SIAM J. SCI. COMPUT.
Vol. 37, No. 6, pp. A2969–A2997

PRECONDITIONED EIGENSOLVERS FOR LARGE-SCALE NONLINEAR HERMITIAN EIGENPROBLEMS WITH VARIATIONAL CHARACTERIZATIONS. II. INTERIOR EIGENVALUES*

DANIEL B. SZYLD†, EUGENE VECHARYNSKI‡, AND FEI XUE§

Abstract. We consider the solution of large-scale nonlinear algebraic Hermitian eigenproblems of the form $T(\lambda)v = 0$ that admit a variational characterization of eigenvalues. These problems arise in a variety of applications and are generalizations of linear Hermitian eigenproblems $Av = \lambda Bv$. In this paper, we propose a preconditioned locally minimal residual (PLMR) method for efficiently computing interior eigenvalues of problems of this type. We discuss the development of search subspaces, preconditioning, and eigenpair extraction procedures based on the refined Rayleigh–Ritz projection. Extension to the block methods is presented, and a moving-window-style soft deflation is described. Numerical experiments demonstrate that PLMR methods provide a rapid and robust convergence toward interior eigenvalues. The approach is also shown to be efficient and reliable for computing a large number of extreme eigenvalues, dramatically outperforming standard preconditioned conjugate gradient methods.

Key words. preconditioned eigensolver, interior eigenvalues, nonlinear Hermitian eigenproblem, variational principle, PLMR, refined projection

AMS subject classifications. 65F15, 65F50, 15A18, 15A22

DOI. 10.1137/15M1016096

1. Introduction.

Nonlinear Hermitian algebraic eigenproblems of the form $T(\lambda)v = 0$ arise naturally in a variety of scientific and engineering applications. Many of these problems allow for a variational characterization (min-max principle) of some eigenvalues on certain intervals. Desirable properties of these eigenvalues and associated eigenvectors can be derived, and special methods can be developed to compute them efficiently. In part I of this study [36], we investigated preconditioned conjugate gradient (PCG) methods for computing extreme eigenvalues of the nonlinear Hermitian eigenproblem that satisfy a variational principle. In this paper, to continue our study, we develop and explore preconditioned locally minimal residual (PLMR) methods for computing interior eigenvalues. Our exploration was motivated by a new class of preconditioned eigensolvers for linear eigenproblems [37], [39].

Interior eigenvalues are intrinsically more difficult to compute than extreme eigenvalues. This is the case even for linear Hermitian problems $Av = \lambda Bv$. To devise an efficient and reliable interior eigenvalue solver, several issues need to be addressed. First, a good preconditioner M approximating $A - \sigma B$ must be available, where σ is a real shift close to the desired eigenvalues. Second, appropriate variants of subspace projection and eigenpair extraction should be used to provide a rapid and robust con-

*Submitted to the journal's Methods and Algorithms for Scientific Computing section April 9, 2015; accepted for publication (in revised form) September 21, 2015; published electronically December 17, 2015. This work was supported by the National Science Foundation under grants DMS-1115520, DMS-1418882, and DMS-1419100.

http://www.siam.org/journals/sisc/37-6/M101609.html

†Department of Mathematics, Temple University, Philadelphia, PA 19122-6094 (szyld@temple.edu).

‡Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (evecharynski@lbl.gov).

§Department of Mathematics, University of Louisiana at Lafayette, Lafayette, LA 70504-1010 (fxue@louisiana.edu).

vergence toward interior eigenvalues. In particular, the extraction should identify and discard spurious Ritz values. Exactly the same types of challenges arise when solving nonlinear interior eigenproblems.

For the first issue, we note that efficient and robust preconditioners for interior eigenvalue computations are typically constructed with indefinite matrices. Their development is rather challenging in general and is out of the scope of this paper. Nevertheless, several options are readily available, e.g., the incomplete $\mathrm{LDL}^{\mathrm{T}}$ factorization [12], [30], the absolute value preconditioning [38], or any iterative solver for a corresponding indefinite linear system. Here, we assume that a suitable preconditioner is at hand and focus on the development of search subspaces and mechanisms for extracting approximate eigenpairs.

The search subspaces suggested within the proposed PLMR method are given by certain preconditioned Krylov subspaces that are augmented with a search direction connecting the eigenvector approximations obtained in two consecutive iterations. Therefore, they can be viewed as a natural generalization of the PCG search subspaces [36]. The main difference with PCG is that the PLMR search subspace is based on an augmented Krylov subspace of a larger dimension, which enhances the robustness of convergence. Nevertheless, if a good preconditioner is available, the PLMR subspace does not have to be significantly larger than that of PCG. Therefore, similar to PCG, the eigenpair approximations in PLMR are normally extracted from subspaces of a small size.

To extract interior eigenpairs of linear eigenproblems, the harmonic Rayleigh–Ritz procedure [25], [32] is widely used. The same idea can be applied in the nonlinear case, which leads to the projected nonlinear eigenproblem $U^*T(\sigma)^*T(\nu)Uy = 0$, where U contains basis vectors of the search subspace. Then the harmonic Ritz pairs (ν, Uy) with ν close to σ provide approximations to the desired interior eigenpairs of the original problem.

The main disadvantage of the harmonic Rayleigh–Ritz projection is that it does not preserve symmetry; i.e., the projected problem $U^*T(\sigma)^*T(\nu)Uy = 0$ is no longer Hermitian. The loss of symmetry is unlikely to be a major issue for linear problems. However, in the nonlinear case, solving projected eigenproblems that do not preserve the original structure could cause considerable complications. In particular, it may require special treatment of invariant pairs [7], [35] and is likely to incur significant loss of accuracy in the final solutions.

We avoid this issue by using the standard Rayleigh–Ritz procedure. The resulting projected eigenproblem $U^*T(\nu)Uy = 0$ is also Hermitian, with eigenvalues satisfying the variational principle. To remedy the slow convergence toward interior eigenvalues, which is commonly intrinsic to the standard Rayleigh–Ritz approach, we propose a simple strategy for discarding spurious Ritz values, followed by an eigenvector refinement procedure (see [18] for linear eigenproblems) that stabilizes and accelerates the convergence. Our experience shows that such a refined projection outperforms the harmonic Rayleigh–Ritz approach and is crucial for maintaining robust convergence. We observe that the effects of the eigenvector refinement are significantly more pronounced in the nonlinear setting.

To understand the local convergence of the new method, we shall discuss a close connection between the PLMR search subspace and that of the basic Jacobi–Davidson (JD) method using the right-preconditioned GMRES as a solver for the correction equation. This connection allows derivation of the order of local convergence of PLMR, established under an assumption on the approximation property of the refined Rayleigh–Ritz procedure. Our analysis shows that PLMR with a search subspace of a

fixed size converges linearly, and it exhibits a higher order of convergence if the search subspace is expanded with every new iteration.

For the case where several eigenpairs are wanted, we present a block variant of the PLMR method, called BPLMR. To the best of our knowledge, BPLMR is the first block variant of a preconditioned eigensolver for computing interior eigenvalues of nonlinear eigenproblems. In this algorithm, special care is taken to ensure the robustness of the eigenvector refinement procedure, which is enhanced to avoid repeated convergence of semisimple and clustered eigenvalues. Moreover, special attention is devoted to computing a large number of successive eigenvalues, for which a moving-window-style soft deflation strategy is described.

The proposed PLMR methods use several well-established techniques that contribute to fast and robust convergence toward interior eigenvalues. They share similarities with the nonlinear Arnoldi method [40] as both are "preconditioned eigensolvers" based on projections onto the Krylov-like search subspaces constructed by a preconditioned linear operator. They also possess features of the nonlinear JD method [6], [42] in the use of stabilized preconditioners. Consequently, we expect that PLMR performs at least as well as nonlinear Arnoldi and JD. In addition, the suggested eigenvector refinement procedure further improves the convergence, especially if the preconditioner is not very strong, or if clustered or semisimple eigenvalues are desired.

The paper is organized as follows. Section 2 reviews the basics of nonlinear Hermitian eigenproblems, including a nonlinear variational principle. In section 3, we propose a basic PLMR method for computing one interior eigenvalue of $T(\lambda)v = 0$ around a given shift. Section 4 provides some insight into the connection between the search subspaces of PLMR and of the right-preconditioned GMRES used as an inner solver for a basic JD method, leading to a local convergence result for PLMR. In section 5, we develop BPLMR for computing several interior eigenvalues simultaneously. Numerical results, which demonstrate the efficiency of PLMR methods, are presented in section 6. Our conclusions can be found in section 7.

2. Nonlinear Hermitian eigenproblem and variational principle. In this section, we describe nonlinear algebraic Hermitian eigenproblems $T(\lambda)v = 0$ that admit a variational characterization on an open interval J. Here, $T(\cdot) : J \subset \mathbb{R} \to \mathbb{C}^{n\times n}$ maps a real scalar $\mu \in J$ continuously to the Hermitian matrix $T(\mu)$. The scalar $\lambda \in J$, for which $T(\lambda)$ is singular, is an eigenvalue of $T(\cdot)$ with a corresponding eigenvector $v \in \operatorname{null} T(\lambda) \setminus \{0\}$.

To simplify our analysis, we assume, as in part I [36], that $T(\cdot)$ does not have infinite eigenvalues on J. This assumption is valid for most Hermitian eigenproblems encountered in practice. Under the assumption, $J = (a, b)$ containing all eigenvalues of interest is finite, where a and b are not eigenvalues of $T(\cdot)$. In certain circumstances, $T(\cdot)$ does have infinite eigenvalues (for instance, linear Hermitian eigenproblems $Av = \lambda Bv$ with a semidefinite B), but those eigenvalues generally have little physical relevance and thus are rarely desired.

We start the description with several definitions.

DEFINITION 2.1. *The* Rayleigh functional $\rho(\cdot) : D \to J$ *is a continuous mapping of a vector* $x \in D \subset \mathbb{C}^n \setminus \{0\}$ *to the unique solution* $\rho(x) \in J$ *of the equation* $x^*T(\rho(x))x = 0$.

DEFINITION 2.2. *Given* $T(\cdot) : J \subset \mathbb{R} \to \mathbb{C}^{n\times n}$, $J \subset \mathbb{R}$ *is called an interval of* positive or negative type *if* $(\mu - \rho(x))(x^*T(\mu)x)$ *is constantly positive or constantly negative, respectively, for all* $x \in D$ *and all* $\mu \in J$, $\mu \neq \rho(x)$. *Both positive and negative type are* definite type.

DEFINITION 2.3. *A real scalar λ is* the kth eigenvalue *of $T(\cdot)$ if zero is the kth largest eigenvalue of the matrix $T(\lambda)$. Unless noted otherwise, the kth eigenvalue is denoted as λ_k.*

Necessary and sufficient conditions for J to be of definite type are given as follows.

PROPOSITION 2.4 (see [36, Proposition 2.4]). *Let $J = (a, b) \subset \mathbb{R}$ be finite, where a, b are not eigenvalues of $T(\cdot)$, and let $\rho : D \to J$ be the Rayleigh functional, where $D = \mathbb{C}^n \setminus \{0\}$. Then J is an interval of positive (negative) type if and only if $T(a)$ is negative (positive) definite and $T(b)$ is positive (negative) definite. Assume that $T(\cdot)$ is continuously differentiable. Then J is of positive (negative) type if $x^*T'(\rho(x))x > 0$ (< 0) for all $x \in D$. If, in addition, $T(\cdot)$ is twice continuously differentiable and $x^*T''(\rho(x))x \neq 0$ for all $x \in D$, then J is of positive (negative) type if and only if $x^*T'(\rho(x))x > 0$ (< 0) for all $x \in D$.*

On an interval of definite type, we have a variational characterization of eigenvalues of $T(\cdot)$ and the orthogonality of eigenvectors [11], [43], [46].

THEOREM 2.5 (nonlinear variational principle). *Let $J \subset \mathbb{R}$ be finite and of definite type, and let $T(\cdot)$ be continuously differentiable on J. Assume that for all $x \in D \subset \mathbb{C}^n \setminus \{0\}$ there exists exactly one Rayleigh functional value $\rho(x) \in J$ such that $x^*T(\rho(x))x = 0$. Then there exist exactly n eigenvalues $\{\lambda_k\}_{k=1}^n$ of $T(\cdot)$ on J that satisfy a variational principle. Specifically, if J is of positive type, then*

$$\begin{aligned} \lambda_k &= \min\{\max\{\rho(x) \,|\, x \in S, x \neq 0\} \,|\, \dim(S) = k\} \quad \text{and} \\ \lambda_k &= \max\{\min\{\rho(x) \,|\, x \in S, x \neq 0\} \,|\, \dim(S) = n - k + 1\}; \end{aligned} \tag{2.1}$$

if J is of negative type, then

$$\begin{aligned} \lambda_k &= \max\{\min\{\rho(x) \,|\, x \in S, x \neq 0\} \,|\, \dim(S) = k\} \quad \text{and} \\ \lambda_k &= \min\{\max\{\rho(x) \,|\, x \in S, x \neq 0\} \,|\, \dim(S) = n - k + 1\}. \end{aligned} \tag{2.2}$$

Moreover, there exist n corresponding eigenvectors $\{v_k\}_{k=1}^n$ that form a basis of $\mathbb{C}^n$, and they are orthogonal with respect to the scalar-valued function $[\cdot, \cdot]$ defined as

$$[x, y] = \begin{cases} y^*\big(T(\rho(y)) - T(\rho(x))\big)x / \left(\rho(y) - \rho(x)\right) & \text{if } \rho(x) \neq \rho(y), \\ y^*T'(\rho(x))x & \text{if } \rho(x) = \rho(y). \end{cases} \tag{2.3}$$

A natural corollary of the nonlinear variational principle (2.1) or (2.2) is the nonlinear Cauchy interlacing theorem.

THEOREM 2.6 (nonlinear Cauchy interlacing theorem [36]). *Let $J = (a, b)$ be finite and of definite type, let $T(\cdot)$ be continuously differentiable on J, and let $U \in \mathbb{C}^{n \times m}$ contain m linearly independent column vectors. Then the projected eigenproblem $U^*T(\nu)Uy = 0$ has exactly m eigenpairs $\{(\nu_j, y_j)\}_{j=1}^m$ satisfying the nonlinear variational principle* (2.1) or (2.2)*. In addition, if J is of positive type, then $\lambda_j \leq \nu_j \leq \lambda_{n-m+j}$; if J is of negative type, then $\lambda_{n-m+j} \leq \nu_j \leq \lambda_j$ $(1 \leq j \leq m)$.*

From Definition 2.3 and the nonlinear variational principle (2.1) or (2.2), we see that the eigenvalues of the nonlinear Hermitian eigenproblem on an interval of definite type can be ordered in the same manner as for linear Hermitian eigenproblems. In particular, this ordering is needed when PLMR is used for computing many successive extreme eigenvalues, as described in section 5.5.

