_{j\,J}$ (s.t., $(R_{j\,J}, Z_{jB\,J}) = I$): $T = \texttt{chol}((R_{j\,J}, Z_{jB\,J}))$; $R_{j\,J} = R_{j\,J}T^{-1}$; $Z_{jB\,J} = Z_{jB\,J}T^{-1}$; compute $Z_{jA\,J}$: $Z_{jA\,J} = AR_{j\,J}$
10: **if** $j > 1$ **then**
11: B-orthonormalize $P_{j\,J}$ (s.t., $(P_{j\,J}, Q_{jB\,J}) = I$): $T = \texttt{chol}((P_{j\,J}, Q_{jB\,J}))$
12: Update $P_{j\,J} = P_{j\,J}T^{-1}$; $Q_{jA\,J} = Q_{jA\,J}T^{-1}$; $Q_{jB\,J} = Q_{jB\,J}T^{-1}$
13: **end if**
/ Perform the RR procedure for the pencil $A - B\Lambda$ in the B-orthonormalized subspace: $\mathcal{W}_j = Span\{X_j, R_{j\,J}, P_{j\,J}\} \in \mathbb{R}^{p_{Y_J} \times p_{Y_J}}$ ($p_{Y_J} = p_{X_j} + p_{R_{j\,J}} + p_{P_{j\,J}} \lll n$) to update the activate Ritz values $\Lambda_{j\,J}$ and the corresponding Ritz vectors $X_{j\,J}$: */*
14: Compute the symmetric Gram matrices G_A described below, and $G_B = I$:
15: **if** $j > 1$ **then**
16: $G_A \in \mathbb{R}^{p_{Y_J} \times p_{Y_J}}$ in (2.15) with J for columns of $R_{j\,J}$ and $P_{j\,J}$
17: **else**
18: $G_A = \begin{bmatrix} \Lambda_j & (X_j, Z_{jA\,J}) \\ (R_{j\,J}, W_{jA}) & (R_{j\,J}, Z_{jA\,J}) \end{bmatrix} \in \mathbb{R}^{(p+p_{R_{j\,J}}) \times (p+p_{R_{j\,J}})}$
19: **end if**
20: Solve the ordinary eigenvalue problem: $G_A Y = Y\Lambda_{j+1}$, where the first p_J eigenvalues in increasing order are in the diagonal matrix $\Lambda_{j+1} \in \mathbb{R}^{p_J \times p_J}$, and the orthonormalized eigenvectors are the columns of Y (i.e., $(Y, Y) = I, (Y, G_A Y) = \Lambda_{j+1}$)
/ Update Ritz vectors and the recycling tall matrices: */*
21: **if** $j > 1$ **then**
22: Partition $Y = [Y_X; Y_R; Y_P], \tilde{Y} = [Y_X - I; Y_R; Y_P] \in \mathbb{R}^{p_{Y_J} \times p}$ ($p_{Y_J} = 3 \times p$ if no partial convergence occurs) according to the number of columns in X_j, $R_{j\,J}$, and $P_{j\,J}$, respectively.
23: Compute $\tilde{Y} = \text{ortho}(\tilde{Y}, Y)$ (the ortho$(\cdot)$ is an orthonormalization process described in [28])
24: Update $X_{j+1} = [X_j, R_{j\,J}, P_{j\,J}]Y$; $W_{j+1\,A} = [W_{jA}, Z_{jA\,J}, Q_{jA\,J}]Y$; $W_{j+1\,B} = [W_{jB}, Z_{jB\,J}, Q_{jB\,J}]Y$
25: Update $P_{j+1} = [X_j, R_{j\,J}, P_{j\,J}]\tilde{Y}$; $Q_{j+1\,A} = [W_{jA}, Z_{jA\,J}, Q_{jA\,J}]\tilde{Y}$; $Q_{j+1\,B} = [W_{jB}, Z_{jB\,J}, Q_{jB\,J}]\tilde{Y}$
26: **else**
27: Partition $Y = [Y_X; Y_R], \tilde{Y} = [Y_X - I; Y_R] \in \mathbb{R}^{(p+p_{R_J}) \times p}$; $\tilde{Y} = \text{ortho}(\tilde{Y}, Y)$
28: Update $X_{j+1} = [X_j, R_{j\,J}]Y$; $W_{j+1\,A} = [W_{jA}, Z_{jA\,J}]Y$; $W_{j+1\,A} = [W_{jB}, Z_{jB\,J}]Y$
29: Update $P_{j+1} = [X_j, R_{j\,J}]\tilde{Y}$; $Q_{j+1\,A} = [W_{jA}, Z_{jA\,J}]\tilde{Y}$; $Q_{j+1\,A} = [W_{jB}, Z_{jB\,J}]\tilde{Y}$
30: **end if**
31: **end for**
32: **return** the approximations Λ_{j+1} and X_{j+1} to the smallest eigenvalues and corresponding eigenvectors

