Int. J. Appl. Math. Comput. Sci., 2025, Vol. 35, No. 2, 281–292
DOI: 10.61822/amcs-2025-0020

A PROJECTION STRATEGY FOR IMPROVING THE PRECONDITIONER IN THE LOBPCG

TAILAI MA [a], SHULI SUN [a,*], FANGYI ZHENG [b], PU CHEN [a]

[a]Department of Mechanics and Engineering Science
Peking University, Yiheyuan Road No. 5, 100871 Beijing, China
e-mail: sunsl@mech.pk.edu.cn,{matailai,chenpu}@pku.edu.cn

[b]Intelligent Science & Technology Academy of CASIC
Fucheng Road No. 8, 100830 Beijing, China
e-mail: zhengfy1@pku.edu.cn

The computational methods for solving the generalized eigenvalue problems of real symmetric matrices are crucial in fields such as structural dynamics analysis. As the scale of the problems to be solved increases, higher efficiency in solving eigenvalue problems is demanded. The LOBPCG (locally optimal block preconditioned conjugate gradient) method is a promising iterative algorithm suitable for solving large-scale eigenvalue problems, capable of quickly solving multiple extreme eigenpairs. In the LOBPCG, the preconditioner can be executed by calling the truncated PCG to approximately solve the 'inner' linear system. However, the convergence rate of the LOBPCG is highly sensitive to the quality of its preconditioner. Only when paired with an appropriate preconditioner, the LOBPCG is notably efficient in minimizing the iterations needed for convergence. This paper proposed a projection strategy which can enhance the quality of the preconditioner, thus improving the overall efficiency and stability of the LOBPCG. The projection strategy first utilizes intermediate vectors from the PCG iterations to construct search subspaces and constraint subspaces for oblique projection, and then executes the oblique projection in truncated PCG when solving inner linear system. This oblique projection technique can find a more accurate approximate solution which minimizes the 2-norm residuals in the search subspace without significantly increasing computational cost, thereby improving the quality of the preconditioner, thus accelerating convergence of the LOBPCG. Numerical experiments show that the projection strategy can improve the LOBPCG algorithm significantly in terms of efficiency and stability.

Keywords: LOBPCG, preconditioner, preconditioned conjugate gradient (PCG), projection method.

1. Introduction

Eigenvalue problems, which involve determining the eigenvalues and corresponding eigenvectors of matrices, play a fundamental role in fields such as structural dynamics analysis. Consider a generalized symmetric definite eigenvalue problem

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x}, \tag{1}$$

where λ and $\boldsymbol{x}$ are the eigenvalues and the corresponding eigenvectors, and $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$ are symmetric positive definite matrices.

For solving eigenvalues of sparse symmetric matrices in the field of structural dynamics, there are already many efficient and stable methods, such as the subspace iteration method(Bathe and Wilson, 1973), the Lanczos method (Lanczos, 1950; Guarracino *et al.*, 2006), or the WYD method (Yuan *et al.*, 1989), etc. However, these methods necessitate employing direct linear system solvers to repeatedly solve the linear system during the solving process. With an increase in the problem scale, the computational burden of using direct linear system solvers escalates to impractical levels. Some methods employ a divide-and-conquer approach. For instance, AMLS (automated multi-level substructuring) (Bennighof and Lehoucq, 2004; Yin *et al.*, 2013) is a technique which operates by decomposing large systems into smaller, more manageable substructures and then solving these substructures at different levels.

*Corresponding author

Alternatively, to overcome the challenges inherent in direct methods for linear systems, several algorithms employ truncated iterative methods for linear systems. These iterative methods approximate the solution of linear systems, presenting a viable alternative to direct methods under computational constraints. The Jacobi-Davidson (JD) method (Sleijpen and Van der Vorst, 2000; Fan *et al*., 2014) and the LOBPCG (locally optimal block preconditioned conjugate gradient) method (Knyazev, 2001) are representative algorithms that utilize this strategy. Grounded in the conjugate gradient (CG) method (Collignon and Gijzen, 2010; Sulaiman *et al*., 2024) and coupled with local optimization strategies, the LOBPCG offers a robust solution for solving multiple extreme eigenpairs. By adopting a block-wise computational approach as described in (Knyazev *et al*., 2007) and integrating preconditioning techniques, the LOBPCG algorithm can simultaneously compute multiple eigenvalue-eigenvector pairs. Currently, several well-established libraries have implemented these methods, such as PRIMME (PReconditioned Iterative MultiMethod Eigensolver) (Stathopoulos and McCombs, 2010; Wu *et al*., 2017) and SLEPc (Scalable Library for Eigenvalue Problem Computations) (Roman *et al*., 2016) which bring state-of-the-art methods from 'bleeding edge' to production.

Several versions of the LOBPCG have been implemented, with these implementations primarily focusing on refining its orthogonalization process. For example, Hetmaniuk and Lehoucq (2006) suggested an additional $\boldsymbol{B}$-orthogonalization of the approximate eigenvectors, and Duersch *et al.* (2018) developed a number of techniques to enhance the Hetmaniuk–Lehoucq (HL) orthogonalization strategy. However, research on the LOBPCG preconditioner is insufficient. Kressner *et al.* (2023) propose mixed precision LOBPCG algorithms that use a (sparse) Cholesky factorization of $\boldsymbol{A}$ computed in lower precision as preconditioner.

In this paper, we focus on the preconditioner executed by calling a truncated PCG solver to solve the inner linear system $\boldsymbol{AW} = \boldsymbol{Q}$ approximately (where $\boldsymbol{Q}$ is the block of residuals), which means that an accurate solution is not needed, and the PCG solver just needs to iterate for a specific number of steps (Knyazev *et al*., 2007). When paired with an appropriate preconditioner, the LOBPCG has been proven to be highly effective in reducing the number of iterations required for convergence. Nevertheless, the LOBPCG is highly sensitive to the preconditioner. It only exhibits a high convergence rate when the preconditioner is sufficiently effective. An inappropriate preconditioner can slow down convergence or even lead to divergence. Consequently, to achieve higher efficiency, the performance of the preconditioner for the LOBPCG must be improved, and the projection method mentioned below has the potential to enhance the preconditioner.

The projection method is an important technique, particularly effective in the iterative method for solving linear systems. It restricts the search space for the solution to a smaller, more relevant subspace, thereby accelerating the convergence. Most of the existing practical iterative techniques for solving large linear systems utilize a projection process in one way or another (Saad, 2003; Yuan *et al*., 2021). In fact, projection methods can also be used to enhance the effectiveness of the preconditioner. Recently, Geng and Sun (2023) applied a projection-improved strategy to the preconditioner for (F)GMRES, significantly accelerating the algorithm's convergence. Inspired by this, we improve the preconditioner for the LOBPCG based on the projection method, aiming to enhance both its stability and efficiency, thereby boost the convergence of the LOBPCG algorithm.

The proposed projection strategy begins by using intermediate vectors from the truncated PCG iterations to construct the search and constraint subspaces for oblique projection. It then applies an oblique projection within the truncated PCG for accelerating the convergence of the inner linear system $\boldsymbol{AW} = \boldsymbol{Q}$. This oblique projection can achieve a more accurate approximation that minimizes the 2-norm residuals within the search subspace, without significantly raising the computational expense. Our projection strategy enhances the effectiveness of the preconditioner, leading to an improved overall efficiency of the LOBPCG. In the future, the proposed projection strategy could be integrated into eigenvalue solvers, such as PRIMME and SLEPc. By incorporating our projection technique, libraries like PRIMME and SLEPc may benefit from improved performance and robustness in solving large-scale eigenvalue computations.