3. The single-vector PLMR. In this section, we present a basic version of the PLMR method for computing an interior eigenvalue and the associated eigenvector of the nonlinear Hermitian eigenproblem. We discuss the main building blocks

of the method, including development of the search subspace, preconditioning, and extraction of the approximate eigenpair.

3.1. Development of the search subspace. Assume that λ is a unique distinct eigenvalue of $T(\cdot)$ closest to $\sigma \in J$, and let v be a corresponding eigenvector. Suppose that in the kth iteration we have an approximate eigenvector x_k. Our goal here is to develop a search subspace $\mathcal{U}_k$, from which a more accurate eigenvector approximation can be extracted.

We start by reviewing the search subspace constructed within a variant of the PCG method, the locally optimal preconditioned conjugate gradient (LOPCG) algorithm [36], for computing the lowest eigenvalue of $T(\cdot)$. Given the current eigenvector approximation x_k, LOPCG defines the search subspace as

$$\mathcal{U}_k^{\mathrm{LOPCG}} = \mathrm{span}\{x_k, M^{-1}\nabla\rho(x_k), p_{k-1}\},$$

where

$$\nabla\rho(x_k) = -\frac{2}{x^*T'\left(\rho(x_k)\right)x_k} T\left(\rho(x_k)\right)x_k$$

is the gradient of the Rayleigh functional $\rho(\cdot)$ at x_k (see [36, Proposition 3.1]), $M \approx T(\sigma)$ is a preconditioner, and p_{k-1} is the search direction connecting x_{k-1} and x_k ($p_{-1} = 0$). Note that $T\left(\rho(x_k)\right)x_k$ defines the residual of the eigenproblem, which is parallel to the gradient $\nabla\rho(x_k)$.

For the sake of simplicity, we let $\rho_k = \rho(x_k)$ when there is no danger of confusion. The above search subspace can then be written as

$$\mathcal{U}_k^{\mathrm{LOPCG}} = \mathcal{K}_2\left(M^{-1}T(\rho_k), x_k\right) + \mathrm{span}\{p_{k-1}\}, \tag{3.1}$$

where $\mathcal{K}_m(A, b) = \mathrm{span}\left\{b, Ab, \ldots, A^{m-1}b\right\}$ denotes an m-dimensional Krylov subspace [29]. This three-dimensional search subspace, unfortunately, is not effective for computing interior eigenvalues. The main issue is that the convergence toward these eigenvalues can be fairly slow, and thus a search subspace of a larger dimension is needed to stabilize and accelerate the convergence. This is especially evident if the preconditioner M is not very strong.

In order to properly enlarge the LOPCG subspace (3.1), we consider the subspace

$$\mathcal{U}_k = \mathcal{K}_{m+1}(M^{-1}T(\rho_k), x_k) + \mathrm{span}\{p_{k-1}\}, \tag{3.2}$$

where $\mathcal{K}_{m+1}(M^{-1}T(\rho_k), x_k)$ generalizes the limit space $\mathcal{K}_{m+1}\left(M^{-1}(A - \rho_k B), x_k\right)$ of the generalized Davidson method that restarts every m steps [27]. The augmentation of this Krylov subspace with the search direction p_{k-1} is expected to accelerate the convergence as it does for PCG methods. Similar ideas have been proposed in the literature, e.g., [22], [23], [24].

3.2. Stabilization of preconditioning. A drawback of the search subspace (3.2) is that it can potentially suffer from numerical instabilities. To see this, let $M = T(\sigma)$ and $\rho_k = \sigma$. Then $M^{-1}T(\rho_k) = I$, and the search subspace degenerates to $\mathrm{span}\{x_k, p_{k-1}\}$. Therefore, an algorithm based on the search subspace (3.2) stagnates, i.e., cannot generate any improvement in eigenvector approximation. In practice, stagnation could arise whenever M is a good approximation to $T(\sigma)$ and ρ_k is sufficiently close to σ. The same issue is known for the Davidson-type methods for linear eigenproblems, which has been fixed by the JD algorithm [8], [32].

A key ingredient contributing to the robustness of the JD methods is the modification of the preconditioning procedure in such a way that it is performed through solution of a correction equation rather than a direct application of M^{-1}. In particular, for nonlinear eigenproblems, the correction equation of the basic JD method[1] is of the form

$$\Pi_1 T(\rho_k)\Pi_2 \Delta x_k = -T(\rho_k)x_k, \tag{3.3}$$

where Π_1 and Π_2 are properly chosen projectors, such that $T(\rho_k)x_k \in \text{range}(\Pi_1)$ and $\Pi_2 \Delta x_k = \Delta x_k$; see, e.g., [6], [31, Chapter 6.2], [34]. For Hermitian $T(\cdot)$, one can choose

$$\Pi_1 = I - \frac{T'(\rho_k)x_k x_k^*}{x_k^* T'(\rho_k)x_k} \quad \text{and} \quad \Pi_2 = \Pi_1^* = I - \frac{x_k x_k^* T'(\rho_k)}{x_k^* T'(\rho_k)x_k}. \tag{3.4}$$

In this case, the coefficient matrix in (3.3) is Hermitian, and thus efficient preconditioned linear solvers such as MINRES [28] or SQMR [10] can be applied. It can be shown that the exact solution Δx_k of (3.3) with the projectors defined in (3.4) satisfies

$$x_k + \Delta x_k = \frac{x_k^* T'(\rho_k)x_k}{x_k^* T'(\rho_k)T(\rho_k)^{-1}T'(\rho_k)x_k} T(\rho_k)^{-1}T'(\rho_k)x_k,$$

which is parallel to the new iterate $x_{k+1} = T(\rho_k)^{-1}T'(\rho_k)x_k$ obtained from the Rayleigh functional iteration; see [31, Chapter 4.3], [34].

Motivated by the structure of the coefficient matrix in the correction equation (3.3), we modify the preconditioner $M \approx T(\sigma)$ by multiplying it with the projectors in (3.4). This replaces M with a stabilized preconditioner

$$M_\Pi = \Pi M \Pi^*, \tag{3.5}$$

where $\Pi = \Pi_1$, as defined in (3.4). Thus, the preconditioner is now applied to a vector through $M_\Pi^\dagger \equiv \Pi M^{-1}\Pi^*$ rather than M^{-1}.

The precise formula that describes the action of $M_\Pi^\dagger$ on a given vector can be derived as follows. We consider vectors y and b such that $b = M_\Pi y = \Pi M \Pi^* y$, where $b \in \text{range}(\Pi)$, and $y \perp T'(\rho_k)x_k$, i.e., $\Pi^* y = y$. These assumptions are standard in the preconditioning for JD, and identical to those used in [34], [45]. Equivalently,

$$b = \left(I - \frac{T'(\rho_k)x_k x_k^*}{x_k^* T'(\rho_k)x_k}\right) My = My - T'(\rho_k)x_k \frac{x_k^* My}{x_k^* T'(\rho_k)x_k},$$

and it follows that

$$y = M^{-1}b + M^{-1}T'(\rho_k)x_k \frac{x_k^* My}{x_k^* T'(\rho_k)x_k}. \tag{3.6}$$

The orthogonality $y \perp T'(\rho_k)x_k$ implies that

$$0 = x_k^* T'(\rho_k)y = x_k^* T'(\rho_k)M^{-1}b + x_k^* T'(\rho_k)M^{-1}T'(\rho_k)x_k \frac{x_k^* My}{x_k^* T'(\rho_k)x_k},$$

[1] The basic variant of JD forms the new approximation as $x_{k+1} = x_k + \Delta x_k$, where Δx_k is an approximate solution to the correction equation; no subspace expansion and projection is involved. It is also referred to as single-vector JD or simplified JD in the literature; see, e.g., [9], [13].

from which we have

$$\frac{x_k^* M y}{x_k^* T'(\rho_k) x_k} = -\frac{x_k^* T'(\rho_k) M^{-1} b}{x_k^* T'(\rho_k) M^{-1} T'(\rho_k) x_k}. \tag{3.7}$$

Thus, after substituting (3.7) into (3.6), we obtain

$$\begin{aligned} y = M_\Pi^\dagger b &= M^{-1} b - \frac{x_k^* T'(\rho_k) M^{-1} b}{x_k^* T'(\rho_k) M^{-1} T'(\rho_k) x_k} M^{-1} T'(\rho_k) x_k \\ &= \left(I - \frac{M^{-1} T'(\rho_k) x_k \, x_k^* T'(\rho_k)}{x_k^* T'(\rho_k) M^{-1} T'(\rho_k) x_k} \right) M^{-1} b. \end{aligned} \tag{3.8}$$

In contrast to M^{-1}, the operator $M_\Pi^\dagger$ does not cancel out with the matrix $T(\rho_k)$ and, instead, applies M^{-1} to $T'(\rho_k)x_k$, magnifying the desired eigenvector component. Note that $M^{-1}T'(\rho_k)x_k$ in (3.8) can be computed only once and further used to evaluate vectors of the form $M_\Pi^\dagger T(\rho_k) z$, for any $T(\rho_k)z \in \text{range}(\Pi)$. This observation is needed for the construction of the PLMR search subspace.

Finally, given a stabilized preconditioner (3.5), whose action on a vector is expressed in (3.8), we define the PLMR search subspace as

$$\mathcal{U}_k^{\text{PLMR}(m)} = \mathcal{K}_{m+1}(M_\Pi^\dagger T(\rho_k), x_k) + \text{span}\{p_{k-1}\}, \tag{3.9}$$

which is exactly (3.2) with M replaced by M_Π. Our numerical experience confirms that the PLMR version built upon (3.9) indeed tends to be significantly more robust than that based on (3.2). Therefore, throughout we only use (3.9), constructed with the stabilized preconditioner, as the search subspace for the PLMR algorithm.

3.3. Subspace projection and extraction. Given the PLMR search subspace (3.9), we now consider the projection of the original eigenproblem onto this subspace and describe the extraction of a new eigenvector approximation.

Similar to the linear setting, the standard Rayleigh–Ritz procedure is ideal in preserving symmetry and is most suitable for computing extreme eigenvalues, but it generally exhibits very slow convergence toward interior eigenvalues. As a remedy, the harmonic Rayleigh–Ritz scheme could be used. However, the main disadvantage of this approach is that it does not preserve symmetry of the original eigenproblem. In general, algorithms for solving interior eigenvalues of nonlinear eigenproblems without symmetry are significantly more complicated and tend to be less robust than those for solving nonlinear Hermitian eigenproblems admitting a variational principle; see, e.g., [44], [45], and references therein. In addition, solving nonlinear projected eigenproblems that fail to preserve the structure (non-Hermitian in our case) can lead to a significant loss of accuracy in the final eigenpair approximations.

To resolve this issue, we propose using the standard Rayleigh–Ritz projection, followed by a procedure to detect and discard spurious Ritz values, and a refinement step to further improve the quality of eigenvector approximation. Note that spurious Ritz values are Ritz values close to the desired shift σ, but they correspond to poor eigenvector approximations, typically given by linear combinations of eigenvectors associated with eigenvalues outside the interval of interest. They arise frequently when interior eigenvalues are sought.

Specifically, let $U_k \in \mathbb{C}^{n\times(m+2)}$ contain orthonormal basis vectors of the search subspace (3.9). The Rayleigh–Ritz scheme then leads to the projected Hermitian eigenproblem

$$U_k^* T(\nu) U_k y = 0, \tag{3.10}$$

which also admits a variational characterization of its eigenvalues (i.e., of the Ritz values) satisfying the nonlinear Cauchy interlacing theorem (Theorem 2.6). The projected eigenproblem can be solved, for example, by PCG [36] or other specially designed methods that exploit the underlying structure. We shall discuss the details in section 6.

After forming the projected problem (3.10), the following approach is used to obtain an approximate eigenpair. First, we solve (3.10) for the r successive Ritz values $\nu_1, \ldots, \nu_r$ closest to σ and the associated eigenvectors $y_1, \ldots, y_r$ and compute the corresponding Ritz vectors $z_i = U_k y_i$ $(1 \le i \le r)$. We then order $\{\nu_i\}$ according to the residual norms of the respective Ritz pairs, such that for any i, j with $1 \le i < j \le r$,

$$\frac{\left\|T\left(\nu_i\right) z_i\right\|_2}{\left\|T\left(\nu_i\right)\right\|_F \left\|z_i\right\|_2} \le \frac{\left\|T\left(\nu_j\right) z_j\right\|_2}{\left\|T\left(\nu_j\right)\right\|_F \left\|z_j\right\|_2}. \tag{3.11}$$

Next, we take s Ritz pairs (ν_i, z_i) of minimal eigenresidual norm from the r candidates and choose the Ritz value, say ν_ℓ $(1 \le \ell \le s)$, that is closest to σ.

The motivation of the above approach is well founded. We first find a relatively large number, r, of Ritz values near σ, so that a good eigenvalue approximation is included in this set. Other Ritz values, not selected, are relatively far from σ and cannot represent accurate approximations to the desired eigenvalue. The ordering of Ritz pairs in terms of eigenresidual norm tends to put promising Ritz pairs to the front of the ordered set and others to the end. This step aims to filter out spurious Ritz pairs, i.e., those with Ritz values close to σ but with large eigenresidual norms. Such pairs commonly arise in the Rayleigh–Ritz projection for computing interior eigenvalues and are excluded from further consideration due to the proposed ordering. As a result, we have a fairly good chance that a promising Ritz pair is included in the set of s candidates with minimal eigenresidual norm. Finally, the Ritz value ν_ℓ closest to σ is selected from the s candidates. In our implementation, by default, $r = \min(m+1, \max(5, \lceil (m+1)/2 \rceil))$ and $s = 2$.

By construction, the selected interior Ritz pair (ν_ℓ, z_ℓ) has a reasonably small eigenresidual. However, in most cases, it can be further significantly improved. To this end, we refine the Ritz vector by substituting it with a new eigenvector approximation that delivers a minimal residual. That is, we solve

$$y_{MR} = \operatorname{argmin}_{\|y\|=1} \|T(\nu_\ell) U_k y\|_2 \tag{3.12}$$

and set the new iterate to $x_{k+1} = U_k y_{MR}$. The corresponding eigenvalue approximation is then given by the Rayleigh functional ρ_{k+1} evaluated at x_{k+1}.

Problem (3.12) can be approached by finding the smallest singular value of the matrix $T(\nu_\ell) U_k \in \mathbb{C}^{n \times (m+2)}$ and its right singular vector, or equivalently by solving the linear eigenproblem $U_k^* T(\nu_\ell)^* T(\nu_\ell) U_k y = \eta y$ for the eigenvector corresponding to the smallest eigenvalue. Note that the entire strategy described above is a direct generalization of the refinement procedure of [18] to the case of nonlinear eigenproblems. As we shall see in section 6, step (3.12) indeed turns out to be crucial for stabilizing the convergence of PLMR. The whole PLMR scheme is summarized in Algorithm 1.

