

Trace-Penalty Minimization for Large-scale Eigenspace Computation

Zaiwen Wen*

Chao Yang[†]

Xin Liu[‡]

Yin Zhang[§]

March 6, 2013

Abstract

The Rayleigh-Ritz (RR) procedure, including orthogonalization, constitutes a major bottleneck in computing relatively high-dimensional eigenspaces of large sparse matrices. Although operations involved in RR steps can be parallelized to a certain level, their parallel scalability, which is limited by some inherent sequential steps, is lower than dense matrix-matrix multiplications. The primary motivation of this paper is to develop a methodology that reduces the use of the RR procedure in exchange for matrix-matrix multiplications. We propose an unconstrained penalty model and establish its equivalence to the eigenvalue problem. This model enables us to deploy gradient-type algorithms that makes heavy use of dense matrix-matrix multiplications. Although the proposed algorithm does not necessarily reduce the total number of arithmetic operations, it leverages highly optimized operations on modern high performance computers to achieve parallel scalability. Numerical results based on a preliminary implementation, parallelized using OpenMP, show that our approach is promising.

Keywords. eigenvalue computation, exact quadratic penalty approach, gradient methods

AMS subject classifications. 15A18, 65F15, 65K05, 90C06

1 Introduction

Eigenvalue and eigenvector calculation is a fundamental computational problem with extraordinarily wide-ranging applications. In the past several decades, a great deal of progress has been made in the development

*Department of Mathematics, MOE-LSC and Institute of Natural Sciences, Shanghai Jiaotong University, CHINA (zw2109@sjtu.edu.cn). Research supported in part by NSFC grant 11101274, and Humboldt Research Fellowship for Experienced Researchers.

[†]Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, UNITED STATES (cyang@lbl.gov). Support for this work was provided through the Scientific Discovery through Advanced Computing (SciDAC) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (and Basic Energy Sciences) under award number DE-SC0008666.

[‡]State Key Laboratory of Scientific and Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, CHINA (liuxin@lsec.cc.ac.cn). Research supported in part by NSFC grant 11101409 and 10831006, and the National Center for Mathematics and Interdisciplinary Sciences, CAS.

[§]Department of Computational and Applied Mathematics, Rice University, UNITED STATES (yzhang@rice.edu). Research supported in part by NSF Grant DMS-0811188, ONR Grant N00014-08-1-1101, and NSF Grant DMS-1115950.

Report Documentation Page			Form Approved OMB No. 0704-0188	
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.				
1. REPORT DATE 06 MAR 2013	2. REPORT TYPE		3. DATES COVERED 00-00-2013 to 00-00-2013	
4. TITLE AND SUBTITLE Trace-Penalty Minimization for Large-scale Eigenspace Computation			5a. CONTRACT NUMBER	
			5b. GRANT NUMBER	
			5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)			5d. PROJECT NUMBER	
			5e. TASK NUMBER	
			5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Rice University, Department of Computational and Applied Mathematics, Houston, TX, 77005			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSOR/MONITOR'S ACRONYM(S)	
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited				
13. SUPPLEMENTARY NOTES				
14. ABSTRACT The Rayleigh-Ritz (RR) procedure, including orthogonalization, constitutes a major bottleneck in computing relatively high-dimensional eigenspaces of large sparse matrices. Although operations involved in RR steps can be parallelized to a certain level, their parallel scalability, which is limited by some inherent sequential steps, is lower than dense matrix-matrix multiplications. The primary motivation of this paper is to develop a methodology that reduces the use of the RR procedure in exchange for matrix-matrix multiplications. We propose an unconstrained penalty model and establish its equivalence to the eigenvalue problem. This model enables us to deploy gradient-type algorithms that makes heavy use of dense matrix-matrix multiplications. Although the proposed algorithm does not necessarily reduce the total number of arithmetic operations, it leverages highly optimized operations on modern high performance computers to achieve parallel scalability. Numerical results based on a preliminary implementation, parallelized using OpenMP, show that our approach is promising.				
15. SUBJECT TERMS				
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Same as Report (SAR)	18. NUMBER OF PAGES 30
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified		