The subsequent sections of the paper are organized as follows. In Section 2, we provide an overview of the LOBPCG algorithm and the oblique projection method. Section 3 introduces the projection strategy for the preconditioner in LOBPCG as well as the discussion of the strategy. In Section 4, we present the numerical results followed by a brief analysis. Finally, Section 5 provides conclusions and the outlook for the future research.

2. LOBPCG algorithm and oblique projection method

2.1. LOBPCG algorithm. The LOBPCG can be seen as an extension of the (preconditioned) conjugate gradient algorithm for eigenvalue problems (Feng and Owen, 1996). Consider computing only the smallest eigenpair, i.e., the block size of $bs = 1$. In the PCG algorithm for eigenvalue problems, the parameter δ_k is

determined to minimize the Rayleigh quotient:

$$\rho\left(\boldsymbol{x}_{k+1}\right)=\rho\left(\boldsymbol{x}_k+\delta_k\boldsymbol{p}_k\right), \tag{2}$$

where $\boldsymbol{p}_k$ is the search direction, and the parameter α_k is determined to make search directions

$$\boldsymbol{p}_k=-\boldsymbol{g}_k+\alpha_k\boldsymbol{p}_{k-1} \tag{3}$$

conjugate, where $\boldsymbol{g}_k$ is the gradient of the Rayleigh quotient ρ.

Knyazev (2001) proposed to rewrite the Rayleigh quotient as

$$\rho\left(\boldsymbol{x}_{k+1}\right)=\rho\left(\boldsymbol{x}_k+\tau_k\boldsymbol{p}_k+\gamma_k\boldsymbol{g}_k\right) \tag{4}$$

and he suggested that these two parameters τ_k and γ_k should be selected to minimize the Rayleigh quotient $\rho\left(\boldsymbol{x}_{k+1}\right)$ using the Rayleigh-Ritz procedure, which he referred to as a 'locally optimal.' Clearly, this result is smaller than before,

$$\begin{aligned}\min_{\tau_k,\gamma_k}\rho\left(\boldsymbol{x}_k+\tau_k\boldsymbol{p}_k+\gamma_k\boldsymbol{g}_k\right)\\ \leq\min_{\delta_k}\left(\boldsymbol{x}_k-\delta_k\left(\boldsymbol{g}_k-\alpha_k\boldsymbol{p}_{k-1}\right)\right).\end{aligned} \tag{5}$$

A block version of the LOBPCG for finding $bs>1$ smallest eigenpairs performs the Rayleigh-Ritz procedure on a $3bs$-dimensional trial subspace $\operatorname{span}\{\boldsymbol{S}\}(\boldsymbol{S}=[\boldsymbol{X},\boldsymbol{P},\boldsymbol{W}]$, where $\boldsymbol{X}$ is the block of approximate eigenvectors, $\boldsymbol{P}$ is the block of 'directions' of the LOBPCG and $\boldsymbol{W}$ is the block of preconditioned residuals.

However, the basic LOBPCG can sometimes encounter serious numerical issues. In the basic LOBPCG algorithm, vector sets $\boldsymbol{P}$ and $\boldsymbol{W}$ are orthogonalized separately with respect to the $\boldsymbol{B}$-metric. While this ensures $\boldsymbol{B}$-orthogonality within each set ($\boldsymbol{P}$ and $\boldsymbol{W}$), it does not guarantee $\boldsymbol{B}$-orthogonality between the vectors of $\boldsymbol{P}$ and $\boldsymbol{W}$. This discrepancy may lead to severe numerical issues. In the software package BLOPEX (Knyazev *et al.*, 2007), a failure in achieving $\boldsymbol{B}$-orthogonality between the vectors of $\boldsymbol{P}$ and $\boldsymbol{W}$ results in non-$\boldsymbol{B}$-orthogonality during the Rayleigh-Ritz process. This, in turn, makes the projection matrix $\boldsymbol{S}^T\boldsymbol{B}\boldsymbol{S}$ on the right-hand side indefinite or even rank deficient, ultimately leading to a breakdown in the algorithm.

To overcome this numerical difficulty, Hetmaniuk and Lehoucq (2006) proposed a basis selection strategy which keeps $\boldsymbol{X}$, $\boldsymbol{W}$, and $\boldsymbol{P}$ in the subspace $\operatorname{span}\{\boldsymbol{S}\}$ mutually $\boldsymbol{B}$-orthogonal. They use a procedure ortho() to $\boldsymbol{B}$-orthonormalize $\boldsymbol{W}$ with respect to both $\boldsymbol{P}$ and $\boldsymbol{X}$ at each iteration. In this paper, we refer to the LOBPCG with basis selection as the 'stable version of LOBPCG' and we will use this version to test our projection strategy in Section 4. Algorithm 1 provides a pseudocode for the stable version of the LOBPCG. By doing this, the generalized eigenvalue problem is turned into a standard eigenvalue problem in the Rayleigh-Ritz procedure.

Algorithm 1. Stable version of LOBPCG.

Input: Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, the maximum number of iterations MaxIter, initial $\boldsymbol{X}$

Output: $\boldsymbol{X}$, λ, the lowest eigenpairs of the eigenvalue problem

1: Perform a Rayleigh-Ritz analysis: $\boldsymbol{X}^T\boldsymbol{A}\boldsymbol{X}\boldsymbol{Y}=\boldsymbol{X}^T\boldsymbol{B}\boldsymbol{X}\boldsymbol{\Theta}\boldsymbol{Y}$
2: $\boldsymbol{X}=\boldsymbol{X}\boldsymbol{Y}$
3: $\boldsymbol{Q}=\boldsymbol{A}\boldsymbol{X}-\boldsymbol{\Theta}\boldsymbol{B}\boldsymbol{X}$
4: $\boldsymbol{P}=[\,]$
5: **for** $j=1,\ldots,$ MaxIter **do**
6: Apply the preconditioner $\boldsymbol{T}$ to the residuals: $\boldsymbol{W}=\boldsymbol{T}\boldsymbol{Q}$
7: Orthonormalize $\boldsymbol{W}$ against $\boldsymbol{X}$ and $\boldsymbol{P}$
8: $\boldsymbol{S}=[\boldsymbol{X},\boldsymbol{P},\boldsymbol{W}]$
9: Perform a Rayleigh-Ritz analysis: $\boldsymbol{S}^T\boldsymbol{A}\boldsymbol{S}\boldsymbol{Y}=\boldsymbol{S}^T\boldsymbol{B}\boldsymbol{S}\boldsymbol{\Theta}\boldsymbol{Y}$
10: $\boldsymbol{X}=\boldsymbol{S}\boldsymbol{Y}$
11: $\boldsymbol{Q}=\boldsymbol{A}\boldsymbol{X}-\boldsymbol{\Theta}\boldsymbol{B}\boldsymbol{X}$
12: Lock converged eigenvectors, exit if done
13: $\boldsymbol{P}=[\boldsymbol{0},\boldsymbol{P},\boldsymbol{W}]\boldsymbol{Y}$
14: **end for**

In the LOBPCG, the performance of the preconditioner $\boldsymbol{T}$ is crucial for the convergence rate of the algorithm. The preconditioner $\boldsymbol{T}$ is typically executed by calling a truncated PCG solver for solving the inner linear equation $\boldsymbol{A}\boldsymbol{W}=\boldsymbol{Q}$. Only when paired with an effective preconditioner, the LOBPCG is notably efficient in minimizing the time needed for convergence. An inappropriate preconditioner would slow down convergence or even lead to divergence. To enhance the efficiency for the LOBPCG, it is necessary to improve the quality of its preconditioner. Indeed, there exists considerable room for such improvement.