4. Local convergence analysis. In this section, we provide an analysis of the PLMR algorithm, explaining the conditions that guarantee its convergence and how rapidly it may converge. Our analysis is based on a close connection between the search subspaces developed by PLMR and the basic JD method and on certain assumptions about the performance of the subspace projection and extraction.

Algorithm 1: The PLMR(m) algorithm for a Hermitian eigenproblem $T(\lambda)v = 0$.

Input: An initial eigenvector approximation $x_0 \in \mathbb{C}^n \setminus \{0\}$, a preconditioner M, a shift $\sigma \in \mathbb{R}$, and integers r, $s > 0$;
Output: An eigenpair (λ, v), where λ is the eigenvalue of $T(\cdot)$ closest to σ;

1: Set $k \leftarrow 0$ and $p_{-1} \leftarrow [\,]$. Compute $\rho_0 = \rho(x_0)$.
2: **while** convergence not reached **do**
3: If $k > 0$, then $p_{k-1} \leftarrow x_k - x_{k-1}$.
4: Compute an orthonormal basis U_k of the search subspace (3.9), where the action of the preconditioner M_Π is given by (3.8).
5: Solve the Rayleigh–Ritz projected eigenproblem (3.10).
6: Select the r Ritz values closest to σ and sort the corresponding Ritz pairs according to their eigenresidual norms (3.11). Then choose the s Ritz pairs with minimal eigenresidual and use them to identify the Ritz value ν_ℓ closest to σ.
7: Compute the right singular vector y_{MR} associated with the smallest singular value of the matrix $T(\nu_\ell)U_k$.
8: Set $x_{k+1} \leftarrow U_k y_{MR}$; $\rho_{k+1} \leftarrow \rho(x_{k+1})$.
9: Normalize x_{k+1}, such that $x_{k+1}^* T'(\rho_{k+1}) x_{k+1} = 1$.
10: $k \leftarrow k + 1$. Check convergence of (ρ_k, x_k).
11: **end while**
12: Set $\lambda \leftarrow \rho_k$; $v \leftarrow x_k$. Return (λ, v).

Specifically, let (ρ_k, x_k) be the current approximation to the desired eigenpair (λ, v), where λ is the eigenvalue of $T(\cdot)$ closest to σ. Recall from (3.3) the basic JD correction equation

$$\Pi T(\rho_k)\Pi^* \Delta x_k = -T(\rho_k)x_k$$

and assume that the preconditioner (3.5) is used for a Krylov subspace method with right-preconditioning to solve this equation. In iteration m, the Krylov subspace developed for the preconditioned linear system is thus

$$\mathcal{K}_m\left(\Pi T(\rho_k)\Pi^* M_\Pi^\dagger, T(\rho_k)x_k\right),$$

where $\Pi = I - \frac{T'(\rho_k)x_k x_k^*}{x_k^* T'(\rho_k) x_k}$. Due to the right-preconditioning, the approximate solution Δx_k of the original JD correction equation (3.3) lies in $M_\Pi^\dagger \mathcal{K}_m\big(\Pi T(\rho_k)\Pi^* M_\Pi^\dagger, T(\rho_k)x_k\big)$, and therefore the new approximation $x_{k+1} = x_k + \Delta x_k$ lies in

$$\begin{aligned}
&\operatorname{span}\{x_k\} + M_\Pi^\dagger \mathcal{K}_m\left(\Pi T(\rho_k)\Pi^* M_\Pi^\dagger, T(\rho_k)x_k\right) \\
&= \operatorname{span}\Big\{x_k, M_\Pi^\dagger T(\rho_k)x_k, M_\Pi^\dagger \Pi T(\rho_k)\Pi^* M_\Pi^\dagger T(\rho_k)x_k, \ldots, \\
&\qquad\qquad M_\Pi^\dagger\left(\Pi T(\rho_k)\Pi^* M_\Pi^\dagger\right)^{m-1} T(\rho_k)x_k\Big\} \\
&= \operatorname{span}\Big\{x_k, M_\Pi^\dagger T(\rho_k)x_k, \left(M_\Pi^\dagger T(\rho_k)\right)^2 x_k, \ldots, \left(M_\Pi^\dagger T(\rho_k)\right)^m x_k\Big\} \\
&= \mathcal{K}_{m+1}\left(M_\Pi^\dagger T(\rho_k), x_k\right),
\end{aligned} \tag{4.1}$$

where we used the identity

$$M_\Pi^\dagger \Pi = \Pi^* M_\Pi^\dagger = M_\Pi^\dagger,$$

which can be derived from (3.4) and (3.8) without much difficulty. Clearly, the subspace (4.1) is a proper subspace of the PLMR(m) search subspace (3.9), an augmented version of (4.1). This observation is summarized in the following lemma.

LEMMA 4.1. *Given the same current iterate x_k, basic* JD *with correction equation* (3.3) *delivers a new eigenvector approximation x_{k+1}^{JD} lying in the search subspace where* PLMR(m) *extracts its new iterate x_{k+1}^{PLMR}.*

Consequently, if x_{k+1}^{PLMR} is of the same quality as x_{k+1}^{JD}, then the convergence of PLMR can be established as a corollary of the local convergence theorem of basic JD, already shown in our problem setting [34, Theorems 7, 11]. Whether the new iterates of the two methods are comparable in quality depends on the approximation properties of the refined Rayleigh–Ritz projection used in PLMR, which have been established for standard linear eigenproblems; see, e.g., [33, Chapter 4.4] and references therein.

Let v be the desired eigenvector, and let U contain basis vectors for the eigensolver search subspace. Roughly speaking, under certain typically nonstringent conditions, $\angle(v, Uy)$, the angle between v and the corresponding Ritz or refined vector Uy, is proportional to $\angle(v, \mathrm{range}(U))$. For nonlinear eigenproblems $T(\lambda)v = 0$, a complete study of similar properties of these techniques is beyond the scope of this paper. Nevertheless, in our numerical experiments, we find that $\angle(v, Uy)$ is also proportional to $\angle(v, \mathrm{range}(U))$ consistently. Therefore, we assume that this property holds and give a major local convergence result of PLMR.

THEOREM 4.2. *Let (λ, v) be a simple eigenpair of the nonlinear Hermitian eigenproblem $T(\lambda)v = 0$, where v is normalized such that $v^*T'(\lambda)v = 1$. Assume that there exist a $\delta > 0$ and a corresponding $\xi > 0$, such that for any eigenpair approximation (μ, x) sufficiently close to (λ, v), namely, with $\|\left[\begin{smallmatrix} x \\ \mu \end{smallmatrix}\right] - \left[\begin{smallmatrix} v \\ \lambda \end{smallmatrix}\right]\| \le \delta$, we have $\|T'(\mu)x\| \le \xi$. Let $x_k = \gamma_k(c_k v + s_k g_k)$ be the eigenvector approximation obtained in the kth iteration of* PLMR, *where γ_k, c_k, and s_k are the generalized norm of x_k and the generalized cosine and sine of $\angle(x_k, v)$, respectively (see the appendix). Suppose that $\angle(x_0, v)$ is sufficiently small, such that $\left\|\left[\begin{smallmatrix} x_0 \\ \rho(x_0) \end{smallmatrix}\right] - \left[\begin{smallmatrix} v \\ \lambda \end{smallmatrix}\right]\right\| \le \delta$. For each x_k, assume that the refined projection extracts a new eigenvector approximation x_{k+1}, such that $\sin\angle(v, x_{k+1}) \le C \sin\angle(v, \mathcal{U}_k^{\mathrm{PLMR}})$ for a small constant C independent of k. Assume that the* JD *correction equation* (3.3) *is solved by right-preconditioned* GMRES(m_k) *with the preconditioner defined in* (3.5). *Let $\tau_k^{(\alpha)} = \tau_0^{(\alpha)}$ be a sufficiently small and fixed tolerance, and let $\tau_k^{(\beta)} \le C_\beta \frac{|s_k|}{|c_k|}$ and $\tau_k^{(\gamma)} \le C_\gamma \frac{|s_k|^2}{|c_k|^2}$ be decreasing sequences of tolerances, where C_β and C_γ are sufficiently small constants independent of k. For each k, assume that m_k is sufficiently large, such that one cycle of* GMRES(m_k) *delivers an approximate solution of* (3.3) *satisfying the relative tolerance $\tau_k^{(\alpha)}$, $\tau_k^{(\beta)}$, or $\tau_k^{(\gamma)}$, respectively. Then* PLMR(m_k) *converges toward (λ, v) at least linearly, quadratically, or cubically, respectively.*

Proof. Given the above assumptions, it is shown in Theorems 7 and 11 in [34] that the basic JD method with approximate inner linear solves that satisfy the tolerances $\tau_k^{(\alpha)}$, $\tau_k^{(\beta)}$, and $\tau_k^{(\gamma)}$, respectively, converges locally toward (λ, v) linearly, quadratically, and cubically, respectively. Note that basic JD generates the new approximation $x_{k+1}^{\mathrm{JD}} \in \mathcal{U}_k^{\mathrm{PLMR}}$ as shown in (4.1), and therefore $\sin\angle(v, x_{k+1}^{\mathrm{JD}}) \ge \sin\angle(v, \mathcal{U}_k^{\mathrm{PLMR}})$. By assumption, the refined projection of PLMR delivers the new eigenvector approximation x_{k+1}^{PLMR} satisfying $\sin\angle(v, x_{k+1}^{\mathrm{PLMR}}) \le C \sin\angle(v, \mathcal{U}_k^{\mathrm{PLMR}})$, and therefore $\sin\angle(v, x_{k+1}^{\mathrm{PLMR}}) \le C \sin\angle(v, x_{k+1}^{\mathrm{JD}})$. The convergence of PLMR thus follows directly from that of basic JD. ∎

5. Block PLMR. In this section, we consider simultaneous computation of a few interior eigenvalues and their eigenvectors. To this end, we develop a block variant of PLMR, referred to as BPLMR. We shall see that most of the techniques used in PLMR can be extended directly to the block case. We also discuss deflation techniques and describe their application to computing large numbers of successive eigenpairs.

5.1. Search subspace and preconditioning. Assume that we want to find the q eigenvalues closest to σ, namely, $\lambda_1, \ldots, \lambda_q$, such that $|\lambda_1 - \sigma| \le \cdots \le |\lambda_q - \sigma|$, together with the associated eigenvectors $v_1, \ldots, v_q$.[2] Let $X_k = [x_k^{(1)} \ \ldots \ x_k^{(q)}] \in \mathbb{C}^{n\times q}$ be the block of eigenvector approximations at iteration k, and let $\Phi_k = \operatorname{diag}(\rho_k^{(1)}, \ldots, \rho_k^{(q)})$ denote a diagonal matrix of the Rayleigh functional values $\rho_k^{(i)} = \rho(x_k^{(i)})$. We define the block of eigenresiduals

$$\mathbb{T}(X_k, \Phi_k) = [T(\rho_k^{(1)})x_k^{(1)} \ \ldots \ T(\rho_k^{(q)})x_k^{(q)}],$$

and hence we can construct an LOBPCG-type search subspace spanned by the columns of X_k, $\mathbb{T}(X_k, \Phi_k)$, and P_{k-1}, where P_{k-1} carries information about the approximate eigenvectors in the previous step ($P_{-1} = \mathbf{0}$); see part I of this study [36]. Similar to the single-vector case, such a LOBPCG subspace can be further expanded to better accommodate approximations of the interior eigenpairs, leading to the BPLMR search subspace

$$\mathcal{U}_k^{\mathrm{BPLMR}(m)} = \mathcal{K}_{m+1}\left(\mathbb{M}_{\mathbf{\Pi}}^{\dagger}\mathbb{T}(\,\cdot\,, \Phi_k), X_k\right) + \operatorname{range}(P_{k-1}), \tag{5.1}$$

where $\mathcal{K}_{m+1}\big(\mathbb{M}_{\mathbf{\Pi}}^{\dagger}\mathbb{T}(\,\cdot\,, \Phi_k), X_k\big)$ is the block Krylov subspace generated by the starting block X_k and the linear operator $\mathbb{L}_k(\cdot) \equiv \mathbb{M}_{\mathbf{\Pi}}^{\dagger}\mathbb{T}(\,\cdot\,, \Phi_k)$. That is,

$$\mathcal{K}_{m+1}\left(\mathbb{M}_{\mathbf{\Pi}}^{\dagger}\mathbb{T}(\,\cdot\,, \Phi_k), X_k\right) = \operatorname{range}\big\{X_k, \mathbb{L}_k(X_k), \mathbb{L}_k\left(\mathbb{L}_k(X_k)\right), \ldots, \mathbb{L}_k^m(X_k)\big\},$$

where $\mathbb{L}^m(\cdot)$ stands for the composition of $\mathbb{L}(\cdot)$ with itself for m times, and $\operatorname{range}(X)$ denotes the column space of X.

By analogy with (3.4) and (3.5), in (5.1), we introduce a stabilized preconditioner

$$\mathbb{M}_{\mathbf{\Pi}} = \mathbf{\Pi} M \mathbf{\Pi}^*, \tag{5.2}$$

where

$$\mathbf{\Pi} = I - Z_k(X_k^* Z_k)^{-1} X_k^*, \quad Z_k \equiv \mathbb{T}'(X_k, \Phi_k) = [T'(\rho_k^{(1)})x_k^{(1)} \ \ldots \ T'(\rho_k^{(q)})x_k^{(q)}].$$

The above projector $\mathbf{\Pi}$ is a direct extension of the one defined in (3.4) to the block case. Similar to (3.8), the action of $\mathbb{M}_{\mathbf{\Pi}}^{\dagger}$ on a block of vectors $B \in \operatorname{range}(\mathbb{M}_{\mathbf{\Pi}})$ can be expressed as

$$\mathbb{M}_{\mathbf{\Pi}}^{\dagger} B = \left(I - M^{-1}Z_k(Z_k^* M^{-1} Z_k)^{-1} Z_k^*\right) M^{-1} B. \tag{5.3}$$

Clearly, (5.1) represents a sum of q PLMR search subspaces (3.9) with starting vectors $x_k^{(i)}$ $(1 \le i \le q)$ and the preconditioner $\mathbb{M}_{\mathbf{\Pi}}$ in (5.2), and is of dimension $(m+2)q$ in general.

[2] To facilitate the description of interior eigenvalue computation, the numbering of eigenvalues here is different from the natural order defined in Definition 2.3.

The block search direction P_{k-1} can have several possible formulations. One option is to define $P_{k-1} = X_k - X_{k-1}$, which represents a direct generalization of the single-vector directions $p_{k-1} = x_k - x_{k-1}$ used is PLMR. An alternative formulation can be given by

$$P_{k-1} = X_k - X_{k-1} \left(X_{k-1}^* X_{k-1} \right)^{-1} X_{k-1}^* X_k, \tag{5.4}$$

which is a residual of the least squares problem $\min_{G \in \mathbb{C}^{q \times q}} \|X_k - X_{k-1} G\|_F$. Hence, definition (5.4) guarantees that P_{k-1} has the smallest norm columnwise for all blocks of the form $X_k - X_{k-1}G$, where $G \in \mathbb{C}^{q \times q}$.