of efficient algorithms and solvers for various types of eigenvalue problems. Iterative methods are usually preferred for solving large-scale problems because of their ability to take advantage of sparsity or other structures existing in the matrices of interest. When a few eigenpairs are needed, the task of sparse matrix-vector multiplications, which can often be performed efficiently on both sequential and modern parallel computers, usually constitutes the dominant computational cost. However, as the number of desired eigenpairs increases, the computational costs in an iterative eigensolver can shift to other linear algebra operations.

There are two types of operations that can potentially become bottlenecks. One is the construction and/or maintenance of orthonormal bases for subspaces from which approximate eigenvalues and eigenvectors are extracted at each iteration. This type of operations is often carried out through either a Gram-Schmidt (including Arnoldi or Lanczos) procedure or a QR factorization at the complexity of at least $O(nk^2)$ where n is the dimension of the target matrix and k is the number of desired eigenpairs. Another potentially high-cost procedure is the Rayleigh-Ritz (RR) calculation [13] used to extract eigenvalue and eigenvector approximations from a subspace of dimension $p \geq k$. The RR procedure involves solving a p -dimensional dense eigenvalue problem and assembling the so-called Ritz vectors which are approximate eigenvectors in the original space. Because the Ritz vectors are mutually orthonormal, the RR procedure can sometimes be viewed as a way to construct an orthonormal basis also. The complexity for the RR procedure is at least $O(nk^2 + k^3)$. When the number k is small, the costs of these two types of operations are minor or even negligible. However, when k increases to a moderate portion of the matrix dimension n , these costs can represent a significant, even dominant, portion of the overall cost.

The use of parallel computers can greatly reduce the solution time. However, to make efficient use of these computers, we must ensure that our algorithm is scalable with respect to the number of processors or cores. Although the standard Krylov subspace iterative algorithms can be parallelized through the parallelization of the sparse matrix vector multiplications (SpMV) and other dense linear algebra operations, the amount of parallelism is limited because SpMVs must be done in sequence in these algorithms, and each SpMV can only make effective use of a limited number of processing units in general. Block methods, such as the locally optimal block preconditioned conjugate gradient (LOBPCG) algorithm [10], the block Krylov-Schur algorithm [21] and the Chebyshev-Davidson algorithm [19, 20], are more scalable because more concurrency can be exploited in multiplying a sparse matrix with a block of vectors.

However, block methods have so far not addressed the relatively high cost of performing an RR calculation at each iteration. Although parallel algorithms for solving the dense projected eigenvalue problem are available in multi-thread LAPACK [2] libraries for shared-memory parallel computers and in the ScaLAPACK [4] library for distributed-memory parallel computers, the parallel efficiency of these algorithms is often limited to a relatively small number of processors or cores. When a large number of processing units are involved, the thread or communication overhead can be significant. One way to address this issue is to use a “spectrum slicing” algorithm [1, 5, 8] that divides the part of the spectrum of interest into a number of intervals and compute eigenvalues within each interval in parallel. However, this approach would require computing interior eigenvalues in each interval which is generally a more difficult task. Moreover, a good initial guess of the eigenvalues of interest is needed so that the spectrum can be divided in an efficient

manner [1].

In the Chebyshev-Davidson algorithm [19, 20], the number of RR steps is amortized over a large number of SpMV's because a Chebyshev matrix polynomial filter is applied to a block of vectors before an RR calculation is performed to update the approximate eigenvectors. However, an apparent drawback of this algorithm is the difficulty to take advantage of a good pre-conditioner when it is available. In addition, it is difficult to apply a Chebyshev polynomial filter to generalized eigenvalue problems.