Appendix B. The sketched version of rLOBPCG.

Algorithm 5 *srLOBPCG: Sketched version of rLOBPCG with RR and implicitly compued sketched vectors*

Require: As the **Require** described in the Algorithm 2

Step 1-5 of the Algorithm 1
Compute $X_1{}^{\Theta} = \Theta X_1,\ W_{1B}{}^{\Theta} = \Theta W_{1B} \in \mathbb{R}^{t\times p}$
Define the index set J of active iterates to be $\{1, ..., p\}$
for $j = 1, 2, \ldots, MaxIterations$: **do**
  Step 7-11 of the Algorithm 1
  Compute the sketched preconditioned active residuals $R_{jJ}{}^{\Theta} = \Theta R_{jJ}$
  if $j > 1$ & B-Θ-orthogonalize preconditioned active residuals R_{jJ} to subspace $[W_{jB}{}^{\Theta}, P_{jJ}{}^{\Theta}]$ **then**
    Compute $R_{jJ}{}^{\Theta} = R_{jJ}{}^{\Theta} - [X_j{}^{\Theta}, P_{jJ}{}^{\Theta}]([W_{jB}{}^{\Theta}, P_{jJ}{}^{\Theta}], R_{jJ}{}^{\Theta})$, and update $R_{jJ} = R_{jJ} - [X_j, P_{jJ}]([W_{jB}{}^{\Theta}, P_{jJ}{}^{\Theta}], R_{jJ}{}^{\Theta})$
  else if B-Θ-orthogonalize preconditioned active residuals R_{jJ} to subspace $W_{jB}{}^{\Theta}$ **then**
    Compute $R_{jJ}{}^{\Theta} = R_{jJ}{}^{\Theta} - X_j{}^{\Theta}(W_{jB}{}^{\Theta}, R_{jJ}{}^{\Theta})$, and update $R_{jJ} = R_{jJ} - X_j(W_{jB}{}^{\Theta}, R_{jJ}{}^{\Theta})$
  end if
  Compute Z_{jBJ} as $Z_{jBJ} = BR_{jJ}$, and sketch Z_{jBJ}: $Z_{jBJ}{}^{\Theta} = \Theta Z_{jBJ}$, B-Θ-orthonormalize R_{jJ} (s.t., $(R_{jJ}, Z_{jBJ})_{\Theta} = I$) as process in Equation (2.22) in Section 2.3.1. Update $R_{jJ}{}^{\Theta} = R_{jJ}{}^{\Theta}\, T^{-1}$, $Z_{jBJ}{}^{\Theta} = Z_{jBJ}{}^{\Theta}\, T^{-1}$ and compute Z_{jAJ} as $Z_{jAJ} = AR_{jJ}$
  if $j > 1$ **then**
    B-Θ-orthonormalize P_{jJ} (s.t., $(P_{jJ}, Q_{jBJ})_{\Theta} = I$) as process in Equation (2.23) in Section 2.3.1. Update $P_{jJ}{}^{\Theta} = P_{jJ}{}^{\Theta}\, T^{-1}$, $Q_{jB}{}^{\Theta} = Q_{jB}{}^{\Theta}\, T^{-1}$, $Q_{jAJ}{}^{\Theta} = Q_{jAJ}{}^{\Theta}\, T^{-1}$
  end if
  Step 14-20 of Algorithm 4: Perform the RR Procedure to solve a generalized eigenvalue problem
  / Update Ritz vectors and the recycling tall matrices: */*
  if $j > 1$ **then**
    Partition $Y = [Y_X; Y_R; Y_P] \in \mathbb{R}^{p_{Y_J}\times p}$ according to the number of columns in X_j, R_{jJ}, and P_{jJ}. And define $\tilde{Y} = [Y_X - I; Y_R; Y_P] \in \mathbb{R}^{p_{Y_J}\times p}$