In exact arithmetic, the two variants of P_{k-1} lead to the same search subspace, because

$$\begin{aligned} \mathcal{U}_k^{\mathrm{BPLMR}(m)} &= \mathrm{range}\{P_{k-1}\} + \mathcal{K}_{m+1}\left(\mathbb{M}_{\mathbf{\Pi}}^{\dagger} \mathbb{T}(\,\cdot\,, \Phi_k), X_k \right) \\ &= \mathrm{range}\{X_{k-1}\} + \mathcal{K}_{m+1}\left(\mathbb{M}_{\mathbf{\Pi}}^{\dagger} \mathbb{T}(\,\cdot\,, \Phi_k), X_k \right). \end{aligned}$$

In practice, BPLMR(m) working with either version of P_{k-1} converges equally rapidly in most cases, but formulation (5.4) tends to perform slightly better occasionally. We have no complete understanding of this, but have an intuitive explanation. The individual eigenvector approximations in X_k are usually properly ordered, e.g., by the distances between the corresponding eigenvalue approximations and σ. As the algorithm proceeds, the ordering of some eigenvector approximations could change due to the change of their eigenvalue approximations. When such a change occurs, $P_{k-1} = X_k - X_{k-1}$ generates poor search directions that represent the difference between approximations to distinct eigenvectors in two consecutive iterations. By contrast, the least squares problem finds a matrix $G = (X_{k-1}^* X_{k-1})^{-1} X_{k-1}^* X_k$ that "reorders" the columns of X_k so that $X_k G$ aligns columnwise with X_k, and thus $P_{k-1} = X_k - X_{k-1}G$ represents the difference between the *subspaces* spanned by the two block iterates, and it is more likely to be numerically favorable.

5.2. Subspace projection and extraction. The subspace projection and extraction for BPLMR also follow PLMR closely. In particular, let $U_k \in \mathbb{C}^{n \times (m+2)q}$ contain orthonormal basis vectors of (5.1). First, we use the standard Rayleigh–Ritz procedure to obtain the projected Hermitian eigenproblem $U_k^* T(\nu) U_k y = 0$, whose $(m+2)q$ eigenvalues (the Ritz values) satisfy the nonlinear variational principle (Theorem 2.5) and the nonlinear Cauchy interlacing theorem (Theorem 2.6). Next, we find the r Ritz values $\nu_1, \ldots, \nu_r$ that are closest to σ and order them according to the eigenresidual norms of the corresponding Ritz pairs, so that (3.11) holds for any $1 \le i < j \le r$. Then, given the r candidates, we choose s Ritz values ν_i, and the associated Ritz vectors $z_i = U_k y_i$, that yield the smallest eigenresiduals. Finally, out of these s Ritz pairs, we select the q Ritz values $\nu_{\ell_1}, \ldots, \nu_{\ell_q}$ that are closest to σ and further use them in the refinement procedure.

As we have already explained, the motivation for this approach is to filter out spurious Ritz values by first including all promising approximations in a relatively large set of r Ritz pairs, and then abandoning those with largest eigenresidual norm to obtain a set of s candidates. This set is used to choose the q most promising Ritz values, i.e., those closest to σ. By default, we let $r = \min\left((m+2)q, \max(3q, \lceil (m+2)q/3 \rceil)\right)$ and $s = 2q$.

Finally, given the Ritz values $\nu_{\ell_1}, \ldots, \nu_{\ell_q}$, we use them in the refinement step

$$y_{MR}^{(i)} = \mathrm{argmin}_{\|y\|=1} \left\| T(\nu_{\ell_i}) U_k y \right\|_2, \quad 1 \le i \le q. \tag{5.5}$$

Solving q minimization problems (5.5) allows defining the block $X_{k+1} = [x_{k+1}^{(1)} \ \dots \ x_{k+1}^{(q)}]$ of new approximate eigenvectors, such that $x_{k+1}^{(i)} = U_k y_{MR}^{(i)}$. The corresponding values of the Rayleigh functional are then placed on the diagonal of Φ_{k+1}.

5.3. Refined projection for semisimple and tightly clustered eigenvalues. The refined projection described in section 5.2 can generate improved approximations to individual eigenvectors if all the targeted eigenvalues $\lambda_1, \dots, \lambda_q$ are simple and well separated. However, if an eigenvalue of interest is semisimple, i.e., $\dim(\operatorname{null} T(\lambda_i)) = g > 1$ for some i, or if g eigenvalues are tightly clustered, then the suggested refinement scheme has difficulties computing the entire invariant subspace. In this situation, the Rayleigh–Ritz procedure generates several (up to g) Ritz values that are very close to each other. The refinement step (5.5) with these Ritz values then delivers almost identical new eigenvector approximations, which leads to an inaccurate approximation to the complete eigenspace.

In order to adapt the refined projection to the case of semisimple or tightly clustered eigenvalues, we propose the following strategy. We first select the most promising Ritz values $\nu_{\ell_1}, \dots, \nu_{\ell_q}$ and distribute them among K groups $G_1, \dots, G_K$ in such a way that all the values in one group are very close to each other, whereas those belonging to different groups are relatively well separated. Next, we presume that the Ritz values inside each group G_τ that contains multiple elements converge to a numerically semisimple eigenvalue, so that each G_τ aims at revealing a distinct semisimple eigenvalue. In this case, instead of computing individual refined eigenvector approximations for the Ritz values in G_τ using (5.5), we extract an orthonormal basis that approximates the entire eigenspace associated with the targeted semisimple eigenvalue. This is achieved by utilizing singular vectors corresponding to several smallest singular values of the reduced matrices $T(\nu)U_k$, where ν is a representative value for the Ritz values in the given group.

More precisely, we take G_1 as an example and assume without loss of generality that it contains $g > 1$ tightly clustered Ritz values, i.e., $G_1 = \{\nu_{\ell_1}, \dots, \nu_{\ell_g}\}$ for some $1 \le g \le q$. We then find the right singular vectors $y_1, \dots, y_g$ corresponding to the g smallest singular values of the matrix $T(\nu_{\ell_1})U_k$ and define the new iterates as $x_{k+1}^{(i)} = U_k y_i$, where $1 \le i \le g$. The constructed vectors $x_{k+1}^{(i)}$ deliver an orthonormal basis that is expected to approximate the eigenspace of a semisimple eigenvalue $\lambda \approx \nu_{\ell_1} \approx \cdots \approx \nu_{\ell_g}$. Note that $y_1, \dots, y_g$ can also be the singular vectors of any of the matrices $T(\nu_{\ell_i})U_k$, since the values ν_{ℓ_i} are very close to each other by construction $(1 \le i \le g)$.

In order to assign the Ritz values to the groups $G_1, \dots, G_K$, an appropriate threshold needs to be chosen to determine if several values are sufficiently close to be included into one group. An excessively small threshold could mistakenly treat a semisimple eigenvalue as several well-separated simple eigenvalues and thus encounter the difficulty described above (failure to generate the complete eigenspace accurately), whereas an overly large threshold could incorrectly treat several distinct simple eigenvalues as a semisimple one, resulting in inaccurate eigenvector approximations. For example, in our BPLMR implementation, the Ritz values $\nu_{\ell_1}, \dots, \nu_{\ell_g}$ are assigned to the same group if

$$\max_{i=1,\dots,g} \frac{|\nu_{\ell_i} - \bar{\nu}|}{|\bar{\nu}|} \le 10^{-8}, \quad \text{where } \bar{\nu} = \frac{\sum_{i=1}^{g} \nu_{\ell_i}}{g}. \tag{5.6}$$

Whenever available, a priori information on distribution of the desired eigenvalues can be exploited for a more flexible threshold estimation.

It is clear that, in practice, the Ritz values of group G_τ can converge to multiple tightly clustered eigenvalues, which contradicts our assumption on the convergence to a single semisimple eigenvalue. Nevertheless, the assumption turns out to be nonrestrictive. In fact, a group of tightly clustered eigenvalues can be considered as those arising from a small perturbation imposed on a semisimple eigenvalue. Consequently, the invariant subspace associated with this group comes from a slight perturbation of the eigenspace corresponding to this semisimple eigenvalue. In this case, it is not necessary, and in fact impractical, to compute each individual eigenvector to very high accuracy. The orthonormal basis obtained from our proposed approach forms a good approximation to the eigenspace corresponding to the presumably semisimple eigenvalue, and thus it provides a good approximation to the invariant subspace for the clustered eigenvalues. If there is need to resolve each individual eigenpair in this clustered group to higher accuracy, we can set the tolerance described in (5.6) moderately smaller. However, our experience indicates that an excessively small tolerance tends to delay the convergence if the desired eigenvalue is indeed semisimple.

We note that additional care needs to be taken in the refinement step to avoid repeated convergence. This is because such a refinement procedure is constructed independently for each numerically distinct Ritz value, and thus the singular vectors coming from two different residual minimization problems (5.5) tend to be numerically linearly dependent whenever two selected Ritz values ν_{ℓ_i} and ν_{ℓ_j} are close but not sufficiently close to be distributed into one group. To tackle this difficulty, for each candidate new eigenvector approximation $x_{k+1}^{(i)} = U_k y_{MR}^{(i)}$ $(1 \le i \le q)$, we check if $\angle(x_{k+1}^{(i)}, \mathcal{X})$ is greater than some threshold, where $\mathcal{X}$ stands for the space spanned by all previously selected new eigenvector approximations $x_{k+1}^{(1)}, \ldots, x_{k+1}^{(i-1)}$. We accept such a candidate if this criterion is satisfied; otherwise, we choose the singular vector associated with the next smallest singular value and test this condition again, until a linearly independent new eigenvector approximation $x_{k+1}^{(i)}$ is found.

5.4. Deflation. Deflation plays a crucial role in simultaneous calculation of several eigenpairs. It allows eigensolvers to exclude the converged quantities from the computation and update only unconverged eigenvector approximations. It also ensures that no repeated convergence occurs. For linear eigenproblems, deflation is based on the eigendecomposition (Hermitian case) or the Schur form (non-Hermitian case) and is usually fulfilled by orthogonalizing the search subspace against the converged invariant subspace. Such a deflation mechanism is often called "hard deflation" (or "hard locking"), as the converged eigenvectors are not explicitly included into the search subspace. For nonlinear eigenproblems $T(\lambda)v = 0$, deflation is performed by working with invariant pairs directly using special variants of Newton-like methods [5], [7], [21], or using the infinite Arnoldi method that allows for a Schur form on a transformed linear space [17], [15].

For nonlinear Hermitian eigenproblems $T(\lambda)v = 0$ that satisfy the variational principle (Theorem 2.5), deflation can be performed without explicitly preserving invariant pairs, since all eigenvectors are linearly independent. One would naturally wonder if hard deflation is possible, e.g., through orthogonalization based on the scalar-valued function $[\cdot, \cdot]$ defined in (2.3). Unfortunately, this approach is not viable, as $[\cdot, \cdot]$ is not bilinear in general, and thus the Gram–Schmidt procedure does not work. Instead, we simply include the converged invariant subspace into the BPLMR search subspace generated by the unconverged eigenvectors and, after performing the refined projection, update only the unconverged pairs. This strategy is usually called "soft deflation" (or "soft locking").

Specifically, assume that the first d columns of X_k have converged. We can then distinguish between the converged and unconverged columns. The former can be placed into the matrix $X_k^{\text{conv}} = [x_k^{(1)} \ \dots \ x_k^{(d)}]$, whereas the latter are used to form the "active" block $X_k^{\text{act}} = [x_k^{(d+1)} \ \dots \ x_k^{(q)}]$. The deflated BPLMR subspace can then be defined as

$$\mathcal{U}_k^{\text{BPLMR}(m)} = \text{range}(X_k^{\text{conv}}) + \mathcal{K}_{m+1}\left(\mathbb{M}_{\mathbf{\Pi}}^{\dagger}\mathbb{T}(\cdot, \Phi_k^{\text{act}}), X_k^{\text{act}}\right) + \text{range}\left(P_{k-1}^{\text{act}}\right), \tag{5.7}$$

where $\Phi_k^{\text{act}} = \text{diag}(\rho(x_k^{(d+1)}), \dots, \rho(x_k^{(q)}))$, and the block search direction P_{k-1}^{act} is constructed according to (5.4) with X_k and X_{k-1} replaced by X_k^{act} and X_{k-1}^{act}, respectively. Here, X_{k-1}^{act} refers to the eigenvector approximations in iteration $k-1$ that correspond to the active set in the current iteration k. Similarly, the preconditioner $\mathbb{M}_{\mathbf{\Pi}}$ is constructed as in (5.2), with X_k, Φ_k, and Z_k replaced by X_k^{act}, Φ_k^{act}, and $\mathbb{T}'(X_k^{\text{act}}, \Phi_k^{\text{act}})$, respectively. Following the convention, we denote an orthonormal basis of (5.7) by U_k, which contains $d + (m+2)(q-d)$ columns.

Given U_k, we perform the Rayleigh–Ritz procedure and solve the projected eigenproblem (3.10) to obtain a set of the Ritz pairs. We then recover the d Ritz pairs that have previously converged. This is done by checking whether a Ritz pair (ν, z) has both $\min_{1\le i\le d} |\nu - \rho^{(i)}|$ and $\angle(z, \text{range}(X_k^{\text{conv}}))$ sufficiently small. Next, we apply the strategy discussed in section 5.2 to select $q-d$ promising Ritz values from the remaining $(m+2)(q-d)$ Ritz pairs, and then use the selected Ritz values as shifts for the refined projection. The refined projection should be performed as described in section 5.3 to avoid repeated convergence. The entire scheme of BPLMR with deflation is summarized in Algorithm 2.

5.5. Computing many successive eigenvalues. In this section, we discuss an extension of the use of PLMR methods for the computation of many successive eigenvalues. Such a computation is crucial in a variety of important applications, for example, where a large number of the lowest eigenvalues and corresponding eigenvectors are desired. Traditional PCG methods are generally most reliable in this setting, but they rely on the min-max property of eigenvalues and thus require complete deflation of all converged eigenvectors. Consequently, both the memory and arithmetic cost gradually become prohibitive as the number of desired eigenvalues, n_d, grows to a few hundred or more. In addition, for nonlinear Hermitian problems, the rapid increase in arithmetic cost is even more dramatic as n_d grows, because soft deflation including all converged eigenvectors is needed for the Rayleigh–Ritz projection. As a result, solving a single projected eigenproblem becomes increasingly time-consuming.

To tackle this issue, we need to perform *partial deflation*, instead of complete deflation, of converged eigenvectors. The motivation for partial deflation is that PLMR methods are designed to generate approximations to eigenvalues around the shift σ, and thus deflation of the eigenvectors associated with eigenvalues far from σ is not necessary since the algorithms would not converge to those eigenvalues anyway, provided that a good preconditioner $M \approx T(\sigma)$ is available. Consequently, only a partial deflation of eigenvectors corresponding to eigenvalues near σ is sufficient to avoid repeated convergence.