2.2. Oblique projection method. The projection method (Saad, 2003) is widely used for solving linear systems, effectively searches for approximate solutions satisfying constraints within a subspace. Consider a linear system

$$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}, \tag{6}$$

where $\boldsymbol{A}\in\mathbb{R}^{n\times n}$, and the vector $\boldsymbol{b}\in\mathbb{R}^n$ is the right-hand side of the system. The core concept of the projection method is to find an approximate solution to the problem from a subspace of $\mathbb{R}^n$, known as the search subspace. This search subspace, denoted as $\mathcal{K}$, is assumed to be m-dimensional, thereby necessitating m constraints for the approximation (where $m\ll n$).

A typical way of describing these constraints is to impose m (independent) orthogonality conditions.

Specifically, the residual vector $\boldsymbol{b} - \boldsymbol{A}\tilde{\boldsymbol{x}}$ is constrained to be orthogonal to m linearly independent vectors where $\tilde{\boldsymbol{x}}$ is the approximate solution. This requirement defines another subspace, termed $\mathcal{L}$, of dimension m, referred to as the subspace of constraints.

Then the approximate problem can be defined as

$$\text{Find } \tilde{\boldsymbol{x}} \in \mathcal{K} \text{ such that } \boldsymbol{b} - \boldsymbol{A}\tilde{\boldsymbol{x}} \perp \mathcal{L}. \tag{7}$$

If we intend to leverage the information embedded in an initial guess $\boldsymbol{x}_0$ towards achieving a solution, it is obvious that the approximation is sought within the affine space $\boldsymbol{x}_0 + \mathcal{K}$. This necessitates a slight adjustment to the previously stated formulation (7). The approximate problem should be redefined as follows:

$$\text{Find } \tilde{\boldsymbol{x}} \in \boldsymbol{x}_0 + \mathcal{K} \text{ such that } \boldsymbol{b} - \boldsymbol{A}\tilde{\boldsymbol{x}} \perp \mathcal{L}. \tag{8}$$

If the search subspace and the constraint subspace satisfy $\mathcal{L} = \boldsymbol{A}\mathcal{K}$, this configuration is termed the oblique projection.

Consider $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m]$ as an $n \times m$ matrix, where its column vectors $\boldsymbol{v}_i\,(i = 1, \ldots, m)$ form a basis for $\mathcal{K}$. Similarly, let $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m]$ be another $n \times m$ matrix, with column vectors $\boldsymbol{u}_i\,(i = 1, \ldots, m)$ forming a basis for $\mathcal{L}$. Then the approximate solution can be formulated as follows:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{V}\boldsymbol{\alpha}, \tag{9}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_m]^T$ is the coefficient vector. The corresponding residuals can be formulated as

$$\begin{aligned} \tilde{\boldsymbol{r}} &= \boldsymbol{b} - \boldsymbol{A}\tilde{\boldsymbol{x}} \\ &= \boldsymbol{b} - \boldsymbol{A}\left(\boldsymbol{x}_0 + \boldsymbol{V}\boldsymbol{\alpha}\right) = \boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{V}\boldsymbol{\alpha}. \end{aligned} \tag{10}$$

The orthogonality condition yields

$$\boldsymbol{U}^T\tilde{\boldsymbol{r}} = \boldsymbol{U}^T(\boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{V}\boldsymbol{\alpha}) = 0. \tag{11}$$

It follows that

$$\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{V}\boldsymbol{\alpha} = \boldsymbol{U}^T\tilde{\boldsymbol{r}}. \tag{12}$$

Then $\boldsymbol{\alpha}$ can be expressed as

$$\boldsymbol{\alpha} = (\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{V})^{-1}\boldsymbol{U}^T\boldsymbol{r}_0 \tag{13}$$

and the approximate solution can be rewritten as

$$\tilde{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{V}(\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{V})^{-1}\boldsymbol{U}^T\boldsymbol{r}_0. \tag{14}$$

When the oblique projection method ($\boldsymbol{U} = \boldsymbol{A}\boldsymbol{V}$) is applied, the method minimizes the residual 2-norm property in the search subspace $\mathcal{K}$

$$\begin{aligned} \tilde{\boldsymbol{x}} &= \arg\min_{\boldsymbol{\alpha}} \|\tilde{\boldsymbol{r}}\|_2 \\ &= \arg\min_{\boldsymbol{\alpha}} \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0 - \boldsymbol{A}\boldsymbol{V}\boldsymbol{\alpha}\|_2 \\ &= \boldsymbol{x}_0 + \boldsymbol{V}\left(\boldsymbol{U}^T\boldsymbol{U}\right)^{-1}\boldsymbol{U}^T\boldsymbol{r}_0. \end{aligned} \tag{15}$$

Through the oblique projection method, we have found an approximate solution in the search subspace with the smallest residual 2-norm. This approximate solution is evidently more accurate than the original one.

As an important technique, the projection method is often used directly in the iterative method for solving linear system. Recently, Geng and Sun (2023) applied a projection strategy in the preconditioning process of the (F)GMRES algorithm, markedly enhancing the stability and efficiency of the algorithm. This demonstrates the efficacy of the projection method in enhancing the preconditioner. Similarly, this paper applies the projection strategy to the preconditioner $\boldsymbol{T}$ within the LOBPCG algorithm, aiming to enhance both its stability and overall efficiency.

3. Projection strategy for LOBPCG preconditioner

3.1. Projection strategy for enhancing the preconditioner. The preconditioner $\boldsymbol{T}$ in Algorithm 1 can be executed by calling a truncated PCG solver to approximately solve the 'inner' linear system, which in our algorithm is $\boldsymbol{A}\boldsymbol{W} = \boldsymbol{Q}$. Designing a high-performance preconditioner is challenging because an ideal preconditioner should be inexpensive to obtain and able to greatly reduce the condition number of the matrix. These two desired properties often remain in conflict. Although increasing the number of 'inner' iterations of the truncated PCG solver could reduce the number of 'outer' iterations needed for convergence, too many inner iterations would reduce the overall efficiency. This is because even when $\boldsymbol{T}$ is equivalent to $\boldsymbol{A}^{-1}$, the convergence rate of the LOBPCG method remains linear (Knyazev *et al.*, 2007). Based on our experiments, it is revealed that an optimal number of inner iterations is around 10, both for ILU(1)-PCG and SPAI(1)-PCG, where ILU(1) and SPAI(1) are the preconditioners of the truncated PCG solver.

To further improve the efficiency and stability of the LOBPCG, we propose a projection strategy for the preconditioner in the LOBPCG. In Algorithm 1, the truncated PCG solver is repeatedly called to solve the inner linear systems. The proposed strategy can recycle the Krylov subspace created in the previous linear system to accelerate the convergence in solving the subsequent linear system by executing an additional projection. This enhances the performance of the preconditioner without markedly increasing the computational cost, thus improving the overall efficiency of the LOBPCG.

Suppose that each time $\boldsymbol{T}$ is executed, the truncated PCG solver iterates m steps. Then m conjugate directions $\{\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{m-1}\}$ in the internal iterations of the truncated PCG can construct a search subspace. The

Algorithm 2. Projection improved preconditioner for LOBPCG.