In this paper, we present a block algorithm for computing k algebraically smallest eigenvalues of a real symmetric matrix $A \in \mathbb{R}^{n \times n}$ and their corresponding eigenvectors, though the same methodology can easily be applied to compute the largest eigenvalues and to compute eigenvalues of complex Hermitian matrices. Our approach starts from the trace minimization formulation for eigenvalue problems. It is well known that the invariant subspace associated with a set of k algebraically smallest eigenvalues of A yields an optimal solution to the following trace minimization problem with orthogonality constraints

$$\min_{X \in \mathbb{R}^{n \times k}} \text{tr}(X^T A X), \text{ s.t. } X^T X = I. \quad (1)$$

A major theoretical result of this paper is to establish an equivalence relationship between problem (1) and the following unconstrained optimization problem

$$\min_{X \in \mathbb{R}^{n \times k}} f_\mu(X) := \frac{1}{2} \text{tr}(X^T A X) + \frac{\mu}{4} \|X^T X - I\|_F^2, \quad (2)$$

when the penalty parameter $\mu > 0$ takes suitable finite values. As is well recognized, the objective function in (2) is the classic quadratic (or Courant) penalty function [6, 12, 15] for the constrained problem (1). Generally speaking, the classic quadratic penalty model approaches the original constrained problem only as the penalty parameter μ goes to infinity. However, we show that in terms of finding an optimal eigenspace problem (2) is essentially equivalent to (1) when the penalty parameter μ is appropriately chosen.

We will call the approach of solving model (2) *trace-penalty minimization*. A key difference between trace minimization and trace-penalty minimization is that explicit orthogonality of X is no longer required in the latter, which immediately opens up the possibility of doing far fewer RR steps including far fewer orthogonalizations and other RR-related operations. In exchange, as will be demonstrated later, more dense matrix-matrix multiplications are performed (to a less extent, also more SpMV). A major potential advantage of replacing RR steps by dense matrix-matrix multiplications is that the latter operations have much better parallel scalability and are highly optimized for modern high performance computers. In addition, one could incorporate pre-conditioning into trace-penalty minimization in a straightforward manner.

In this paper, we consider applying gradient-type methods to the trace-penalty minimization problem (2). These methods can often quickly reach the vicinity of an optimal solution and produce a moderately accurate approximation. In many applications, rough or moderately accurate approximations are often sufficient. One of such instances is when solving a nonlinear eigenvalue problem, one approximately solves a sequence of linearized eigenvalue problems one after another (such as in solving the Kohn-Sham equation in electronic structure calculation by “self-consistent field” iterations [14]). In this paper we evaluate the

efficiency of our algorithm not by measuring the time it takes to compute eigenpairs to a high accuracy close to machine precision, but rather the time it takes to achieve a moderate accuracy in computed eigenpairs. Once good estimates are at hand, there exist a number of techniques that can perform further refinements to obtain a higher accuracy. For example, moderately accurate estimates can be further improved by using a “spectral slicing” type of algorithm [1, 5, 8]. Another possibility is to apply polynomial filtering to do refinements. For the proposed trace-penalty minimization problem (2), we have experimented with various algorithmic options in Matlab, developed a Fortran implementation and parallelize it using OpenMP. Preliminary numerical comparison with some of the existing approaches shows that our approach is promising.

The rest of this paper is organized as follows. We analyze the trace-penalty minimization model in Section 2. Our algorithms and several implementation details are discussed in Section 3. Numerical results are reported in Section 4. Finally, we conclude the paper in Section 5.

2 Trace-Penalty Minimization: Model Analysis

For a given real symmetric matrix $A = A^T \in \mathbb{R}^{n \times n}$, an eigenvalue decomposition of A is defined as

$$A = Q_n \Lambda_n Q_n^T, \quad (3)$$

where, for any integer $i \in [1, n]$,

$$Q_i = [q_1, q_2, \dots, q_i] \in \mathbb{R}^{n \times i}, \quad \Lambda_i = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_i) \in \mathbb{R}^{i \times i}, \quad (4)$$

so that $Q_i^T Q_i = I \in \mathbb{R}^{i \times i}$ and Λ_i is diagonal. The columns $q_1, \dots, q_n$ of Q_n are eigenvectors of A associated with eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$, respectively, which are assumed to be in an ascending order,

$$\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n.$$

We note the non-uniqueness of eigenvalue decomposition (3). One could not only alter the signs of eigenvectors, but also choose different unit eigenvectors associated with eigenvalues of multiplicity greater than one. For convenience, we will treat (3) as a generic form of decomposition that represents all possible alternatives.