In fact, the partial deflation strategy can be easily developed based on the soft deflation we studied. Specifically, note that we can use soft deflation to avoid repeated convergence to any previously found eigenvectors, so that additional desired eigenpairs can be computed in an incremental manner. Let $W \in \mathbb{C}^{n\times\ell}$ contain ℓ converged eigenvectors already obtained. To deflate these eigenvectors, BPLMR simply develops

Algorithm 2: The BPLMR(m) algorithm for a Hermitian eigenproblem $T(\lambda)v = 0$.

Input: Initial eigenvector approximations $X_0 \in \mathbb{C}^{n\times q}$, a preconditioner M, the shift $\sigma \in \mathbb{R}$, and integers r, $s > 0$;
Output: q eigenpairs (λ_i, v_i), where λ_i's are the eigenvalues of $T(\cdot)$ closest to σ;

1: Set $X_0^{\rm act} \leftarrow X_0$, $k \leftarrow 0$, $d \leftarrow 0$, and compute $\Phi_0^{\rm act} \leftarrow {\rm diag}(\rho(x_0^{(1)}), \ldots, \rho(x_0^{(q)}))$.
2: Set $X_0^{\rm conv} \leftarrow [\,]$ and $P_{-1}^{\rm act} \leftarrow [\,]$.
3: **while** convergence not reached **do**
4: If $k > 0$, then $P_{k-1}^{\rm act} \leftarrow X_k^{\rm act} - X_{k-1}^{\rm act}\left(X_{k-1}^{act*} X_{k-1}^{\rm act}\right)^{-1} X_{k-1}^{act*} X_k^{\rm act}$.
5: Compute an orthonormal basis U_k of the search subspace (5.7), where the action of the preconditioner $\mathbb{M}_{\mathbf{\Pi}}$ is given by (5.3) with $Z_k \equiv Z_k^{\rm act} = \mathbb{T}'(X_k^{\rm act}, \Phi_k^{\rm act})$.
6: Solve the Rayleigh–Ritz projected eigenproblem (3.10) for all Ritz pairs.
7: Restore the converged d Ritz pairs, then select the r *unconverged* Ritz values closest to σ and sort the corresponding Ritz pairs according to their eigenresidual norms (3.11). Then choose the s Ritz pairs with minimal eigenresidual and take the $q - d$ Ritz values $\nu_{\ell_1}, \ldots, \nu_{\ell_{q-d}}$ closest to σ from the s candidates.
8: Distribute the $q - d$ Ritz values among K groups $G_1, \ldots, G_K$, such that (5.6) is satisfied. That is, the values in the same group are tightly clustered, and those in different groups are well separated.
9: (a) For $\tau = 1, 2, \ldots, K$, let G_τ be the current group containing $g(\tau)$ Ritz values, and let $\nu_{\ell(\tau)}$ be a Ritz value in G_τ. Find the smallest singular values of $T(\nu_{\ell(\tau)})U_k$ and associated right singular vectors y_1, y_2, ...
(b) For $i = 1, 2, \ldots, d + (m + 2)(q - d)$, compute the candidate new eigenvector approximation $U_k y_i$ and accept it only if $\angle(U_k y_i, \mathcal{X}) > \delta$, where $\mathcal{X}$ is spanned by all columns of $\mathrm{X}_{k-1}^{\rm conv}$ and all previous accepted new eigenvector approximations.
(c) Once $g(\tau)$ new eigenvector approximations are obtained for group G_τ, reorder all new eigenvector approximations such that $\left|\rho(x_{k+1}^{(d+1)}) - \sigma\right| \le \cdots \le \left|\rho(x_{k+1}^{(q)}) - \sigma\right|$. Normalize each column such that $x_{k+1}^{(i)*} T'(\rho(x_{k+1}^{(i)}))x_{k+1}^{(i)} = 1$ for all $d + 1 \le i \le q$. Move to process the next group until all K groups are processed.
10: Determine the number d of converged eigenvectors. Set $X_{k+1}^{\rm conv} \leftarrow [x_{k+1}^{(1)}, \ldots, x_{k+1}^{(d)}]$, $X_{k+1}^{\rm act} \leftarrow [x_{k+1}^{(d+1)}, \ldots, x_{k+1}^{(q)}]$, and $\Phi_{k+1}^{\rm act} \leftarrow {\rm diag}\big(\rho(x_{k+1}^{(d+1)}), \ldots, \rho(x_{k+1}^{(q)})\big)$.
11: $k \leftarrow k + 1$. If $d = q$, then declare convergence.
12: **end while**
13: Set $\lambda_i \leftarrow \rho_k^{(i)}$; $v_i \leftarrow x_k^{(i)}$. Return (λ_i, v_i) for $i = 1, \ldots, q$.

the search subspace

$$\mathrm{range}(W) + \mathrm{range}(X_k^{\rm conv}) + \mathcal{K}_{m+1}\left(\mathbb{M}_{\mathbf{\Pi}}^{\dagger}\mathbb{T}(\cdot, \Phi_k^{\rm act}), X_k^{\rm act}\right) + \mathrm{range}\left(P_{k-1}^{\rm act}\right)$$

and treats W the same way as $X_k^{\rm conv}$. Specifically, it performs the Rayleigh–Ritz projection and obtains the $\ell + d$ converged Ritz pairs. It then finds the $q - d$ most promising Ritz values from the unconverged Ritz pairs and uses them as the shifts for the refinement procedure. New eigenvector approximations are generated as usual from the singular vectors corresponding to the smallest singular values of relevant matrices, and each candidate $U_k y$ is accepted only if $\angle\big(U_k y, \mathcal{X}\big)$ is not very small, where $\mathcal{X}$ is the space spanned by the columns of W, $X_k^{\rm conv}$, and all previously selected new eigenvector approximations in iteration k.

With the above extension of soft deflation, we now propose the "moving-window"-style partial deflation for computing successive eigenvalues of $T(\cdot)$ on an interval $(a, b) \subset \mathbb{R}$. We start BPLMR with the set of converged eigenvectors $W = \emptyset$ to

compute the q eigenvalues $\lambda_{1,1}, \ldots, \lambda_{1,q}$ closest to $\sigma_1 = a$ and set the columns of W to be the corresponding eigenvectors $v_{1,1}, \ldots, v_{1,q}$. Then we choose a nearby shift $\sigma_2 > \sigma_1$ and use BPLMR with W to find the q eigenvalues $\lambda_{2,1}, \ldots, \lambda_{2,q}$ near σ_2. The two sets of eigenvalues should have no intersection due to the use of deflation. Then the new set of eigenvectors $v_{2,1}, \ldots, v_{2,q}$ are added to W, and we choose a new shift $\sigma_3 > \sigma_2$ and invoke BPLMR again. At a certain step, if the current shift σ_i is far from σ_1, for example, we remove the first set of eigenvectors $v_{1,1}, \ldots, v_{1,q}$ from W. We also update the preconditioner when necessary to maintain rapid convergence for eigenvalues near the new shift. The maximum window size, i.e., the largest number of columns of W allowed, is determined upon a trade-off between the storage cost and the occurrence of repeated convergence.

The described partial deflation strategy is critical for keeping the total computational cost roughly proportional to the total number, n_d, of desired eigenvalues. Recently, a strategy with similar motivation, called "local numbering of eigenvalues," has been successfully used with a basic nonlinear Arnoldi method for computing many successive eigenvalues [3]. As we shall see in section 6, our proposed approach is highly reliable and efficient in this problem setting.

6. Numerical experiments. We illustrate the performance of the PLMR methods on a few Hermitian eigenproblems satisfying the variational characterization (2.1) or (2.2). We shall see that the new algorithms exhibit rapid and robust convergence toward interior eigenvalues, provided that good preconditioners are available. Unless otherwise noted, the experiments were performed on a Macbook computer running Mac OS X 10.7.5, MATLAB R2012b, with a 2.4 GHz Intel Core 2 Duo processor and 4GB 667MHz DDR2 memory.

TABLE 1
Description of the test problems.

Problem	Type	Order	Interval
wiresaw	quadratic	1024	$(0, 3250)$
genhyper	quadratic	4096	$(-843, 0.3943)$
sleeper	quadratic	16384	$(-16.33, -1.61)$
string	rational	10000	$(4.4, 1.2 \times 10^9)$
pdde	nonlinear	39601	$(-20.87, 4.08)$
artificial	nonlinear	16129	$(-0.43, 3.34)$
Laplace2D	linear	10000	$(0, 8)$
Laplace3D	linear	125000	$(0, 12)$

We choose eight Hermitian eigenproblems for the test. The six nonlinear eigenproblems have been introduced in part I of our study [36], but we describe them here again to make this paper self-contained. Table 1 summarizes these problems, among which the quadratic and the rational eigenproblems are constructed from the NLEVP toolbox [4]. The first quadratic eigenproblem *wiresaw* of order 1024 comes from the vibration analysis of a wiresaw, constructed using the command `nlevp('wiresaw1',1024)`. The eigenvalues of this gyroscopic eigenproblem are purely imaginary and thus do not satisfy the variational principle (2.1) or (2.2), but they can be mapped to real eigenvalues of a transformed Hermitian eigenproblem by substituting λ with $i\lambda$. The transformed problem has 1024 pairs of real eigenvalues $\{\lambda_i^{\pm}\}$, where $\lambda_i^- = -\lambda_i^+$, and $\{\lambda_i^-\}$ and $\{\lambda_i^+\}$ lie in $I_\ell = (-3250, 0)$ and $I_r = (0, 3250)$, respectively. The variational principle (2.1) holds on I_ℓ and (2.2) on I_r, respectively, and we look for the eigenvalues on I_r. Next, the hyperbolic quadratic problem *genhyper*

of order 4096 is constructed by the command `nlevp('genhyper',ev,[eye(4096) eye(4096)])`, where `ev` is a vector whose entries are the reciprocals of 8192 random numbers generated by `randn` function initialized with a zero seed. The elements of `ev` are set to be the eigenvalues of this problem, 4096 of which are distributed on the left interval $I_\ell = (-843, 0.3943)$, and the rest lie in the right interval $I_r = (0.3943, 20061)$. The variational principle (2.1) is satisfied on I_r and (2.2) on I_ℓ, respectively, and we aim at solving the eigenvalues on I_ℓ. The third quadratic problem *sleeper* of the form $T(\lambda) = A_0 + \lambda A_1 + \lambda^2 A_2$ of order 16384 models the oscillations of a rail track lying on sleepers. The problem is constructed by the command `nlevp('sleeper',128)`, and then the matrix corresponding to the constant term is changed from A_0 to $A_0 - 2I$, so that the modified problem satisfies the variational principle (2.2) on $(-16.33, -1.61)$. The rational eigenproblem *string* of the form $T(\lambda) = A - \lambda B + \frac{\lambda}{\lambda - 1} C$ of order 10000 is generated by the command `nlevp('string',10000)`; it arises in the finite element discretization of a boundary problem describing the eigenvibration of a string attached to a spring. The variational principle (2.2) holds on the interval $(4.4, 1.2 \times 10^9)$.

Two truly nonlinear eigenproblems are described as follows. The first arises from the modeling of a partial delay differential equation (*pdde*) [14] $u_t(x,t) = \Delta u(x,t) + a(x)u(x,t) + b(x)u(x,t-2)$ defined on $\Omega = [0,\pi] \times [0,\pi]$ for $t \geq 0$, where $a(x) = 8\sin(x_1)\sin(x_2)$ and $b(x) = 100|\sin(x_1 + x_2)|$, with Dirichlet boundary condition $u(x,t) = 0$ for all $x \in \partial\Omega$ and $t \geq 0$. Assume that the solution is of the form $u(x,t) = e^{\lambda t} v(x)$. Using the standard 5-point stencil finite difference approximation to the Laplacian operator on a 200×200 uniform grid, we obtain an algebraic eigenproblem $T(\lambda) = \lambda I + (M + A) + e^{-2\lambda} B$, where the matrices M, A, and B of order 39601 are the discretized form of the Laplacian operator, $a(x)$, $b(x)$, respectively. The variational principle (2.2) holds on the interval $(-20.87, 4.08)$. The second is an artificial problem of order 16129 of the form $T(\lambda) = -\sin\frac{\lambda}{5} A + \sqrt{\lambda + 1} B + e^{-\lambda/\sqrt{\pi}} C$, where $A = I$, $B = \text{tridiag}[1; -2; 1]$, and C comes from the standard 5-point stencil finite difference discretization of the Laplacian, based on a 128×128 uniform grid on the unit square, without scaling by the mesh size factor $\frac{1}{h^2} = 128^2$ as done for the *pdde* problem. The variational principle (2.2) holds on $(-0.43, 3.34)$.

The eigenproblems *Laplace*2*D* and *Laplace*3*D* arise from the standard 5-point and 7-point stencil finite difference discretization of the Laplacian with Dirichlet boundary conditions on the unit square and unit cube, using 100×100 and $50 \times 50 \times 50$ uniform grids, respectively. Both are linear eigenproblems with the majority of eigenvalues being semisimple. Although our focus is on nonlinear problems, we include these examples to demonstrate the BPLMR's capability to resolve multiplicities. The matrices A are of order 10000 (*Laplace*2*D*) and 125000 (*Laplace*3*D*) and are generated using the MATLAB function `laplacian.m` downloaded from the MATLAB Central File Exchange, developed by A. Knyazev.

To facilitate the discussion of the performance of PLMR methods, we briefly explain the algorithms used for solving the projected Hermitian eigenproblems (3.10) that arise in each iteration of PLMR. For the linear eigenproblems *Lplace*2*D* and *Laplace*3*D*, we simply use MATLAB's `eig` function. For the quadratic eigenproblems *wiresaw*, *genhyper*, and *sleeper* (modified) formulated as $T(\lambda)v = (\lambda^2 A + \lambda B + C)v = 0$, we apply the linearization

$$\begin{bmatrix} -C & 0 \\ 0 & A \end{bmatrix} \begin{bmatrix} v \\ \lambda v \end{bmatrix} - \lambda \begin{bmatrix} B & A \\ A & 0 \end{bmatrix} \begin{bmatrix} v \\ \lambda v \end{bmatrix} = 0,$$

which is equivalent to the original problem (see, e.g., [1] and references therein), and we solve the linearized problem by `eig`. The rational problem *string* is transformed

into a quadratic eigenproblem and is then linearized and solved by `eig`. The truly nonlinear eigenproblems *pdde* and *artificial* are handled as follows. We define a subinterval $[\mu_i, \mu_{i+1}] \subset J$ as *active* if the projected Hermitian matrices $T_{U_k}(\mu_i) = U_k^* T(\mu_i) U_k$ and $T_{U_k}(\mu_{i+1}) = U_k^* T(\mu_{i+1}) U_k$ have different inertia. By Sylvester's law of inertia in our nonlinear setting [19, Theorem 3.3], an active interval necessarily contains eigenvalues of the projected problem $T_{U_k}(\nu)y = 0$. We start from the initial active interval J such that $\rho(x) \in J$ for all $x \neq 0$. Then we keep bisecting each active interval until it becomes sufficiently small. Without loss of generality, let $[\mu_i, \mu_{i+1}]$ be such a sufficiently small active interval. Then we apply two steps of Rayleigh functional iteration [31] starting with the approximate Ritz pair (ν_0, y_0), where $\nu_0 = \frac{1}{2}(\mu_i + \mu_{i+1})$ and y_0 is the singular vector of $T_{U_k}(\nu_0)$ corresponding to the smallest singular value. We find this procedure generally reliable for finding *all* Ritz pairs of the projected eigenproblem. Note that the main cost of this algorithm is the computation of the inertia of many small Hermitian matrices, which can be done efficiently by MATLAB's LDL^T factorization. Other options are also possible for solving truly nonlinear projected eigenproblems; see [45, Chapter 5]. An in-depth discussion of the strengths and weaknesses of each dense eigensolver is beyond the scope of this paper, because the way the projected eigenproblems are solved is independent to the framework of our PLMR methods.