Input: matrix $\boldsymbol{A}$, right-hand side $\boldsymbol{q}$, initial guess $\boldsymbol{w}_0$ and parameter m

Output: Approximation $\tilde{\boldsymbol{w}}$

1: $\boldsymbol{r}_0 = \boldsymbol{q} - \boldsymbol{A}\boldsymbol{w}_0$
2: $\boldsymbol{z}_0 = \boldsymbol{M}^{-1}\boldsymbol{r}_0, \boldsymbol{g}_0 = \boldsymbol{z}_0$
3: **for** $k = 1, \ldots, m$ **do**
4: $\quad \boldsymbol{d}_{k-1} = \boldsymbol{A}\boldsymbol{g}_{k-1}$
5: $\quad a_{k-1} = \frac{\boldsymbol{r}_{k-1}^{\top}\boldsymbol{z}_{k-1}}{\boldsymbol{g}_{k-1}^{\top}\boldsymbol{d}_{k-1}}$
6: $\quad \boldsymbol{V}[:,k] = \boldsymbol{g}_{k-1}$
7: $\quad \boldsymbol{U}[:,k] = \boldsymbol{d}_{k-1}$
8: $\quad \boldsymbol{w}_k = \boldsymbol{w}_{k-1} + a_{k-1}\boldsymbol{g}_{k-1}$
9: $\quad \boldsymbol{r}_k = \boldsymbol{r}_{k-1} - a_{k-1}\boldsymbol{d}_{k-1}$
10: $\quad \boldsymbol{z}_k = \boldsymbol{M}^{-1}\boldsymbol{r}_k$
11: $\quad b_{k-1} = \frac{\boldsymbol{r}_k^{\top}\boldsymbol{z}_k}{\boldsymbol{r}_{k-1}^{\top}\boldsymbol{z}_{k-1}}$
12: $\quad \boldsymbol{g}_k = \boldsymbol{z}_k + b_{k-1}\boldsymbol{g}_{k-1}$
13: $\quad k = k + 1$
14: **end for**
15: Update the approximation:
$\tilde{\boldsymbol{w}} = \boldsymbol{w}_m + \boldsymbol{V}_{\text{prev}}\left(\boldsymbol{U}_{\text{prev}}^T\boldsymbol{U}_{\text{prev}}\right)^{-1}\boldsymbol{U}_{\text{prev}}^T\boldsymbol{r}_m$
16: Update the subspaces:$\boldsymbol{V}_{\text{prev}} = \boldsymbol{V}, \boldsymbol{U}_{\text{prev}} = \boldsymbol{U}$

matrix form of this search subspace can be represented as

$$\boldsymbol{V} = [\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{m-1}]. \tag{16}$$

Consider using the product of matrix $\boldsymbol{A}$ and conjugate directions $\boldsymbol{g}_i\,(i = 0, \ldots, m-1)$ to construct a constraint subspace. Then the matrix form of this constraint subspace can be represented as

$$\boldsymbol{U} = [\boldsymbol{A}\boldsymbol{g}_0, \boldsymbol{A}\boldsymbol{g}_1, \ldots, \boldsymbol{A}\boldsymbol{g}_{m-1}]. \tag{17}$$

It is important to note that $\boldsymbol{A}\boldsymbol{g}_k\,(k = 0, \ldots, m-1)$ can also be directly obtained from the internal iterations of truncated PCG without the additional sparse matrix-vector multiplication(SpMV) computations. Clearly, $\boldsymbol{U}$ and $\boldsymbol{V}$ satisfy the condition of the oblique projection

$$\boldsymbol{U} = \boldsymbol{A}\boldsymbol{V}. \tag{18}$$

Hence we can obtain the search subspace and the constraint subspace without the need for additional SpMV computations, and only some storage space is needed. At the end of the truncated PCG iterations, the proposed strategy executes an additional projection, which takes $\boldsymbol{V}_{\text{prev}}$ and $\boldsymbol{U}_{\text{prev}}$ as the search subspace and the constraint subspace, where $\boldsymbol{V}_{\text{prev}}$ and $\boldsymbol{U}_{\text{prev}}$ are created in solving the previous linear system, and the approximate solution can be represented as

$$\tilde{\boldsymbol{w}} = \boldsymbol{w}_m + \boldsymbol{V}_{\text{prev}}\left(\boldsymbol{U}_{\text{prev}}^T\boldsymbol{U}_{\text{prev}}\right)^{-1}\boldsymbol{U}_{\text{prev}}^T\boldsymbol{r}_m. \tag{19}$$

This implies that, throughout the entire LOBPCG algorithm, the first call of the preconditioner $\boldsymbol{T}$ does not involve the projection. Algorithm 2 provides a pseudocode of the projection strategy for the preconditioner of LOBPCG, which corresponds to the process outlined in Line 6 of Algorithm 1.

3.2. Discussion of the projection strategy. It is known that the PCG method itself belongs to the category of subspace-based iterative methods, which aims to find an optimal solution within the constructed Krylov subspace. The search subspace $\operatorname{span}\{\boldsymbol{V}\}$ we construct is equivalent to the Krylov subspace implicitly formed by the truncated PCG algorithm. Consequently, executing an additional projection, which takes $\boldsymbol{V}$ and $\boldsymbol{U}$ as the search subspace and the constraint subspace in truncated PCG algorithm, would not lead to enhanced efficiency.

In contrast, executing an additional projection using $\boldsymbol{V}_{\text{prev}}$ and $\boldsymbol{U}_{\text{prev}}$ can be considered as expanding the search subspace, thereby accelerating the convergence of the truncated PCG solver.

Erhel and Frédéric (1997) demonstrate the effectiveness of recycling Krylov subspaces constructed in solving the previous linear system to accelerate the convergence in solving the subsequent linear system, even if the two right-hand sides are unrelated. Geng and Sun (2023) show that projection strategies can effectively improve the performance and stability of the preconditioners. These two works demonstrate the efficacy of the proposed projection strategy. In our method, the Krylov subspace constructed in solving the previous linear system is recycled to accelerate the convergence of the subsequent linear system through a single oblique projection.

Through (19), we have found a better approximate solution with smaller residuals in the augmented affine subspace to replace the original one, accelerating the convergence of the truncated PCG solver, thereby enhancing the performance of the preconditioner.

The projection strategy can find a more accurate approximate solution, the one that minimizes the 2-norm residuals in the search subspace.

Alternatively, the projection process can also be understood through a low-rank system. Equation (19) can be rewritten as

$$\begin{aligned}\tilde{\boldsymbol{w}} &= \boldsymbol{w}_m \\ &\quad + \boldsymbol{V}_{\text{prev}}\left(\boldsymbol{V}_{\text{prev}}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{V}_{\text{prev}}\right)^{-1}\boldsymbol{V}_{\text{prev}}^T\boldsymbol{A}^T\boldsymbol{r}_m, \\ &= \boldsymbol{E}\boldsymbol{A}^T\boldsymbol{r}_m,\end{aligned} \tag{20}$$

where $\boldsymbol{E} = \boldsymbol{V}_{\text{prev}}\left(\boldsymbol{V}_{\text{prev}}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{V}_{\text{prev}}\right)^{-1}\boldsymbol{V}_{\text{prev}}^T$

When $\boldsymbol{V}_{\text{prev}}$ is full rank, that is, $\boldsymbol{V}_{\text{prev}} \in \mathbb{R}^{n\times n}$ is a

square matrix and invertible, then we can obtain

$$\begin{aligned} \boldsymbol{E} &= \boldsymbol{V}_{\text{prev}} \boldsymbol{V}_{\text{prev}}^{-1} \left(\boldsymbol{A}^T \boldsymbol{A}\right)^{-1} \boldsymbol{V}_{\text{prev}}^{-T} \boldsymbol{V}_{\text{prev}}^T \\ &= \left(\boldsymbol{A}^T \boldsymbol{A}\right)^{-1}. \end{aligned} \tag{21}$$

In this case, the solution obtained from (20) is exact. In practical scenarios, the rank of $\boldsymbol{V}_{\text{prev}}$ is much smaller than n, under which circumstances $\boldsymbol{E}$ degenerates into a low-rank approximation (Il'in, 2019) of the inverse matrix $(\boldsymbol{A}^T\boldsymbol{A})^{-1}$.