Given a positive integer $k \leq n$, it is well known that the eigenvector matrix Q_k is a solution to the trace minimization problem (1). As is stated in the introduction, instead of solving (1) directly, we propose to solve the trace-penalty minimization problem (2). We first analyze the relationship between the two problems (1) and (2), and then derive some useful properties for (2).

2.1 Equivalence and other Properties

We start with the following definition of equivalence.

Definition 2.1. Problem (2) is said to be equivalent to (1) if each of its global minimizers spans a k -dimensional eigenspace associated with k smallest eigenvalues of A .

The first-order necessary conditions for trace minimization (1) can be written as

$$AX - X(X^T AX) = 0, \quad X^T X = I.$$

On the other hand, the first-order necessary condition for trace-penalty minimization (2) is simply

$$\nabla f_\mu(X) = AX + \mu X(X^T X - I) = 0. \quad (5)$$

Let $L(\mathbb{R}^{n \times k}, \mathbb{R}^{n \times k})$ be the space of linear operators that map $\mathbb{R}^{n \times k}$ to $\mathbb{R}^{n \times k}$. The Fréchet derivative of ∇f_μ at X is defined as the (unique) function $\nabla^2 f_\mu : \mathbb{R}^{n \times k} \rightarrow L(\mathbb{R}^{n \times k}, \mathbb{R}^{n \times k})$ such that

$$\lim_{\|S\|_F \rightarrow 0} \frac{\|\nabla f_\mu(X + S) - \nabla f_\mu(X) - \nabla^2 f_\mu(X)(S)\|_F}{\|S\|_F} = 0.$$

It can be easily verified that

$$\nabla^2 f_\mu(X)(S) = AS + \mu S(X^T X - I) + \mu X(S^T X + X^T S), \quad (6)$$

from which one can also derive the matrix representation of $\nabla^2 f_\mu(X)$ in terms of Kronecker products.

The first-order necessary condition (5) implies that each stationary point X of (2) spans an invariant subspace of A , since $\nabla f_\mu(X) = 0$ is obviously equivalent to

$$AX = X(I - X^T X)\mu.$$

Trivially, $X = \mathbf{0} \in \mathbb{R}^{n \times k}$ is always a stationary point of (2). We first study this trivial stationary point and show that it can be eliminated as a minimizer if the penalty parameter μ is sufficiently large.

Lemma 2.2. Let $\mu > 0$. If $\mu \leq \lambda_1$, the zero matrix $X = \mathbf{0} \in \mathbb{R}^{n \times k}$ is the only stationary point of problem (2); otherwise, it is not a minimizer. Moreover, $X = \mathbf{0}$ is a maximizer when $\mu > \lambda_n$.

Proof. Rearranging (5), we have

$$(\mu I - A)X = \mu X(X^T X). \quad (7)$$

Multiplying X^T on both side of (7) yields

$$X^T(\mu I - A)X = \mu(X^T X)^2. \quad (8)$$

If $\mu \leq \lambda_1$, the matrix on the left is negative semidefinite while the one on the right is positive semidefinite, forcing the only solution $X = \mathbf{0}$. When $\mu > \lambda_1$, it suffices to note that the Hessian of f_μ at $X = \mathbf{0}$ is $\nabla^2 f_\mu(\mathbf{0}) = I \otimes (A - \mu I)$ which is not positive semidefinite. Finally, we note that $\nabla^2 f_\mu(\mathbf{0})$ is negative definite when $\mu > \lambda_n$. $\square$

The next lemma shows that any stationary point of (2) can be expressed in terms of eigenpairs of A .