6.1. PLMR vs. PLHR. In this section, we demonstrate that the PLMR's symmetry-preserving extraction strategy based on the refined Rayleigh–Ritz is crucial for the eigensolver's robustness. In particular, we compare PLMR to its version where the refined Rayleigh–Ritz is replaced with the harmonic projection. To distinguish between the two schemes, we refer to the latter as the preconditioned locally harmonic residual (PLHR) algorithm. Note that the same name is used for an interior linear eigenvalue solver [39], which only loosely relates to the approach considered here. The non-Hermitian nonlinear eigenproblem $U_k^* T(\sigma)^* T(\nu) U_k y = 0$, encountered by PLHR within the harmonic Rayleigh–Ritz procedure, is solved for the harmonic Ritz pair associated with the harmonic Ritz value closest to σ, using the residual inverse iteration [16], [26].

Table 2 summarizes the performance of the two methods for computing the eigenvalue closest to a given shift. Let us take the problem *wiresaw*, for instance, to explain the results. We choose the shift $\sigma = 800$, and we use the incomplete LDL$^{\mathrm{T}}$ factorization of $T(\sigma)$ with drop tolerance $\tau_d = 0.8$ as the preconditioner to compute the eigenvalue $\lambda = 801.026019$. The algorithms are terminated once the relative eigenresidual $\|r_k\| = \frac{\|T(\mu_k)x_k\|_2}{\|T(\mu_k)\|_F \|x_k\|_2}$ of the computed eigenpair (μ_k, x_k) satisfies $\|r_k\| \leq \tau_e = 10^{-10}$. Starting with the same random initial approximation x_0, it takes 14 iterations for both PLMR(2) and PLHR(2) to find the desired eigenvalue.

TABLE 2
Comparison of PLMR(2) *and* PLHR(2) *in iterations.*

Problem	σ	λ	Parameters	PLMR(2)	PLHR(2)
wiresaw	800	801.026019	$\tau_d = 0.8$, $\tau_e = 10^{-10}$	14	14
genhyper	-300	-283.145566	$\tau_d = 0$,[3] $\tau_e = 10^{-10}$	6	8
sleeper	-9.1	-9.09813985	$\tau_d = 0.005$, $\tau_e = 10^{-10}$	19	19
string	4.9×10^7	48974187.5	$\tau_d = 0.06$, $\tau_e = 10^{-12}$	22	22
artificial	0.2	0.19999002	$\tau_d = 0.01$, $\tau_e = 10^{-10}$	20	∞
pdde	0	0.00149342689	$\tau_d = 0.001$,$\tau_e = 10^{-10}$	14	∞

Table 2 shows that PLMR converges at least as rapidly as PLHR for the four initial tests, whereas PLHR fails to converge (marked as ∞) for the two remaining problems. The convergence failure occurs because PLHR stagnates with an eigenvalue approximation of low accuracy (e.g., with an eigenresidual norm around 10^{-4}) not very close to σ. This stagnation can be fixed, however, by choosing a new shift σ closer to the desired eigenvalue, or by using a stronger preconditioner.

Thus, while the convergence rates of PLMR and its PLHR variant are similar, the former tends to be more robust with respect to the quality of preconditioners and the choice of the shift σ. In addition, the robustness of PLMR strongly relies on a properly defined eigenvector extraction procedure, based on the symmetry-preserving refined Rayleigh–Ritz approach.

6.2. Effectiveness of the refinement procedure. In this section, we illustrate the importance of using the refined projection to stabilize the convergence of PLMR. We show that PLMR converges considerably more robustly than the version without the refinement step, which only discards spurious Ritz values and uses the strategy described in section 3.3 to choose a Ritz pair as the new eigenpair approximation.

To demonstrate the effect of the refined projection, we compare PLMR(2) with the simplified variants of PLMR(2) and PLMR(4) that do not invoke the refinement procedure. We run the three methods with the same random initial approximations, repeating the experiment 20 times, and show in Table 3 the number of times each method successfully finds the desired eigenvalue in 100 iterations and the average count of preconditioned matvecs needed for the successful runs.

Table 3

Comparison of PLMR(2) *with two variants that do not perform the refinement step: number of successful runs and average counts of preconditioned matvecs.*

Problem	σ	Parameters	PLMR(2)		PLMR(2) w/o refinement		PLMR(4) w/o refinement	
wiresaw	1000	$\tau_d = 10^{-3}$, $\tau_e = 10^{-10}$	20/20	12.3	3/20	95.0	19/20	30.7
gen_hyper	-200	$\tau_d = 0$, $\tau_e = 10^{-10}$	20/20	11.1	6/20	26.7	10/20	15.2
loaded_string	10^5	$\tau_d = 10^{-3}$, $\tau_e = 5 \times 10^{-12}$	20/20	6.0	16/20	19.3	20/20	8.0
sleeper	-2	$\tau_d = 10^{-3}$, $\tau_e = 10^{-10}$	20/20	11.3	5/20	41.6	19/20	58.9
pdde	-1	$\tau_d = 10^{-4}$, $\tau_e = 10^{-10}$	18/20	9.2	9/20	30.7	19/20	17.8
artificial	0.5	$\tau_d = 10^{-4}$, $\tau_e = 10^{-10}$	20/20	11.4	5/20	12.4	17/20	14.6

We see clearly that the refined projection is crucial for the stabilization of convergence for PLMR. For example, we look for the eigenvalue of the problem *wiresaw* closest to $\sigma = 1000$, using the incomplete $\mathrm{LDL}^{\mathrm{T}}$ preconditioner with drop tolerance $\tau_d = 10^{-3}$. The relative tolerance for the computed eigenpair is $\tau_e = 10^{-10}$. PLMR(2) always managed to find the desired eigenvalue, and it took 12.3 preconditioned matvecs on average to achieve convergence. Without the refinement step, by contrast, this method succeeded only 3 times, and on average it took 95 preconditioned matvecs to converge. In addition, PLMR(4) without the refinement step converged 19 times, and it took an average 30.7 preconditioned matves to find the desired eigenpair. In fact, with only one exception (for the problem *pdde*), PLMR(2) exhibits more robust and rapid convergence than the two variants without the refinement step. Note that for *pdde*, PLMR(2) was only marginally less robust than

[3]There is no need to construct an incomplete $\mathrm{LDL}^{\mathrm{T}}$ preconditioner for *genhyper* because $T(\sigma)$ is diagonal.

PLMR(4) without refinement, but was still considerably more efficient than the latter. In fact, with a stronger incomplete $\mathrm{LDL}^{\mathrm{T}}$ preconditioner with drop tolerance 10^{-5}, PLMR(2) managed to outperform the latter in both robustness and efficiency.

6.3. Order of local convergence. In section 4, we presented a local convergence analysis of PLMR(m_k), showing that the new method could exhibit local linear, quadratic, and cubic convergence if m_k is sufficiently large for each step k. Here, we provide some numerical evidence to support the analysis. We note that, in general, perfect quadratic and cubic convergence are rarely observed in practice. In addition, it is impractical to choose the optimal m_k for each k to achieve the expected order of convergence. Instead, we simply let $m_{k+1} = \beta m_k$, where β is a small fixed integer, to illustrate that PLMR(m_k) can easily achieve superlinear and superquadratic convergence.

TABLE 4
Estimate of the order of local convergence for PLMR(m_k).

wiresaw ($\sigma = 800$)	$m_k = 2$	$m_{k+1} = 2m_k$	$m_{k+1} = 3m_k$
$\tau_d = 0.8$, $\gamma_0 = 0.42$	0.96	2.06	2.43
sleeper ($\sigma = -9.1$)	$m_k = 2$	$m_{k+1} = 2m_k$	$m_{k+1} = 3m_k$
$\tau_d = 0.01$, $\gamma_0 = 0.15$	0.98	1.72	2.51
string ($\sigma = 4.9 \times 10^7$)	$m_k = 3$	$m_{k+1} = 2m_k$	$m_{k+1} = 3m_k$
$\tau_d = 0.06$, $\gamma_0 = 0.1$	0.97	1.76	2.31
artificial ($\sigma = 0.2$)	$m_k = 3$	$m_{k+1} = 2m_k$	$m_{k+1} = 4m_k$
$\tau_d = 0.01$, $\gamma_0 = 0.2$	1.01	2.45	3.91
pdde ($\sigma = 0$)	$m_k = 3$	$m_{k+1} = 2m_k$	$m_{k+1} = 3m_k$
$\tau_d = 0.001$, $\gamma_0 = 0.25$	0.98	1.78	2.26

Table 4 gives the estimates of the order of local convergence of PLMR(m_k) for five test problems. Let us again take the problem *wiresaw* as an example to interpret the results. We are looking for the eigenvalue closest to $\sigma = 800$, and we use the incomplete $\mathrm{LDL}^{\mathrm{T}}$ factorization of $T(\sigma)$ with drop tolerance $\tau_d = 0.8$. The initial iterate is set as $x_0 = v + \gamma_0 u$, where v is the desired unit eigenvector, $\gamma_0 = 0.42$, and u is a fixed unit vector generated by MATLAB's `randn`. We let $m_k = 2$ and run twenty PLMR(m_k) steps. Then we record the relative eigenresidual $\|r_k\| = \frac{\|T(\rho_k)x_k\|_2}{\|T(\rho_k)\|_F\|x_k\|_2}$ for each k and generate points $(\log\|r_k\|, \log\|r_{k+1}\|)$, for which we find the corresponding linear least squares fitting $y = ax + b$. The estimated order of convergence is the slope $a = 0.96$ of the linear fit. Next, we let $m_0 = 2$, $m_{k+1} = 2m_k$, and run three PLMR(m_k) steps. Using the same approach, we obtain an estimated convergence of order 2.06. Finally, we let $m_0 = 2$, $m_{k+1} = 3m_k$, and we run two PLMR(m_k) steps to get the convergence order of 2.43. For all problems, we run 20, 3, and 2 steps, respectively, to capture linear, quadratic, and cubic convergence.

Our results show clearly that PLMR(m_k) with a small fixed m_k converges linearly, and it converges superlinearly and superquadratically with $m_{k+1} = 2m_k$ and $m_{k+1} = 3m_k$ (or $m_{k+1} = 4m_k$), respectively. In fact, higher order of local convergence is observed for the problem *artificial*. We note, however, the efficiency of PLMR(m_k) primarily depends on the total number of preconditioned matvecs, instead of the order of local convergence. A higher order of convergence is achieved with increasingly larger value of m_k as the method proceeds. Overall, our experience is that the total number of preconditioned matves needed to achieve a certain level of eigenresidual tolerance largely depends on the quality of the preconditioner, and it is relatively insensitive to the order of local convergence.

6.4. PLMR vs. JD. We now compare PLMR(m) with the JD methods where the right-preconditioned GMRES(m) is used as an inner solver (further referred to as JD-GMRES(m)). Our numerical results show that PLMR(m) is at least as efficient as, and is often superior to, JD-GMRES(m), especially when working with search subspaces of a modest dimension.

In section 4 we studied a close connection between PLMR(m) and the basic variant of the JD-GMRES(m) algorithm. It is now of interest to compare the two methods, both in terms of local and global convergence.

As in the previous section, we choose a shift σ for each test problem, construct the corresponding incomplete LDL$^{\mathrm{T}}$ preconditioner with certain drop tolerance, and run PLMR(m) and basic JD-GMRES(m) with the same initial iterate x_0 to find the eigenvalue around the given shift. With a randomly generated x_0, PLMR(m) with a sufficiently large m always converges to the desired eigenvalue closest to σ, whereas basic JD-GMRES(m) always misconverges to a different eigenvalue. This is what we expected, as basic JD without subspace expansion has poor global convergence, unless a fixed shift is used in the correction equation for sufficiently many steps before a good eigenvector approximation can be obtained. The PLMR(m) method, in contrast, consistently exhibits a robust global convergence.

Next, let us compare the local convergence of the two algorithms with optimal values of m. To construct the initial iterate, for both methods, we let $x_0 = v + \gamma_0 u$, where v is the desired unit eigenvector, u is a fixed unit perturbation vector generated by `random` function, and γ_0 is a small scalar representing the error of x_0.

Take the problem *wiresaw* with shift $\sigma = 800$ as an example. The preconditioner used is the incomplete LDL$^{\mathrm{T}}$ factorization of $T(\sigma)$ with drop tolerance $\tau_d = 0.05$, 0.5, and 0.8, respectively. The initial iterate is constructed as $x_0 = v + \gamma_0 u$, where $\gamma_0 = 0.25$. Table 5 shows that with $\tau_d = 0.05$, the optimal value m for both PLMR(m) and JD-GMRES(m) is $m = 2$ (in parentheses), since it leads to the smallest total number of preconditioned matvecs. PLMR(2) converges in 6 steps, taking 12 preconditioned matvecs and 14.05 seconds, to achieve the relative tolerance $\frac{\|T(\rho_k)x_k\|_2}{\|T(\rho_k)\|_F\|x_k\|_2} \le \tau_e = 10^{-12}$. JD-GMRES(2) also converges in 6 iterations, taking 18 preconditioned matvecs and 20.84 seconds. The results of the algorithms for higher drop tolerances of incomplete LDL$^{\mathrm{T}}$ preconditioners are obtained similarly.

We see the following patterns in the performance comparison of the local convergence.