Through the projection step, a better approximation that minimizes the 2-norm of residuals in the search subspace is obtained. The oblique projection process eliminates the component of the residual on the subspace $\text{span}\{\boldsymbol{V}_{\text{prev}}\}$. It is obvious that $\|\tilde{\boldsymbol{r}}\|_2 \leq \|\boldsymbol{r}_m\|_2$, indicating an improvement in the preconditioner $\boldsymbol{T}$.

The proposed projection strategy is a promising way to improve the preconditioner $\boldsymbol{T}$, and most importantly, it could achieve better performance with a reduced computational overhead.

3.3. Computational overhead. Let us now evaluate the additional computational overhead required by the proposed algorithm. Equation (19) is computed in the following manner. First, calculate

$$\boldsymbol{l}_1 = \boldsymbol{U}_{\text{prev}}^T \boldsymbol{r}_0, \tag{22}$$

where $\boldsymbol{l}_1$ is an m-dimensional vector. This step requires m inner products of n-dimensional vectors. Then, calculate

$$\boldsymbol{N}_1 = \boldsymbol{U}_{\text{prev}}^T \boldsymbol{U}_{\text{prev}}, \tag{23}$$

where $\boldsymbol{N}_1$ is an m-dimensional square dense matrix. Considering the symmetry of $\boldsymbol{U}_{\text{prev}}^T \boldsymbol{U}_{\text{prev}}$, this step requires $m(m+1)/2$ inner products of n-dimensional vectors. In the subsequent calculation,

$$\boldsymbol{l}_2 = {\boldsymbol{N}_1}^{-1} \boldsymbol{l}_1, \tag{24}$$

where $\boldsymbol{l}_2$ is an m-dimensional vector, and we should use a pseudo-inverse method for inversion for the sake of stability. This part of the calculation is of order m and can be neglected as $m \ll n$. Finally, calculate

$$\tilde{\boldsymbol{w}} = \boldsymbol{w}_m + \boldsymbol{V}_{\text{prev}} \boldsymbol{l}_2 \tag{25}$$

and perform m times n-dimensional vector additions with coefficients are required.

Based on the above analysis, the additional computational efforts required by the projection strategy mainly consist of $m(m+3)/2$ inner products of n-dimensional vectors and m times n-dimensional vector additions with coefficients. Compared with the extra computational efforts by increasing the iterations of the PCG, the advantage of the projection strategy lies in eliminating the need for SpMV. Although the theoretical time complexity of SpMV is $\boldsymbol{O}(nnz)$ where nnz represents the number of non-zero elements of the matrix, factors like element distribution may lead to longer computational times. This is one of the reasons why the projection strategy makes the algorithm more efficient. Overall, the proposed technique based on the projection strategy is a promising approach to enhance the performance of the preconditioner.

4. Numerical test

In this section, the effectiveness of the proposed strategy is shown through a numerical test. Our projection strategy on stable version of the LOBPCG is implemented based on the PETSc (Balay *et al.*, 2001; 2022) and SLEPc (Hernandez *et al.*, 2005; 2003) library. The convergence boundaries of all eigenvectors are taken as relative errors, which satisfy

$$\begin{aligned} e_r &= \frac{\|\boldsymbol{A}\boldsymbol{x}_i - \lambda_i \boldsymbol{B}\boldsymbol{x}_i\|}{\|\boldsymbol{A}\boldsymbol{x}_i\|} \\ &\approx \frac{\|\boldsymbol{A}\boldsymbol{x}_i - \lambda_i \boldsymbol{B}\boldsymbol{x}_i\|}{\lambda_i \|\boldsymbol{B}\boldsymbol{x}_i\|} < 1 \times 10^{-3}. \end{aligned}$$

The matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ used in the experiments are obtained from the SuiteSparse Matrix Collection (Kolodziej *et al.*, 2019) and those used as $\boldsymbol{A}$ are all symmetric positive definite matrices, the basic information of these matrices is shown in Table 1. For each matrix, n represents the number of unknowns of the matrix (the dimension of the matrix) and nnz represents the number of non-zero elements of the matrix. Table 1 includes several stiffness matrices, such as `nd24k`, `bcsstk17`, and `bcsstk35`. For cases where the corresponding mass matrix is available, we set $\boldsymbol{B}$ to the associated mass matrix. For example, in the case of `bcsstk35/bcsstm35`, we take $\boldsymbol{A}$ as `bcsstk35` and $\boldsymbol{B}$ as `bcsstm35`. However, for stiffness matrices without a corresponding mass matrix or matrices from other domains, for simplicity, we set $\boldsymbol{B} = \boldsymbol{I}$.

All algorithms solve for the first 15 minimum eigenvalues of $\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x}$ and their corresponding eigenvectors, and we take 10 as the block size of the LOBPCG algorithm. The solution from the previous LOBPCG iteration (outer iteration) is used as the initial guess for the inner iteration. Specifically, in Line 6 of Algorithm 1, when solving $\boldsymbol{A}\boldsymbol{W} = \boldsymbol{Q}$, the initial value for $\boldsymbol{W}$ is set to the value of $\boldsymbol{W}$ from the previous iteration. This approach aligns with the implementations of the LOBPCG in both BLOPEX (Knyazev *et al.*, 2007) and SLEPc (Roman *et al.*, 2016). The experiment environment is a personal computer with an AMD Ryzen 5950X CPU, maximum available memory of 128 GB, Ubuntu 20.04 operating system, PETSc and SLEPc

Table 1. Source information of the tested matrices.

Matrix	Problem	*nnz*	*n*
1138_bus	power network	4054	1138
bcsstk38	structural	355460	8032
bcsstm38	structural	10485	8032
bcsstk17	structural	428650	10974
bcsstk37	structural	1140977	25503
bcsstm37	structural	15525	25503
bcsstk35	structural	1450163	30237
bcsstm35	structural	20619	30237
bcsstk39	structural	2060662	46772
bcsstm39	structural	46772	46772
nd24k	2D/3D	28715634	72000
apache1	structural	542184	80800
thermal1	thermal	574458	82654
2cubes_sphere	electromagnetics	1647264	101492
boneS01	model reduction	6715152	127224
bmwcra_1	Structural	10641602	148770
G2_circuit	circuit simulation	726674	150102
apache2	structural	4817870	715176
tmt_sym	electromagnetics	5080961	726713
ecology2	2D/3D	4995991	999999
thermal2	thermal	8580313	1228045
Serena	structural	64531701	1391349
Geo_1438	structural	63156690	1437960
Hook_1498	structural	60917445	1498023
G3_circuit	circuit simulation	7660826	1585478

version 3.20.1, and the 'type' of matrix in PETSc is MATAIJ.

Tables 2 and 3 display the statistics for the original LOBPCG and the LOBPCG applying projection strategy, where the truncated PCG solver employs ILU(1) as the preconditioner. Here, we set the maximum number of iterations for the PCG to 10, which is also the optimal number of iterations for numerical performance when using ILU(1)-PCG. The first row of each matrix is the result of the original LOBPCG, the second row is the results of the LOBPCG with projection strategy, respectively. In the table, 'iterations' is the number of iterations corresponding to the algorithms. Meanwhile, 'time' is the iteration time in seconds.

From Tables 2 and 3, it is revealed that the LOBPCG applying projection strategy exhibits pronounced higher solving efficiency compared with the original LOBPCG. The projection strategy not only improved computational efficiency but also enabled convergence for some cases that were originally non-convergent. For instance, the matrix boneS01, which failed to converge with the original LOBPCG even after 5000 iterations, it converged in just 531 steps after applying the projection strategy.