Lemma 2.3. *Let $\mu > 0$ and $(U, D) \in \mathbb{R}^{n \times k} \times \mathbb{R}^{k \times k}$ denote k eigenpairs of A so that $AU = UD$, $U^T U = I$ and D is diagonal. A matrix $X \in \mathbb{R}^{n \times k}$ is a stationary point of (2) if and only if*

$$X = U[P(I - D/\mu)]^{1/2}V^T, \quad (9)$$

where $V \in \mathbb{R}^{k \times k}$ is an arbitrary orthogonal matrix, and $P \in \mathbb{R}^{k \times k}$ is a diagonal, projection matrix with diagonal entries

$$P_{ii} = \begin{cases} 0, & \text{if } \mu \leq D_{ii}, \\ 0 \text{ or } 1, & \text{otherwise.} \end{cases} \quad (10)$$

In Particular, X is a rank- k stationary point only if $P = I$ and $\mu I - D \succ 0$ (being positive definite).

Proof. We will provide a proof for the case where X is of full-rank. The rank-deficient cases can be proved along the similar line, though more notationally involved and tedious.

Suppose that X is a full rank stationary point, which spans an invariant subspace of A . Since every k -dimensional invariant subspace of A can be spanned by a set of k eigenvectors, we can write $X = UW$ where U consists of k unit eigenvectors of A and $W \in \mathbb{R}^{k \times k}$ is nonsingular. Upon substituting $X = UW$ into (7), we derive

$$U(\mu I - D)W = \mu UW(W^T W) \Leftrightarrow I - D/\mu = WW^T \Leftrightarrow W = (I - D/\mu)^{1/2}V^T$$

for some orthogonal $V \in \mathbb{R}^{k \times k}$ (which can possibly hold only if $\mu > D_{ii}$ for $i = 1, 2, \dots, k$). $\square$

Now we establish the equivalence between the trace-penalty minimization model (2) and the trace minimization model (1) for proper μ values.

Theorem 2.4. *Problem (2) is equivalent to (1) if and only if*

$$\mu > \max(0, \lambda_k). \quad (11)$$

Specifically, any global minimizer $\hat{X}$ of (2) has a singular-value decomposition of the form:

$$\hat{X} = Q_k(I - \Lambda_k/\mu)^{1/2}V^T \quad (12)$$

where Q_k and Λ_k are defined as in (4), and $V \in \mathbb{R}^{k \times k}$ is any orthogonal matrix.

Proof. It can be easily seen from (8) that condition (11) is necessary for the existence of a rank- k stationary point. On the other hand, suppose that μ satisfies (11). Using Lemma 2.3, it is suffice to consider the representation $X = UW$, where U consists of any k eigenvectors of A and $W \in \mathbb{R}^{k \times k}$. Hence, we obtain

$$2f_\mu(X) = \text{tr}(DWW^T) + \frac{\mu}{2}\|W^T W - I\|_F^2,$$

where $D = \text{Diag}(d) \in \mathbb{R}^{k \times k}$ is a diagonal matrix with k eigenvalues of A on the diagonal corresponding to eigenvectors in U . A short calculation shows that

$$\begin{aligned}
2f_\mu(X) &= \frac{\mu}{2} \|WW^T + (D/\mu - I)\|_F^2 + \text{tr}(D) - \frac{1}{2\mu} \text{tr}(D^2) \\
&\geq \frac{\mu}{2} \|(D/\mu - I)_+\|_F^2 + \text{tr}(D) - \frac{1}{2\mu} \text{tr}(D^2) \\
&= \sum_{i=1}^k \left(\frac{\mu}{2} \left(\frac{d_i}{\mu} - 1 \right)_+^2 + d_i - \frac{d_i^2}{2\mu} \right) \\
&= \sum_{i=1}^k \theta(d_i),
\end{aligned}$$

where $t_+ = \max(0, t)$ and

$$\theta(d) = \frac{\mu}{2} \left(\frac{d}{\mu} - 1 \right)_+^2 + d - \frac{d^2}{2\mu} = \begin{cases} d - d^2/(2\mu), & d < \mu, \\ \mu/2, & d \geq \mu. \end{cases}$$

Note that $\theta(d)$ is monotonically nondecreasing since $\theta'(d) = 1 - d/\mu > 0$ in $(-\infty, \mu)$.