1. PLMR(m) outperforms the basic JD-GMRES(m) in the total number of preconditioned matvecs in essentially all circumstances. This is partially due to the fact that the former and the latter take m and $m + 1$ preconditioned matvecs, respectively, in each iteration step. Consequently, PLMR(m) also tends to perform better in CPU time if the preconditioned matvec is expensive. This is an advantage of PLMR over other types of preconditioned eigensolvers, such as the nonlinear Arnoldi method, which converges considerably slower than JD if the preconditioner is weak [41], [44].
2. As the quality of preconditioners deteriorates, the optimal values of m for PLMR(m) and basic JD-GMRES(m) increase, and the latter tends to take less CPU time. This is natural, because as m increases, the cost of other computational components in PLMR(m), such as the refined Rayleigh–Ritz projection, becomes more pronounced. Such algorithmic components are intrinsically more expensive than the linear solver GMRES(m) for large m. In this case, the superiority of PLMR might be retrieved by replacing the weak

TABLE 5
Comparison of PLMR(m) *and basic* JD-GMRES(m) *in iterations, preconditioned* **matvecs**, *CPU time, and optimal values of* m *(in parentheses).*

wiresaw		$\tau_d = 0.05$	$\tau_d = 0.5$	$\tau_d = 0.8$
$\sigma = 800$	PLMR	6 **12** 14.05s (2)	6 **24** 18.22s (4)	4 **28** 19.66s (7)
$\tau_e = 10^{-12}$, $\gamma_0 = 0.25$	JD-GMRES	6 **18** 20.84s (2)	4 **28** 22.31s (6)	4 **36** 23.42s (8)
genhyper		$\tau_d = -$		
$\sigma = -300$	PLMR	6 **12** 0.22s (2)		
$\tau_e = 10^{-12}$, $\gamma_0 = 0.04$	JD-GMRES	5 **20** 0.14s (3)		
sleeper		$\tau_d = 0.002$	$\tau_d = 0.01$	$\tau_d = 0.02$
$\sigma = -9.1$	PLMR	4 **16** 1.07s (4)	3 **39** 1.81s (13)	3 **66** 3.98s (22)
$\tau_e = 10^{-12}$, $\gamma_0 = 0.1$	JD-GMRES	4 **20** 0.78s (4)	3 **42** 1.09s (13)	4 **72** 2.26s (17)
string		$\tau_d = 0.01$	$\tau_d = 0.06$	$\tau_d = 0.1$
$\sigma = 4.9 \times 10^7$	PLMR	4 **8** 0.82s (2)	12 **36** 2.96s (3)	8 **72** 4.73s (9)
$\tau_e = 10^{-14}$, $\gamma_0 = 0.1$	JD-GMRES	5 **15** 0.98s (2)	7 **56** 2.48s (7)	5 **70** 2.66s (13)
artificial		$\tau_d = 0.002$	$\tau_d = 0.01$	$\tau_d = 0.02$
$\sigma = 0.2$	PLMR	7 **28** 4.43s (4)	6 **54** 6.33s (9)	7 **343** 48.67s (49)
$\tau_e = 10^{-12}$, $\gamma_0 = 0.1$	JD-GMRES	9 **45** 5.88s (4)	7 **56** 5.50s (7)	8 **385** 32.68s (54)
pdde		$\tau_d = 0.0002$	$\tau_d = 0.001$	$\tau_d = 0.0016$
$\sigma = 0$	PLMR	3 **24** 6.99s (8)	3 **30** 8.25s (10)	4 **44** 11.81s (11)
$\tau_e = 10^{-12}$, $\gamma_0 = 0.1$	JD-GMRES	5 **30** 7.94s (5)	3 **39** 8.41s (12)	3 **54** 10.69s (17)

preconditioned operation $M^{-1} \approx T(\sigma)^{-1}$ with a stronger one, e.g., an approximate linear solve with the coefficient matrix $T(\sigma)$ to a reasonably small tolerance, e.g., 10^{-3} to 10^{-6}.

In the following set of tests, we show that PLMR(m) is also more efficient than the full-featured JD-GMRES(m) method with a search subspace of variable dimension for the Rayleigh–Ritz projection (also referred to as full JD with subspace acceleration). Specifically, we compare PLMR(5) with JD-GMRES(5) working with a search subspace of dimensions 5, 10, and 20 (denoted as JD-GMRES(5)+RR(5), etc.), respectively, for computing 10 eigenvalues closest to σ. Our implementation of JD is based on that described in [6], where the only difference is that our correction equation is formulated as in (3.3) and (3.4), using an identical projector for both algorithms. We let both methods start with the same random initial approximation x_0, repeat the experiment 10 times, and take the average of the number of preconditioned matvecs. The parameters used to run the tests, together with results, are summarized in Table 6.

We see from Table 6 that PLMR(5) is considerably more efficient than JD-GMRES(5)+RR(5) and is essentially at least as efficient as JD-GMRES(5)+RR(10). We note that PLMR(5) uses a search subspace of dimension 5, whereas the two variants of JD-GMRES(5), respectively, work with search subspaces of total dimension $5 + 5 = 10$ and $5 + 10 = 15$, for the inner GMRES and outer JD iterations.

We also tested other small values of m for the two algorithms and found similar patterns in performance. The robust convergence of PLMR is ensured by the refined projection, whereas such a robustness of JD can only be achieved by the use of a large search subspace. For example, as can be seen in Table 6, the JD methods require three to five times more storage to become competitive to PLMR(m). Thus, PLMR(m) is more efficient in both arithmetic and storage costs when a search subspace is of a modest dimension.

TABLE 6
Comparison of PLMR(5) *and* JD-GMRES(5) + Rayleigh–Ritz*: preconditioned matvecs counts for computing* 10 *eigenvalues around* σ.

	PLMR(5)	JD-GMRES(5) + RR(5)	JD-GMRES(5) + RR(10)	JD-GMRES(5) + RR(20)
wiresaw ($\sigma = 1000$) $\tau_e = 10^{-10}$, $\tau_d = 10^{-3}$	255	1375	484	248
genhyper ($\sigma = -200$) $\tau_e = 10^{-10}$, $\tau_d = 0$	628	1581	636	344
sleeper ($\sigma = -2$) $\tau_e = 10^{-10}$, $\tau_d = 10^{-3}$	265	2281	775	396
string ($\sigma = 10^5$) $\tau_e = 5 \times 10^{-12}$, $\tau_d = 10^{-3}$	273	812	371	191
pdde ($\sigma = -1$) $\tau_e = 10^{-10}$, $\tau_d = 10^{-4}$	172	495	165	148
artificial ($\sigma = 0.5$) $\tau_e = 10^{-10}$, $\tau_d = 10^{-4}$	181	325	174	110

6.5. PLMR vs. BPLMR. In this section, we perform some tests to show that BPLMR is generally more competitive than PLMR in arithmetic cost for solving a group of clustered eigenvalues. Such a conclusion has been well established for linear eigenproblems, but to the best of our knowledge, this is the first time it has been done in a nonlinear setting.

We compare PLMR(2) and BPLMR(2) with the same random initial approximations to compute five eigenvalues around the shift σ. The drop tolerances τ_d of incomplete $\mathrm{LDL}^{\mathrm{T}}$ preconditioner and eigenresidual tolerances τ_e are also given. Recall that both algorithms use soft deflation of converged eigenpairs, and PLMR computes the eigenvalues sequentially, whereas BPLMR generates the desired approximations simultaneously.

Table 7 shows that BPLMR(2) is more efficient than PLMR(2) in arithmetic cost and CPU time for most problems. For the problem *wiresaw*, for instance, it takes PLMR(2) 172 iterations, equivalently 344 preconditioned matvecs, and 324.56 seconds to find the desired 5 eigenpairs. In contrast, it takes BPLMR(2) 11 iterations, or equivalently 150 preconditioned matvecs, and only 37.50 seconds to converge. The performance difference for the two methods is minimal for the problem *pdde*. For the problem *artificial*, PLMR(2) failed to find the fourth eigenvalue around σ in 500 iterations, but BPLMR(2) managed to find all five eigenvalues. Clearly, for the same m, the block method is preferable in arithmetic efficiency, as long as sufficient memory is available for the larger search subspace it develops.

6.6. Computing many successive eigenvalues. To verify the reliability of the new deflation techniques, we use BPLMR with the moving-window-style partial deflation described in section 5.5 to compute a large number of extreme eigenvalues. We then compare the results with those obtained by PCG methods, which are most reliable in this setting.

Table 8 summarizes the performance of LOBPCG and BPLMR for computing a few hundred or more extreme eigenvalues of the eight test problems. We take the problem *string* as an example to explain the results. The $n_d = 400$ lowest (L) eigenpairs of this problem are computed to the relative tolerance $\frac{\|T(\lambda_i)v_i\|_2}{\|T(\lambda_i)\|_F\|x_i\|_2} \leq 10^{-10}$ ($1 \leq i \leq 400$). Block methods are used to find these eigenvalues sequentially, 10

TABLE 7

Comparison of PLMR(2) *and* BPLMR(2) *for computing* 5 *eigenvalues around the shift* σ*, in iterations, preconditioned* **matvecs**, *and CPU time.*

wiresaw ($\sigma = 800$)	PLMR(2)	172	**344**	324.56s
$\tau_d = 10^{-3}$,$\tau_e = 10^{-10}$	BPLMR(2)	11	**150**	37.50s
genhyper ($\sigma = -300$)	PLMR(2)	155	**310**	5.74s
$\tau_d = 0$, $\tau_e = 10^{-10}$	BPLMR(2)	18	**178**	3.32s
sleeper ($\sigma = -9.1$)	PLMR(2)	117	**234**	22.68s
$\tau_d = 10^{-3}$, $\tau_e = 10^{-10}$	BPLMR(2)	10	**150**	14.85s
string ($\sigma = 4.9 \times 10^7$)	PLMR(2)	115	**230**	30.68s
$\tau_d = 10^{-3}$, $\tau_e = 10^{-12}$	BPLMR(2)	9	**138**	13.61s
artificial ($\sigma = 0.2$)	PLMR(2)			∞
$\tau_d = 10^{-4}$, $\tau_e = 10^{-10}$	BPLMR(2)	6	**50**	8.02s
pdde ($\sigma = 0$)	PLMR(2)	36	**72**	34.36s
$\tau_d = 10^{-4}$, $\tau_e = 10^{-10}$	BPLMR(2)	16	**76**	31.64s

TABLE 8

Comparison of LOBPCG *and* BPLMR *for computing successive eigenvalues: preconditioned matvecs and CPU time (in secs). Eigenresidual tolerance is* 10^{-12} *for problem string and* 10^{-10} *for others.*

				LOBPCG		LOBPCG+BPLMR(2)				
Problem	n_d	Group size	Block size	Precond. matvecs	CPU time	Window size	Precond. matvecs	CPU time	Missed	Repeated
wiresaw	500 (L)	10	12	6010	**6566**	4	7916	**3704**	0	0
genhyper	500 (H)	10	12	8657	**7623**	4	9028	**359**	0	0
string	400 (L)	10	12	5095	**49501**	4	5843	**1434**	0	0
artificial	500 (H)	10	12	5337	**77905**	3	7317	**2300**	1	0
pdde	400 (H)	5	8	4973	**61155**	4	6457	**3284**	1	0
sleeper	501 (L)	10	12	5239	**9452**	4	5804	**1234**	0	0
Laplace2d	2001 (L)	10	12	-	-	5	63749	**13269**	1	0
Laplace3d	10002 (L)	16	20	-	-	6	199606	**369806**[4]	3	7

eigenvalues each group, from the lowest to the highest ones. For each group of 10 eigenvalues, the block size is set to be 12 (slightly greater than 10) to stabilize the convergence. Once one group of eigenvalues is found, we compute the midpoint $\sigma = \frac{\lambda_r+\lambda_{r+1}}{2}$ between the two rightmost distinct computed eigenvalues λ_r and λ_{r+1}, and let the new preconditioner be the $\mathrm{LDL}^{\mathrm{T}}$ factorization of $T(\sigma)$. Such a preconditioner is expected to accelerate convergence toward subsequent eigenvalues near σ.

To illustrate the performance of our new method, we use LOBPCG to find the lowest four (window size) blocks of eigenvalues, and then run BPLMR(2) with the moving-window-style partial deflation of the most recently converged window-sized blocks of eigenvalues. This approach is compared with LOBPCG alone for computing all desired eigenvalues. As Table 8 shows, it takes LOBPCG 5095 preconditioned matvecs and 49501 seconds to find the 400 lowest eigenvalues, whereas it takes the new method 5843 preconditioned matvecs and only 1434 seconds. Our new approach not only avoided repeated convergence by partial deflation, but also did not miss any eigenvalues for this problem. The CPU time of the new method is lower, because it uses partial deflation, whereas the orthogonalization costs needed for complete defla-

[4]Performed on an iMac desktop computer running Mac OS X 10.8.5, MATLAB R2012b, with a 2.9 GHz Intel Core i5 processor and 16GB 1600MHz DDR3 memory.

tion are the bottleneck in LOBPCG, despite that the latter has lower matvec counts. Similarly for the problem *artificial*, the highest (H) 500 eigenvalues are computed. It takes LOBPCG and the new method 5337 and 7317 preconditioned matvecs and 77905 and 2300 seconds, respectively. Only 1 eigenvalue is missed by BPLMR, and no repeated convergence occurs. We note that even block PCG methods could occasionally miss a few extreme eigenvalues of linear Hermitian eigenproblems [2].

As we can see, our new approach is essentially as reliable as PCG methods for computing extreme eigenvalues, but is significantly less expensive when a large number n_d of eigenvalues are desired. The more eigenvalues are needed, the more advantage our method has over PCG methods. We emphasize that our test is simply an illustration of the reliability and efficiency of the new algorithm in the setting of computing all successive eigenvalues on a real interval. This method can be used to find many successive interior eigenvalues as well.

Table 8 also shows that BPLMR is quite reliable in finding semisimple eigenvalues with correct multiplicities. As usual, we count a distinct eigenvalue with multiplicity ℓ as ℓ eigenvalues. The last three problems in the table, namely, *sleeper*, *Laplace*2*d*, and *Laplace*3*d*, all have a dominant majority of semisimple eigenvalues. Specifically, only the lowest and the highest eigenvalues of *sleeper* are simple, and the rest are semisimple with multiplicity 2. We see that BPLMR finds all the lowest 501 eigenvalues with correct multiplicities. For *Laplace*2*d*, among the lowest 2001 eigenvalues, 35 are simple and others are semisimple with multiplicity 2. BPLMR obtains all these eigenvalues with correct multiplicities, with the only exception being that one semisimple eigenvalue is found with an incorrectly lowered multiplicity 1. *Laplace*3*d* is the most challenging problem, as only 15 eigenvalues among the lowest 10002 ones are simple, and there are 379 distinct semisimple eigenvalues with multiplicity 3, 1391 with multiplicity 6, and 42 with multiplicity 12. BPLMR finds a vast majority of these eigenvalues correctly. It misses 3 eigenvalues, and it converges repeatedly to 7 eigenvalues because those eigenvalues had already moved out of the window. Using a larger window size will reduce the occurrence of repeated convergence.