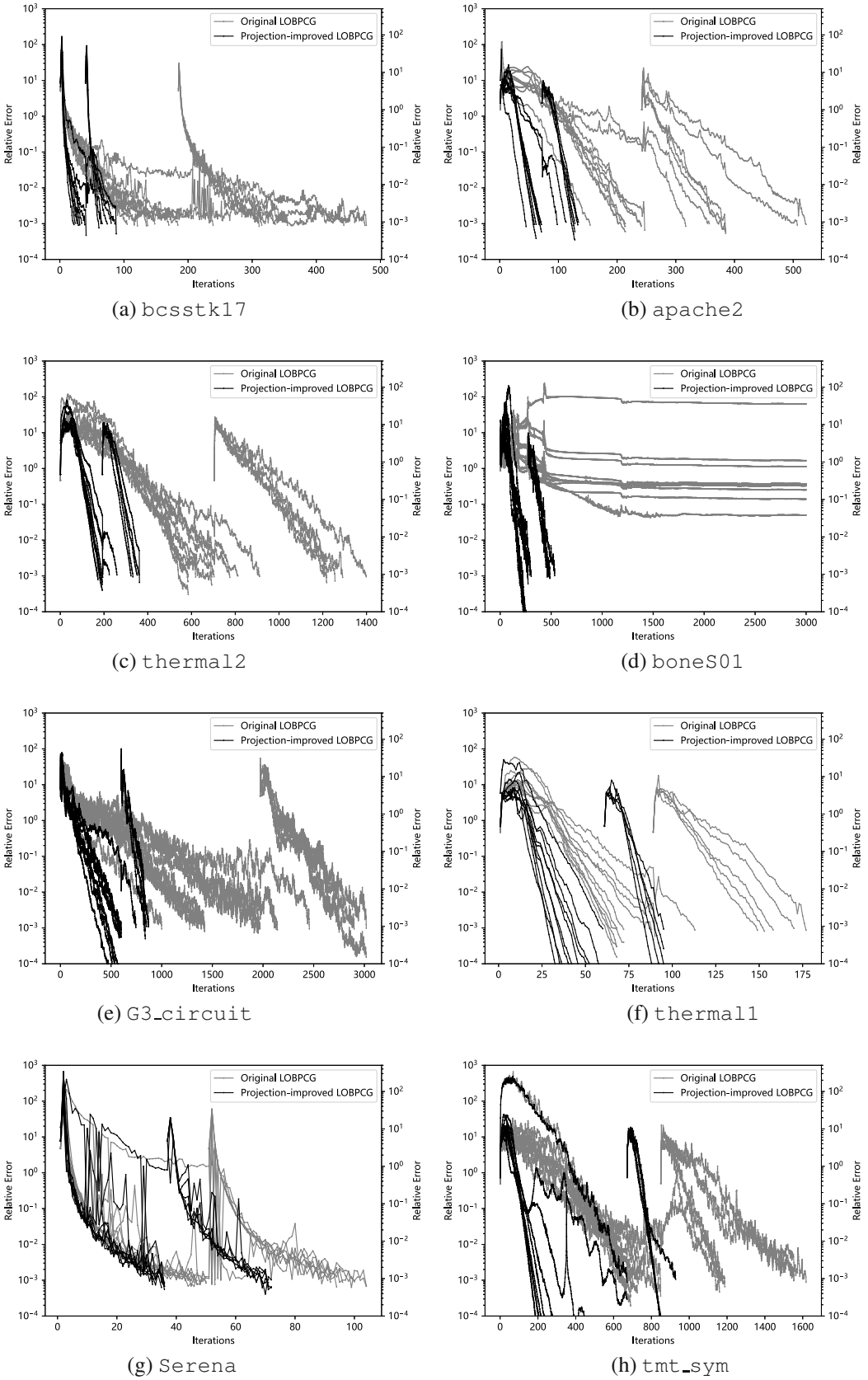
The projection strategy is effective not only when the ILU is employed as the preconditioner in the truncated PCG solver. Tables 4 and 5 display the result for the original LOBPCG and the LOBPCG applying projection strategy where the truncated PCG solver employs SPAI as the preconditioner. Clearly, the projection strategy is also highly effective with the SPAI preconditioner applied in the truncated PCG solver, significantly enhancing computational efficiency and robustness. Figures 1 and 2 show the convergence process of the eigenvectors of some matrices, indicating that the projection strategy could reduce oscillations in the convergence curve and make it smoother, greatly decreasing the number of iterations required for convergence. Since the projection strategy does not markedly increase the extra computational time per iteration step, the residual curves depicted in Figs. 1 and 2 can also demonstrate the enhanced time efficiency achieved by the improved algorithm.

5. Conclusions

We have proposed a projection strategy for improving the preconditioner $\boldsymbol{T}$ in the LOBPCG. The projection strategy begins by utilizing intermediate vectors from the truncated PCG iterations to construct search subspaces and constraint subspaces for oblique projection. Then it executes the oblique projection in the truncated PCG when solving the 'inner' linear systems. This oblique projection technique can find a more accurate approximate solution which minimizes the 2-norm residuals in the search subspace without markedly increasing the computational cost, thereby improving the quality of the preconditioner, thus accelerating the convergence of the LOBPCG.

The proposed algorithm was implemented and tested numerically based on PETSc and SLEPc with the SuiteSparse Matrix Collection, and the matrices originate from different fields of application. The projection improved preconditioner was compared with the original preconditioner, and our results show that for most cases the original LOBPCG converges slowly or even diverges, but the projection strategy can significantly accelerate the iterative process to reach the convergence condition. The projection improved preconditioner for LOBPCG demonstrates notable enhancements in both stability and efficiency.

However, despite all testing results of numerical experiments reflecting the effectiveness of the projection