Substituting the formulation of $\hat{X}$ defined in (12) into $f_\mu(\hat{X})$, we obtain

$$2f_\mu(\hat{X}) = \text{tr}(\Lambda_k) - \frac{1}{2\mu} \text{tr}(\Lambda_k^2) = \sum_{i=1}^k \theta(\lambda_i) \leq 2f_\mu(X),$$

which verifies that $\hat{X}$ is a global minimizer. This completes the proof. $\square$

The next theorem indicates that our trace-penalty minimization model (2) can have far fewer undesirable, full-rank stationary points than the trace minimization model (1). Hence, when the penalty parameter is suitably chosen, one could reasonably argue that from an optimization point of view trace-penalty minimization is theoretically more desirable than trace minimization.

Theorem 2.5. *If $\mu \in (\max(0, \lambda_k), \lambda_n)$, then $f_\mu(X)$ has no local maxima, nor local minima other than the global minimum attained by $\hat{X}$ defined in (12). Moreover, if $\mu \in (\max(0, \lambda_k), \lambda_{k+p})$ where λ_{k+p} is the smallest eigenvalue greater than λ_k , then all k -dimensional stationary points of $f_\mu(X)$ must be global minimizers.*

Proof. To prove the first statement, we show that for $\mu \in (\max(0, \lambda_k), \lambda_n)$ any stationary point other than the global minimizers can only be saddle points.

Without loss of generality, consider stationary points in the form of (9) with $V = I$, that is

$$\hat{X} = U[P(I - D/\mu)]^{1/2} = U[(I - D/\mu)P]^{1/2}, \tag{13}$$

where $AU = UD$, $U^T U = I$, and D is diagonal. The proof still holds for an arbitrary orthogonal matrix V since the function value $f_\mu(\hat{X})$ is invariant with respect to V . Substituting (13) into the Hessian formula

(6), we obtain

$$\nabla^2 f_\mu(\hat{X})(S) = AS - S(\mu(I - P) + DP) + \mu\hat{X}(S^T\hat{X} + \hat{X}^T S). \quad (14)$$

We next show that there exists two different matrices $S \in \mathbb{R}^{n \times k}$ such that $\text{tr}(S^T \nabla^2 f_\mu(\hat{X})(S)) < 0$ and $\text{tr}(S^T \nabla^2 f_\mu(\hat{X})(S)) > 0$, respectively, unless the stationary point $\hat{X}$ is constructed from eigenvectors associated with a set of k smallest eigenvalues which corresponds to the global minimum.

First assume that $\hat{X}$ has full rank. Then $\mu I \succ D$ and $P = I$ in (13). Letting $P = I$ in (14) yields

$$\nabla^2 f_\mu(\hat{X})(S) = AS - SD + \mu\hat{X}(S^T\hat{X} + \hat{X}^T S).$$

For $S = U$, we have $S^T\hat{X} = \hat{X}^T S = (I - D/\mu)^{1/2}$ and

$$\text{tr}(S^T \nabla^2 f_\mu(\hat{X})(S)) = 0 + 2 \text{tr}(\mu I - D) > 0.$$

On the other hand, if $\hat{X}$ is not a global minimizer, without loss of generality we can assume that U contains q_j but not q_i where $\lambda_i < \lambda_j$. Let S contain all zero columns except a single nonzero column that is q_i at the position so that the only nonzero column of SD is $q_i \lambda_j$. For such an S , we have $S^T\hat{X} = 0$ and

$$\text{tr}(S^T \nabla^2 f_\mu(\hat{X})(S)) = q_i^T(Aq_i - q_i \lambda_j) + \mu \text{tr}(S^T\hat{X}(S^T\hat{X} + \hat{X}^T S)) = (\lambda_i - \lambda_j) + 0 < 0.$$

Hence, all full-rank stationary points are saddle points except the global minimizers.