In terms of efficiency, the most remarkable pattern we see from Table 8 is as follows. LOBPCG based on the optimization of Rayleigh functional values always converges in fewer iterations than BPLMR, but the latter is significantly less expensive in arithmetic cost and thus takes much less CPU time if many (a few hundred or more) eigenvalues are desired. This observation is also clearly illustrated in Figure 1 for problems *sleeper* and *genhyper* as an example. As we explained, this is because BPLMR uses partial deflation, instead of the highly expensive complete deflation, as LOBPCG does. Moreover, BPLMR based on partial deflation only needs a *fixed* amount of memory that depends on the block size, the search subspace dimension, and the window size, but *not* on the total number of desired eigenvalues n_d. The converged eigenvectors that have moved out of the window can be put on external storage because they will not be involved in subsequent computation of new eigenvalues. We see from Table 8 and Figure 1 that BPLMR with the moving-window-style deflation strategy is highly competitive if a large number of successive eigenvalues are desired.

7. Conclusion. We have developed a preconditioned locally minimal residual (PLMR) method for computing interior eigenvalues of nonlinear Hermitian eigenproblems $T(\lambda)v = 0$ that admit a variational characterization of eigenvalues. We discussed the construction of the search subspace, stabilization of preconditioning, subspace projection and extraction, deflation, local convergence, and the extension to block variants. Our new algorithms are competitive in the rate and the robustness of

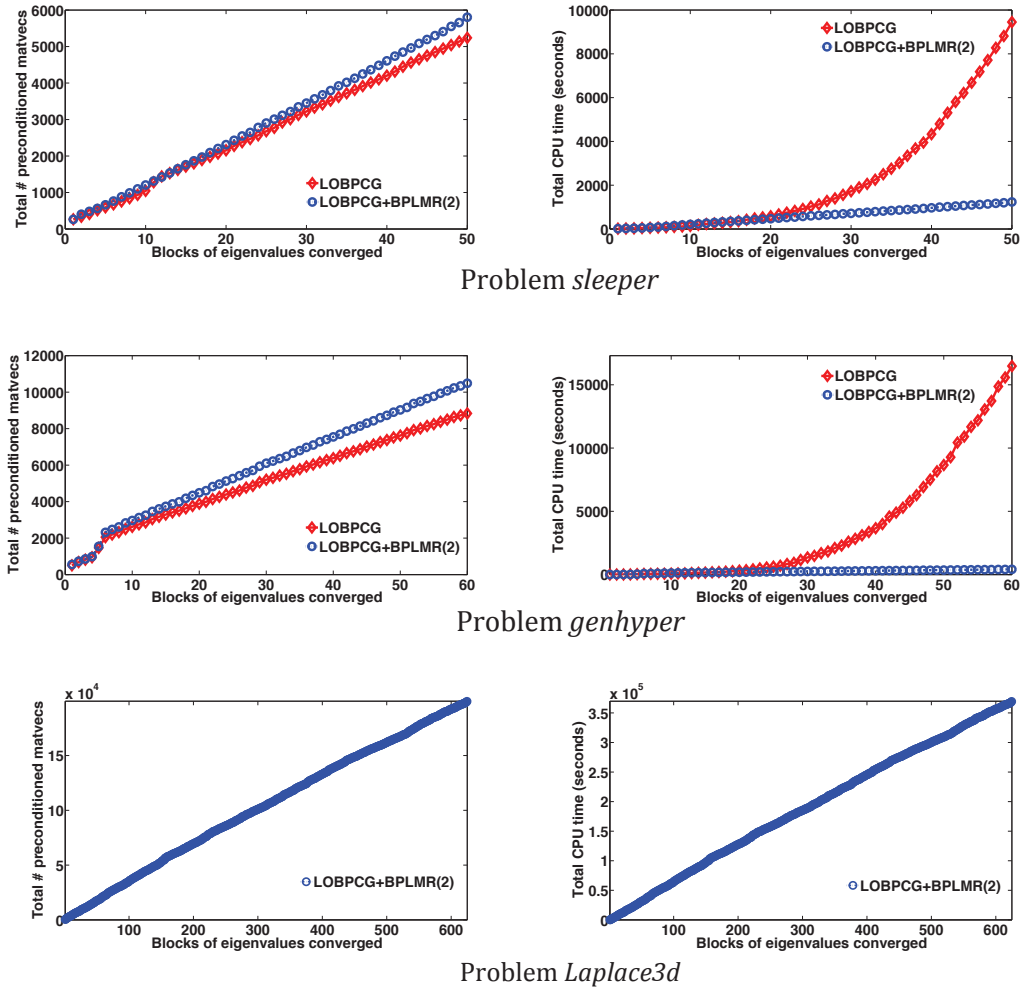

FIG. 1. *Total number of preconditioned matrix-vector products and CPU time used by* LOBPCG *and* LOBPCG+BPLMR(2). *Top: sleeper, lowest* 500 *eigenvalues Medium: genhyper, highest* 600 *eigenvalues Bottom: Laplace3d, lowest* 10002 *eigenvalues* (*only* LOBPCG+BPLMR(2) *is used).*

convergence toward desired interior eigenvalues near a given shift. We also proposed a moving-window-style partial deflation strategy that enables BPLMR to compute a large number of successive eigenvalues. Numerical experiments show that the new approach is reliable and is dramatically more efficient than PCG methods for computing many extreme eigenvalues.

Appendix.

DEFINITION A.1. *Assume that* λ *is an isolated simple eigenvalue of the analytic matrix-valued function* $T(\cdot) : \mathbb{R} \to \mathbb{C}^{n\times n}$ *associated with the left eigenvector* w *and right eigenvector* v *such that* $T(\lambda)v = 0$, $w^*T(\lambda) = 0$, *and* $w^*T'(\lambda)v = 1$; *see* [20, Theorem A.10.1]. *Let* $W_{n-1} \in \mathbb{C}^{n\times(n-1)}$ *be a matrix whose orthonormal columns form a basis of the orthogonal complement of* $\operatorname{span}\{T'(\lambda)v\}$. *Then, for a given vector*

$x \neq 0$, define $\gamma = \left\| \left[\begin{smallmatrix} w \\ W_{n-1}^* \end{smallmatrix} \right] T'(\lambda)x \right\|$ as the generalized norm of *x, and define*

$$s = \frac{\|W_{n-1}^* T'(\lambda)x\|}{\gamma}, \quad c = \frac{w^* T'(\lambda)x}{\gamma}, \quad \text{and } t = \frac{s}{c} = \frac{\|W_{n-1}^* T'(\lambda)x\|}{w^* T'(\lambda)x} \tag{A.1}$$

as the generalized sine, cosine, and tangent of the angle between x and v, *respectively.*

Acknowledgment. We thank the referees for their comments and suggestions, which helped us improve our presentation.

REFERENCES

[1] M. Al-Ammari and F. Tisseur, *Hermitian matrix polynomials with real eigenvalues of definite type. Part* I: *Classification*, Linear Algebra Appl., 436 (2012), pp. 3954–3973.

[2] P. Arbenz, U. L. Hetmaniuk, R. B. Lehoucq and R. S. Tuminaro, *A comparison of eigensolvers for large-scale* 3*D modal analysis using AMG-preconditioned iterative methods*, Internat. J. Numer. Methods Engrg., 64 (2005), pp. 204–236.

[3] M. M. Betcke and H. Voss, *Restarting Iterative Projection Methods for Hermitian Nonlinear Eigenvalue Problems with Minmax Property*, Report 157, Institute of Mathematics, Hamburg University of Technology, 2014.

[4] T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder, and F. Tisseur, *NLEVP: A collection of nonlinear eigenvalue problems*, ACM Trans. Math. Software, 39 (2013), article 7.

[5] T. Betcke and D. Kressner, *Perturbation, extraction and refinement of invariant pairs for matrix polynomials*, Linear Algebra Appl., 435 (2011), pp. 514–536.

[6] T. Betcke and H. Voss, *A Jacobi-Davidson type projection method for NEPs*, Future Generation Computer Systems, 20 (2004), pp. 363–372.

[7] C. Effenberger, *Robust successive computation of eigenpairs for nonlinear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1231–1256.

[8] D. R. Fokkema, G. L. G. Sleijpen, and H. A. Van der Vorst, *Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils*, SIAM J. Sci. Comput., 20 (1998), pp. 94–125.

[9] M. A. Freitag and A. Spence, *Rayleigh quotient iteration and simplified Jacobi-Davidson method with preconditioned iterative solves*, Linear Algebra Appl., 428 (2008), pp. 2049–2060.

[10] R. W. Freund and N. M. Nachtigal, *Software for simplified Lanczos and QMR algorithms*, Appl. Numer. Math., 19 (1995), pp. 319–341.

[11] K. P. Hadeler, *Variationsprinzipien bei nichtlinearen Eigenwertaufgaben*, Arch. Rational Mech. Anal., 30 (1968), pp. 297–307.

[12] M. Hagemann and O. Schenk, *Weighted matchings for preconditioning symmetric indefinite linear systems*, SIAM J. Sci. Comput., 28 (2006), pp. 403–420.

[13] M. E. Hochstenbach and Y. Notay, *Controlling inner iterations in the Jacobi–Davidson method*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 460–477.

[14] E. Jarlebring, *The Spectrum of Delay-Differential Equations: Numerical Methods, Stability and Perturbation*, Ph.D. thesis, Inst. Comp. Math, TU Braunschweig, 2008.

[15] E. Jarlebring, K. Meerbergen, and W. Michiels, *Computing a partial Schur factorization of nonlinear eigenvalue problems using the infinite Arnoldi method*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 411–436.

[16] E. Jarlebring and W. Michiels, *Analyzing the convergence factor of residual inverse iteration*, BIT, 51 (2011), pp. 937–957.

[17] E. Jarlebring, W. Michiels, and K. Meerbergen, *A linear eigenvalue algorithm for the nonlinear eigenvalue problem*, Numer. Math., 122 (2012), pp. 169–195.

[18] Z. Jia, *Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems*, Linear Algebra Appl., 259 (1997), pp. 1–23.

[19] A. Kostić and H. Voss, *On Sylvester's law of inertia for nonlinear eigenvalue problems*, Electron. Trans. Numer. Anal., 40 (2013), pp. 82–93.

[20] V. Kozlov and V. Mazýa, *Differential Equations with Operator Coefficients: With Applications to Boundary Value Problems for Partial Differential Equations*, Springer Monographs in Mathematics, Springer-Verlag, Berlin, Heidelberg, 1999.

[21] D. Kressner, *A block Newton method for nonlinear eigenvalue problems*, Numer. Math., 114 (2009), pp. 355–372.
[22] R.-C. Li, *Rayleigh Quotient Based Optimization Methods for Eigenvalue Problems*, Technical Report 2014-04, Department of Mathematics, University of Texas at Arlington, 2014. Lecture summary for 2013 Gene Golub SIAM Summer School; to appear in Series in Contemporary Applied Mathematics, Vol. 19; also available at http://www.siam.org/students/g2s3/2013/lecturers/RCLi/Summary_RCLi.pdf.
[23] X. Liang and R.-C. Li, *The hyperbolic quadratic eigenvalue problem*, Forum Math., Sigma, 3 (2015), e13.
[24] J. Money and Q. Ye, *Algorithm* 845: *EIGIFP: A MATLAB program for solving large symmetric generalized eigenvalue problems*, ACM Trans. Math. Software, 31 (2005), pp. 270–279.
[25] R. B. Morgan, *Computing interior eigenvalues of large matrices*, Linear Algebra Appl., 154–156 (1991), pp. 289–309.
[26] A. Neumaier, *Residual inverse iteration for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal., 22 (1985), pp. 914–923.
[27] E. Ovtchinnikov, *Convergence estimates for the generalized Davidson method for symmetric eigenvalue problems* II: *The subspace acceleration*, SIAM J. Numer. Anal., 41 (2003), pp. 272–286.
[28] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
[29] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[30] O. Schenk, M. Bollhöfer, and R. A. Römer, *On large-scale diagonalization techniques for the Anderson model of localization*, SIAM Rev., 50 (2008), pp. 91–112.
[31] K. Schreiber, *Nonlinear Eigenvalue Problems: Newton-Type Methods and Nonlinear Rayleigh Functionals*, Ph.D. thesis, Department of Mathematics, TU Berlin, 2008.
[32] G. L. G. Sleijpen and H. A. Van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.
[33] G. W. Stewart, *Matrix Algorithms. Volume* II: *Eigensystems*, SIAM, Philadelphia, 2001.
[34] D. B. Szyld and F. Xue, *Local convergence analysis of several inexact Newton-type algorithms for general nonlinear eigenvalue problems*, Numer. Math., 123 (2013), pp. 333–362.
[35] D. B. Szyld and F. Xue, *Several properties of invariant pairs of nonlinear algebraic eigenvalue problems*, IMA J. Numer. Anal., 34 (2014), pp. 921–954.
[36] D. B. Szyld and F. Xue, *Preconditioned Eigensolvers for Large-Scale Nonlinear Hermitian Eigenproblems with Variational Characterizations.* I. *Conjugate Gradient Methods*, Research Report 14-08-26, Department of Mathematics, Temple University, 2014. To appear in Mathematics of Computation.
[37] E. Vecharynski, *Preconditioned Iterative Methods for Linear Systems, Eigenvalue and Singular Value Problems*, Ph.D. thesis, Department of Mathematics, University of Colorado, Denver, 2011.
[38] E. Vecharynski and A. Knyazev, *Absolute value preconditioning for symmetric indefinite linear systems*, SIAM J. Sci. Comput., 35 (2013), pp. A696–A718.
[39] E. Vecharynski and A. Knyazev, *Preconditioned locally harmonic residual method for computing interior eigenpairs of certain classes of Hermitian matrices*, SIAM J. Sci. Comput., 37 (2015), pp. S3–S29.
[40] H. Voss, *An Arnoldi method for nonlinear eigenvalue problems*, BIT, 44 (2004), pp. 387–401.
[41] H. Voss, *Iterative projection methods for computing relevant energy states of a quantum dot*, J. Comput. Phys., 217 (2006), pp. 824–833.
[42] H. Voss, *A Jacobi-Davidson method for nonlinear and nonsymmetric eigenproblems*, Comput. & Structures, 85 (2007), pp. 1284–1292.
[43] H. Voss, *A minmax principle for nonlinear eigenproblems depending continuously on the eigenparameter*, Numer. Linear Algebra Appl., 16 (2009), pp. 899–913.
[44] H. Voss, *Iterative projection methods for large-scale nonlinear eigenvalue problems*, in Computational Technology Reviews, Vol. 1, B. H. V. Topping, J. M. Adams, F. J. Pallarés, R. Bru, and M. L. Romero, eds., Saxe-Coburg Publications, Stirling, UK, 2010, pp. 187–214.
[45] H. Voss, *Nonlinear eigenvalue problems*, in Handbook of Linear Algebra, L. Hogben, ed., CRC Press, Boca Raton, FL, 2014, Chapter 60.
[46] H. Voss and B. Werner, *A minimax principle for nonlinear eigenvalue problems with applications to nonoverdamped systems*, Math. Methods Appl. Sci., 4 (1982), pp. 415–424.