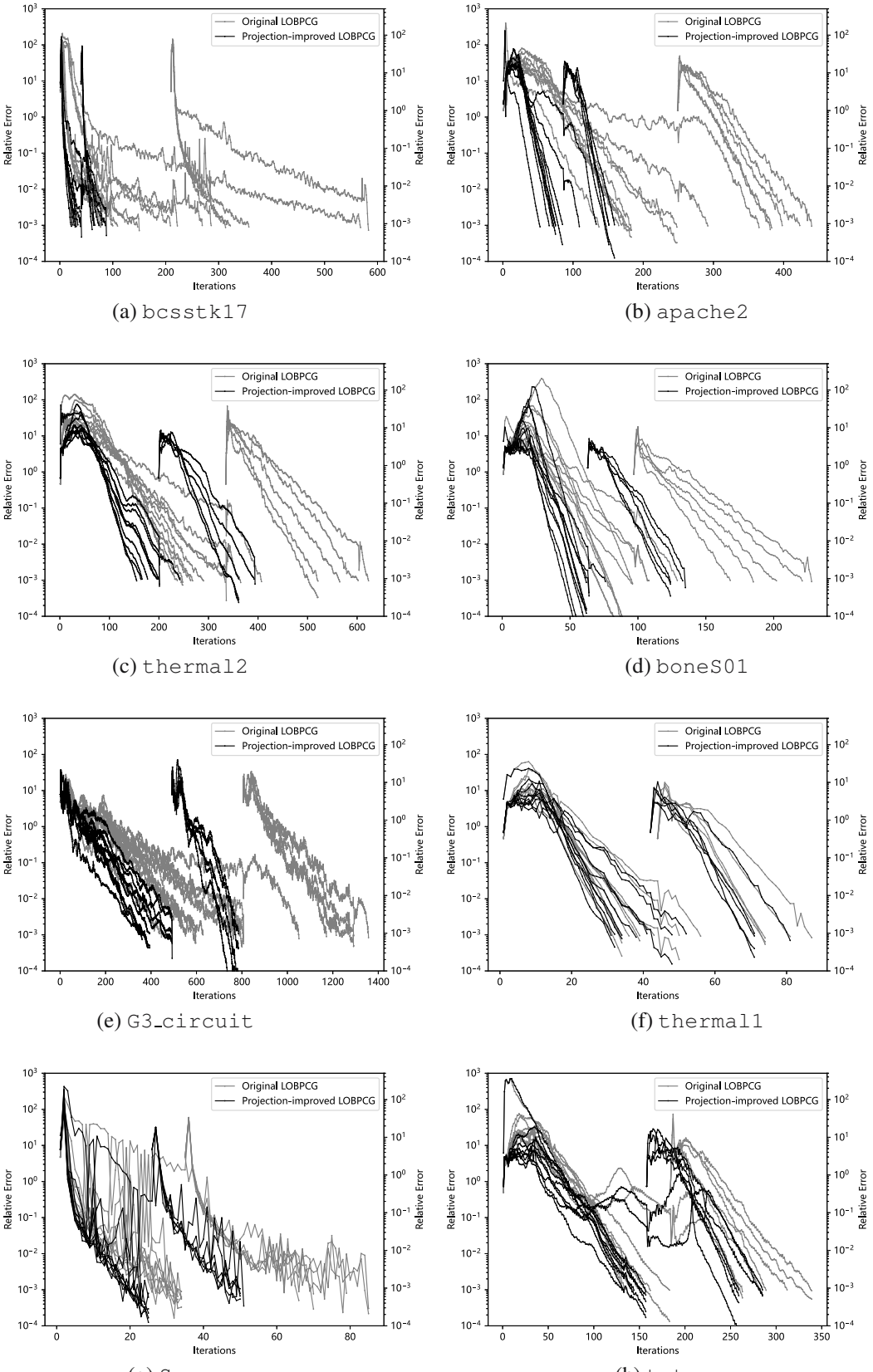
(a) bcsstk17 (b) apache2

(c) thermal2 (d) boneS01

(e) G3_circuit (f) thermal1

(g) Serena (h) tmt_sym

Fig. 1. Convergence comparison between the original LOBPCG and the LOBPCG applying projection strategy where the truncated PCG solver employs ILU(1) as the preconditioner.



(a) bcsstk17 (b) apache2

(c) thermal2 (d) boneS01

(e) G3_circuit (f) thermal1

(g) Serena (h) tmt_sym

Fig. 2. Convergence comparison between the original LOBPCG and the LOBPCG applying projection strategy where the truncated PCG solver employs SPAI(1) as the preconditioner.

Table 2. Comparison between the original LOBPCG and the LOBPCG applying the projection strategy, where the PCG solver employs ILU(1) as the preconditioner in cases taking $\boldsymbol{B} = \boldsymbol{I}$.

matrix	iter/time (original)	iter/time (improved)
1138_bus	22 / 1.28	21 / 1.24
bcsstk17	442 / 4.49	111 / 1.30
nd24k	5000[1]/3162.88	5000[1]/ 3181.40
apache1	81 / 6.81	39 / 3.83
thermal1	171 / 17.14	92 / 12.46
2cubes_sphere	91 / 11.22	70 / 6.02
boneS01	5000[1]/ 762.33	531 / 91.58
bmwcra_1	4091 / 2007.53	2549 / 1259.29
G2_circuit	320 / 59.24	123 / 30.10
apache2	478 / 456.59	139 / 159.90
tmt_sym	1618 / 1790.88	928 / 1092.77
ecology2	1073 / 1913.70	319 / 651.49
thermal2	1394 / 4097.97	417 / 1308.33
Serena	104 / 530.59	72 / 378.61
Geo_1438	212 / 749.69	83 / 429.7
Hook_1498	1465 / 8024.54	773 / 4710.73
G3_circuit	3016 / 9846.06	867 / 4242.39

[1] The solver did not converge within a limited number of iterations.

Table 3. Comparison between the original LOBPCG and the LOBPCG applying the projection strategy, where the PCG solver employs ILU(1) as the preconditioner in cases taking $\boldsymbol{B}$ as the corresponding mass matrices.

matrix	iter/time (original)	iter/time (improved)
bcsstk38/bcsstm38	20000[1]/ 134.78	20000[1]/ 144.05
bcsstk37/bcsstm37	20000[1]/ 302.22	20000[1]/ 321.13
bcsstk35/bcsstm35	20000[1]/ 318.24	20000[1]/ 331.23
bcsstk39/bcsstm39	19337 / 477.53	8588 / 332.05

[1] The solver did not converge within a limited number of iterations.

Table 4. Comparison between the original LOBPCG and the LOBPCG applying the projection strategy, where the PCG solver employs SPAI(1) as the preconditioner in cases taking $\boldsymbol{B} = \boldsymbol{I}$.

matrix	iter/time (original)	iter/time (improved)
1138_bus	36 / 0.8	32 / 0.79
bcsstk17	578 / 9.39	96 / 2.46
nd24k	934 / 802.06	280 / 264.31
apache1	5000[1]/ 192.27	68 / 3.35
thermal1	82 / 3.94336	76 / 3.75
2cubes_sphere	90 / 4.34	90 / 4.74
boneS01	213 / 51.87	121 / 36.55
bmwcra_1	5000[1]/ 4727.52	5000[1]/ 5477.05
G2_circuit	138 / 17.11	93 / 14.21
apache2	437 / 275.59	152 / 118.28
tmt_sym	339 / 284.71	285 / 251.75
ecology2	396 / 483.39	356 / 339.45
thermal2	618 / 1098.42	402 / 845.15
Serena	85 / 385.25	51 / 256.82
Geo_1438	5000[1]/ 20872.31	818 / 3592.38
Hook_1498	560 / 1905.44	446 / 1655.55
G3_circuit	1357 / 2736.69	785 / 1712.91

[1] The solver did not converge within a limited number of iterations.

Table 5. Comparison between the original LOBPCG and the LOBPCG applying the projection strategy, where the PCG solver employs SPAI(1) as the preconditioner in the cases taking $\boldsymbol{B}$ as the corresponding mass matrices.

matrix	iter/time (original)	iter/time (improved)
bcsstk38/bcsstm38	20000[1]/ 165.67	1667 / 15.55
bcsstk37/bcsstm37	462 / 13.42	428 / 11.55
bcsstk35/bcsstm35	1443 / 41.98	1270 / 34.88
bcsstk39/bcsstm39	736 / 19.58	524 / 13.88

[1] The solver did not converge within a limited number of iterations.

strategy, a rigorous theoretical proof to support it is still missing. We can only claim that the projection strategy, by reducing the 2-norm of the residuals of the linear system, improves the performance of the preconditioner $\boldsymbol{T}$, thereby accelerating convergence. In addition to using traditional methods, approaches based on machine learning can also be combined to further accelerate the eigenvalue solving algorithms, such as in (Li *et al.*, 2023), where a GNN network was used to improve the preconditioner and the initial solution of the linear system. An algorithm for determining eigenvalues could also be integrated with this approach to further enhance the adaptability and robustness of the projection strategy in the future.

Acknowledgment

This work is supported by the National Key R&D Program of China (grant no. 2021YFB2401700) and the National Natural Science Foundation of China (grant no. 11672362).