    Compute and store $P_{j+1}{}^{\Theta} = R_{jJ}{}^{\Theta}Y_R + P_{jJ}{}^{\Theta}Y_P$, $Q_{j+1B}{}^{\Theta} = Z_{jBJ}{}^{\Theta}Y_R + Q_{jB}{}^{\Theta}Y_P$ (for Step 8 and Step 14); $X_{j+1}{}^{\Theta} = X_j{}^{\Theta}Y_X + P_{j+1}{}^{\Theta}$, $W_{j+1B}{}^{\Theta} = W_{jB}{}^{\Theta}Y_X + Q_{j+1B}{}^{\Theta}$ (for Step 8-10)
    Update $\tilde{P}_{j+1} = R_{jJ}Y_R + P_{jJ}Y_P$, $\widetilde{Q_{j+1A}} = Z_{jAJ}Y_R + Q_{jAJ}Y_P$, $\widetilde{Q_{j+1B}} = Z_{jBJ}Y_R + Q_{jBJ}Y_P$
    Update $X_{j+1} = X_jY_X + \tilde{P}_{j+1}$, $W_{j+1A} = W_{jA}Y_X + \widetilde{Q_{j+1A}}$, $W_{j+1B} = W_{jB}Y_X + \widetilde{Q_{j+1A}}$
    if update P_{j+1} by previous $[R_{jJ},\ P_{jJ}]$ **then**
      Update $P_{j+1} = \tilde{P}_{j+1}$, $Q_{j+1A} = \widetilde{Q_{j+1A}}$, $Q_{j+1B} = \widetilde{Q_{j+1B}}$
    else if update P_{j+1} by the trial space $\mathcal{W}_j = \text{Span}\{X_j, R_{jJ}, P_{jJ}\} \in \mathbb{R}^{n\times p_Y}$ (refer to [28]) **then**
      Orthogonalize the $\tilde{Y}$ to the orthogonalized Y: $\tilde{Y} = \tilde{Y} - Y(Y, \tilde{Y})$
      Update $P_{j+1} = \mathcal{W}_j\tilde{Y} = [X_j, R_{jJ}, P_{jJ}]\tilde{Y}$, $Q_{j+1A} = A\mathcal{W}_j\tilde{Y} = [W_{jA}, Z_{jAJ}, Q_{jAJ}]\tilde{Y}$, $Q_{j+1B} = B\mathcal{W}_j\tilde{Y} = [W_{jB}, Z_{jBJ}, Q_{jBJ}]\tilde{Y}$
    end if
  else
    Partition $Y = [Y_X; Y_R] \in \mathbb{R}^{(p+p_{R_{jJ}})\times p}$ according to the number of columns in X_j and R_{jJ}
    Compute and store $P_{j+1}{}^{\Theta} = R_{jJ}{}^{\Theta}Y_R$, $Q_{j+1B}{}^{\Theta} = Z_{jBJ}{}^{\Theta}Y_R$ (or Step 8 and Step 14); $X_{j+1}{}^{\Theta} = X_j{}^{\Theta}Y_X$, $W_{j+1B}{}^{\Theta} = W_{jB}{}^{\Theta}Y_X$ (for Step 8-10)
    Update $\tilde{P}_{j+1} = R_{jJ}Y_R$; $\widetilde{Q_{j+1A}} = Z_{jAJ}Y_R$; $\widetilde{Q_{j+1B}} = Z_{jBJ}Y_R$
    Update $X_{j+1} = X_jY_X + \tilde{P}_{j+1}$, $W_{j+1A} = W_{jA}Y_X + \widetilde{Q_{j+1A}}$, $W_{j+1B} = W_{jB}Y_X + \widetilde{Q_{j+1A}}$
    Step 23 and Step 25-26: To update $P_{j+1}, Q_{j+1A}, Q_{j+1B}$
  end if
end for
return the approximations Λ_{j+1} and X_{j+1} to the smallest eigenvalues and corresponding eigenvectors

Appendix C. Numerical results of a massive large sparse matrices with $t = l$.

TABLE C.1

Numerical results of LOBPCG variants with `chol` *for the B-/B-Θ-orthonormalization in terms of both $\#iter$ and $\#time(s)$ for all SPD matrices listed in Table 3.1 with the $p = 10$ eigenparis to be computed, $t = l = 2 \times m \times log(n)/log(m)$, and $\epsilon = 10^{-4}$. The superscript $^{*(i/p)}$ indicates that there are i eigenpairs ($0 < i \leq p$) out of the whole p are not converged within the maximal iteration steps.*

Matrix	$\#iter$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG	$\#time(s)$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG
af_shell3	$50000^{*(p/p)}$ / $50000^{*(p/p)}$ / $3060^{*(p/p)}$	58191.7152 / 69959.9576 / 4042.0799
apache1	**8377** / 25751 / 22577	685.5327 / 3127.6916 / 1485.2969
bodyy4	3441 / **2662** / 4515	60.8325 / 145.6819 / 108.3525
bodyy5	10461 / 10678 / **8814**	223.9471 / 638.0448 / 332.9481
bodyy6	33744 / 31539 / **31507**	815.3969 / 2270.9582 / 1410.8943
boneS01	1807 / 1692 / **1686**	298.5458 / 487.6838 / 344.3694
cant	14364 / 16255 / **10790**	1147.515 / 2236.4811 / 1134.2142
cbuckle	$50000^{*(p/p)}$ / $50000^{*(9/p)}$ / $50000^{*(p/p)}$	2578.0864 / 4179.9783 / 3126.4563
cfd1	838 / 825 / **781**	72.6699 / 169.9087 / 98.0807
cfd2	**1816** / 1827 / 1921	356.6375 / 630.0687 / 424.9975
crystm02	1 / 1 / 1	0.19666 / 0.18304 / 0.17966
crystm03	1 / 1 / 1	0.1176 / 0.11247 / 0.13536
Dubcova1	184 / 188 / 184	5.8309 / 9.4585 / 6.4943
Dubcova2	**289** / 465 / 367	27.6319 / 63.9986 / 35.2451
Dubcova3	**258** / 274 / 274	52.53 / 112.6756 / 67.4197
finan512	254 / 254 / **243**	28.6832 / 60.4514 / 37.0953
fv1	265 / **182** / 215	5.9695 / 8.6119 / 6.6753
fv2	**140** / 181 / 179	4.3522 / 9.138 / 6.4275
fv3	**140** / 179 / 179	4.1413 / 9.1372 / 6.1779
G2_circuit	$50000^{*(1/p)}$ / $50000^{*(1/p)}$ / **16508**	4826.8928 / 12038.6758 / 3097.8449
Kuu	548 / 548 / 548	14.6049 / 22.3613 / 16.8009
minsurfo	1013 / 841 / **818**	62.77 / 122.921 / 74.522
nd6k	**2712** / 3370 / 2821	201.7934 / 366.1987 / 239.9164
nd12k	4550 / 4431 / **3377**	489.68 / 746.0664 / 473.6129
nd24k	**6137** / 7847 / 10616	1171.4693 / 2153.4075 / 1771.2421
obstclae	556 / 558 / **472**	23.1745 / 57.7073 / 33.4985
Pres_Poisson	**2362** / 2694 / 2567	95.9393 / 168.3822 / 120.0484
pdb1HYS	**1869** / 6827 / 2372	155.855 / 775.7444 / 238.2939
qa8fm	3 / 3 / 3	0.40802 / 0.67466 / 0.56305
ted_B	10 / 10 / 10	0.49506 / 0.7511 / 0.6012
ted_B_unscaled	2 / 2 / 2	0.12105 / 0.14042 / 0.14559
thermal1	658 / **551** / 571	79.6275 / 154.2 / 98.3406
torsion1	556 / 486 / **443**	37.1215 / 73.5062 / 44.8412
wathen100	779 / 1847 / **652**	35.7605 / 126.2668 / 36.2006
wathen120	339 / 425 / **338**	23.6703 / 59.8576 / 33.2119
bundle1	16309 / **15230** / 15776	411.8445 / 600.8327 / 409.1157
shallow_water1	713 / 700 / **662**	47.9217 / 104.2028 / 53.3687
shallow_water2	997 / 982 / **953**	106.8643 / 186.0694 / 106.6338