References

Balay, S., Abhyankar, S., Adams, M., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. et al. (2022). PETSc users manual, https://petsc.org/release/docs/manual.

Balay, S., Buschelman, K., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Smith, B.F. and Zhang, H. (2001). PETSc, http://www.mcs.anl.gov/petsc.

Bathe, K.-J. and Wilson, E.L. (1973). Solution methods for eigenvalue problems in structural mechanics, *International Journal for Numerical Methods in Engineering* **6**(2): 213–226.

Bennighof, J.K. and Lehoucq, R.B. (2004). An automated multilevel substructuring method for eigenspace computation in linear elastodynamics, *SIAM Journal on Scientific Computing* **25**(6): 2084–2106, DOI: 10.1137/S1064827502400650.

Collignon, T. and Gijzen, M.V. (2010). Two implementations of the preconditioned conjugate gradient method on heterogeneous computing grids, *International Journal of Applied Mathematics and Computer Science* **20**(1): 109–121, DOI: 10.2478/v10006-010-0008-4.

Duersch, J.A., Shao, M., Yang, C. and Gu, M. (2018). A robust and efficient implementation of LOBPCG, *SIAM Journal on Scientific Computing* **40**(5): C655–C676, DOI: 10.1137/17M1129830.

Erhel, J. and Frédéric, G. (1997). *An Augmented Subspace Conjugate Gradient*, PhD thesis, INRIA, Rennes.

Fan, X., Chen, P., Wu, R. and Xiao, S. (2014). Parallel computing study for the large-scale generalized eigenvalue problems in modal analysis, *Science China Physics, Mechanics and Astronomy* **57**(3): 477–489.

Feng, Y. and Owen, D. (1996). Conjugate gradient methods for solving the smallest eigenpair of large symmetric eigenvalue problems, *International Journal for Numerical Methods in Engineering* **39**(13): 2209–2229.

Geng, M. and Sun, S. (2023). Projection improved SPAI preconditioner for FGMRES, *Numerical Mathematics: Theory, Methods and Applications* **16**(4): 1035–1052.

Guarracino, M., Perla, F. and Zanetti, P. (2006). A parallel block Lanczos algorithm and its implementation for the evaluation of some eigenvalues of large sparse symmetric matrices on multicomputers, *International Journal of Applied Mathematics and Computer Science* **16**(2): 241–249.

Hernandez, V., Roman, J.E. and Vidal, V. (2003). SLEPc: Scalable Library for Eigenvalue Problem Computations, *Lecture Notes in Computer Science* **2565**: 377–391.

Hernandez, V., Roman, J.E. and Vidal, V. (2005). SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems, *ACM Transactions on Mathematical Software (TOMS)* **31**(3): 351–362.

Hetmaniuk, U. and Lehoucq, R. (2006). Basis selection in LOBPCG, *Journal of Computational Physics* **218**(1): 324–332.

Il'in, V. (2019). Projection methods in Krylov subspaces, *Journal of Mathematical Sciences* **240**(6): 772–782.

Knyazev, A.V. (2001). Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method, *SIAM Journal on Scientific Computing* **23**(2): 517–541.

Knyazev, A.V., Argentati, M.E., Lashuk, I. and Ovtchinnikov, E.E. (2007). Block locally optimal preconditioned eigenvalue xolvers (BLOPEX) in Hypre and PETSc, *SIAM Journal on Scientific Computing* **29**(5): 2224–2239.

Kolodziej, S.P., Aznaveh, M., Bullock, M., David, J., Davis, T.A., Henderson, M., Hu, Y. and Sandstrom, R. (2019). The suitesparse matrix collection website interface, *Journal of Open Source Software* **4**(35): 1244.

Kressner, D., Ma, Y. and Shao, M. (2023). A mixed precision LOBPCG algorithm, *Numerical Algorithms* **94**(4): 1653–1671, DOI: 10.1007/s11075-023-01550-9.

Lanczos, C. (1950). An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, *Journal of Research of the National Bureau of Standards* **45**(4): 255—282.

Li, Y., Chen, P.Y., Du, T. and Matusik, W. (2023). Learning preconditioners for conjugate gradient PDE solvers, *International Conference on Machine Learning, Honolulu, USA,* pp. 19425–19439.

Roman, J.E., Campos, C., Romero, E. and Tomás, A. (2016). SLEPc users manual, *Departamento di Sistemas Informàticos y Computación, Universitat Politècnica de València, TR DSIC-II/24/02, Rev* **3**.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, USA.

Sleijpen, G.L. and Van der Vorst, H.A. (2000). A Jacobi–Davidson iteration method for linear eigenvalue problems, *SIAM Review* **42**(2): 267–293.

Stathopoulos, A. and McCombs, J.R. (2010). PRIMME: PReconditioned Iterative MultiMethod Eigensolver: Methods and software description, *ACM Transactions on Mathematical Software* **37**(2): 21:1–21:30.

Sulaiman, I.M., Kaelo, P., Khalid, R. and Nawawi, M.K.M. (2024). A descent generalized RMIL spectral gradient algorithm for optimization problems, *International Journal of Applied Mathematics and Computer Science* **34**(2): 225–233, DOI: 10.61822/amcs-2024-0016.

Wu, L., Romero, E. and Stathopoulos, A. (2017). Primme_svds: A high-performance preconditioned SVD solver for accurate large-scale computations, *SIAM Journal on Scientific Computing* **39**(5): S248–S271, DOI: 10.1137/16M1082214.

Yin, J., Voss, H. and Chen, P. (2013). Improving eigenpairs of automated multilevel substructuring with subspace iterations, *Computers & Structures* **119**(1): 115–124.

Yuan, M., Chen, P., Xiong, S., Li, Y. and Wilson, E.L. (1989). The WYD method in large eigenvalue problems, *Engineering computations* **6**(1): 49–57.

Yuan, Y., Sun, S., Chen, P. and Yuan, M. (2021). Adaptive relaxation strategy on basic iterative methods for solving linear systems with single and multiple right-hand sides, *Advances in Applied Mathematics and Mechanics* **13**(2): 378–403.

Tailai Ma received his BS degree in engineering structure analysis from Peking University, China, in 2020. He is currently working toward a PhD degree in engineering mechanics in the College of Engineering at Peking University, China. His research interests include parallel computations in numerical algebra, GPU programming, parallel computations for engineering software.

Shuli Sun is currently an associate professor in the Department of Mechanics and Engineering Science at the College of Engineering, Peking University, China. He also serves as a researcher at the Nanchang Key Laboratory of Industrial Software Research and Application of Peking University Nanchang Innovation Institute, and a member of the Structural Dynamics Committee of the Chinese Society of Vibration Engineering. He received his BS degree in theoretical and applied mechanics from Peking University in 1990 and a PhD degree in computational mechanics from Peking University in 1997. His research interests include efficient algorithms in computational mechanics, modeling and analysis of complex structures, optimization of mesh quality, mesh deformation and dynamic grid techniques.

Fangyi Zheng received his BS degree in engineering structure analysis in 2015 and his PhD degree in engineering mechanics in 2023, both from Peking University, Beijing, China. His research interests include computational mechanics, applied mathematics and machine learning.

Pu Chen is a professor in the College of Engineering at Peking University, China. He is a senior member of the Chinese Society of Theoretical and Applied Mechanics and Chinese Society of Vibration Engineering. He received his PhD from the Technical University of Darmstadt in 1993, and his MS in Peking University in 1984. His research interests include parallel computation on desk machines, parallel computation for engineering software, structural reanalysis, structural safety for civil engineering, vibration analysis.

Received: 27 July 2024
Revised: 20 November 2024
Re-revised: 26 November 2024
Accepted: 16 December 2024