TABLE C.2
Numerical results of LOBPCG variants with `chol` *for the B-/B-Θ-orthonormalization in terms of both $\#iter$ and $\#time(s)$ for all SYS matrices listed in Table 3.2 with $t = l = 2 \times m \times log(n)/log(m)$, $p = 10$ eigenparis to be computed, and $\epsilon = 10^{-4}$.*

Matrix	$\#iter$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG	$\#time(s)$ LOBPCG / sLOBPCG$_{\text{e-s}}$ / sLOBPCG
af_shell1	$50000^{*(1/p)}$ / **49932** / $50000^{*(p/p)}$	46519.4361 / 68531.5316 / 60187.0462
3dtube	**521** / 537 / 1070	24.0108 / 45.7385 / 46.89
barth5	**263** / 298 / 300	4.7017 / 11.0823 / 6.7026
bratu3d	312 / 194 / **163**	6.5368 / 12.3161 / 7.0949
copter1	**309** / 358 / 356	5.0411 / 16.997 / 8.4176
c-55	2403 / **1549** / 1702	83.01 / 171.5085 / 109.5617
darcy003	**574** / 585 / 594	223.2386 / 393.775 / 260.8115
d_pretok	16536 / **15914** / 16839	3476.9654 / 6896.7456 / 4648.0397
ecology1	2043 / **1937** / 1993	3463.2613 / 5400.2671 / 3987.438
fcondp2	287 / 293 / **249**	67.0584 / 117.2711 / 71.6106
gearbox	137 / 178 / **132**	32.9626 / 79.321 / 43.4845
gupta1	68 / 68 / 68	2.3454 / 4.3801 / 2.9343
helm3d01	84 / 84 / 84	2.8235 / 5.5746 / 3.5432
helm2d03	**627** / 639 / 659	309.6515 / 556.4794 / 374.2936
ins2	**21** / 23 / 24	7.2733 / 15.0166 / 9.814
k1_san	3079 / **1075** / 1367	121.9847 / 172.6896 / 91.3173
Lin	938 / **725** / 734	196.3639 / 288.7714 / 196.1482
lp1	47 / 47 / 47	33.1668 / 70.2819 / 42.6356
Matrix_1	1226 / 605 / **605**	44.4931 / 44.3248 / 33.0606
Matrix_2	**885** / 1163 / 1238	436.7147 / 561.2614 / 510.4952
mario001	255 / 281 / **281**	10.7535 / 29.4323 / 17.1613
mario002	**574** / 576 / 621	216.136 / 414.1154 / 256.4557
nemeth01	1175 / 1088 / **941**	17.2773 / 38.6624 / 22.4024
nlpkkt200	2849 / 4147 / **2394**	75324.8433 / 146130.2151 / 76235.5588
onera_dual	**203** / 223 / 220	19.6708 / 42.4751 / 27.4636
pct20stif	438 / **312** / 409	16.0717 / 88.4485 / 16.7503
pkustk03	406 / 394 / **394**	18.3095 / 34.1514 / 20.3616
qa8fk	95 / **93** / 105	8.4976 / 19.7928 / 12.765
rajat06	376 / 376 / **309**	3.7484 / 9.5159 / 4.6122
rajat07	**488** / 489 / 489	5.6224 / 12.8694 / 7.1177
rajat09	**391** / 541 / 563	7.8982 / 25.1533 / 13.8701
rajat10	818 / **345** / 706	16.2255 / 21.0323 / 18.3344
saylr4	712 / **634** / 797	3.7118 / 6.0596 / 4.7747
struct3	**110** / 114 / 141	7.153 / 14.1237 / 10.7336
tandem_vtx	**319** / 461 / 325	5.0124 / 18.111 / 7.4846
tuma1	260 / **210** / 259	6.619 / 14.919 / 10.1883
tuma2	237 / 308 / **200**	3.5863 / 10.364 / 4.9175
turon_m	**737** / 929 / 1666	187.289 / 380.0812 / 535.6576
brainpc2	799 / **796** / 812	38.9432 / 63.9516 / 40.8305
F1	45368 / 43432 / **31220**	17554.6462 / 28925.5126 / 14462.9188
kkt_power	**4216** / 4392 / 4663	21004.9903 / 28198.4593 / 24128.138