We now consider the rank-deficient case, namely, there exists at least one zero entry in the diagonal of P , say $P_{ii} = 0$ for some $i \in [1, k]$. Let $\bar{U}$ be the remaining matrix after deleting the i -th column from U . Since $\text{rank}(\bar{U}) = k - 1$, there must exist at least one column, denoted by q_j , of Q_k that is not contained in $\bar{U}$. Then it holds $q_j^T \bar{U} = 0$ and $q_j^T A q_j \leq \lambda_k$. Let S contain all zero columns except one nonzero column that is q_j at the i -th position so that both $SP = 0$ and $S^T\hat{X} = 0$. Consequently, in view of (14) we have

$$\text{tr}(S^T \nabla^2 f_\mu(\hat{X})(S)) = q_j^T A q_j - \mu + \mu \text{tr}(S^T\hat{X}(S^T\hat{X} + \hat{X}^T S)) \leq (\lambda_k - \mu) + 0 < 0.$$

On the other side, let S contain all zero columns except that the i -th column is q_n . For any integer $l \in [1, k]$, if the column $U_l = q_n$, then it follows from Lemma 2.3 that $P_{ll} = 0$ and $q_n^T \hat{X}_l = 0$. Otherwise, the column $U_l \neq q_n$, thus $q_n^T U_l = 0$ which implies $q_n^T \hat{X} = 0$. By our assumption, $\mu < q_n^T A q_n = \lambda_n$. Hence, $\text{tr}(S^T \nabla^2 f_\mu(\hat{X})(S)) = \lambda_n - \mu > 0$. This complete the proof of the first statement.

The second part of this theorem is a direct consequence of the full-rank requirement and the stationary-point expression (9) which, together, demands $\mu I \succ D$. Hence, for $\mu \in (\max(0, \lambda_k), \lambda_{k+p})$, D can only have a set of k smallest eigenvalues of A on its diagonal. $\square$

2.2 Error Bounds between Optimality Conditions

After establishing the equivalence between our trace-penalty minimization model (2) and the original trace minimization model (1), we investigate the relationship between the first-order optimality conditions of the

two models, which would play an important role in setting the stopping tolerance for an iterative algorithm to solve (2).

Given any approximate solution X of (2), an orthonormal basis for the range space of X , say $Y(X)$, is a feasible solution of (1). Specifically, let X be of full rank and $X = U\Sigma V^T$ denote the partial (or economy-form) singular value decomposition (SVD) of X , where $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{k \times k}$ have orthonormal columns, and $\Sigma \in \mathbb{R}^{k \times k}$ is a diagonal matrix with the singular values of X on its diagonal. Then a particular choice for $Y(X)$ is

$$Y(X) \triangleq U. \quad (15)$$

Consequently, the violation of the first-order necessary conditions of the trace minimization (1) can be measured by the Frobenious norm of the residual

$$R(X) \triangleq AY(X) - Y(X) (Y(X)^T AY(X)). \quad (16)$$

Lemma 2.6. *Let $\mu > \max(0, \lambda_k)$, and $\nabla f_\mu(X)$ and $R(X)$ be defined as in (5) and (16), respectively. Then*

$$\|R(X)\|_F \leq \sigma_{\min}^{-1}(X) \|\nabla f_\mu(X)\|_F, \quad (17)$$

where $\sigma_{\min}(X)$ is the smallest singular value of X . Moreover, for any global minimizer $\hat{X}$ and any $\epsilon > 0$, there exists $\delta > 0$ such that whenever $\|X - \hat{X}\|_F \leq \delta$,

$$\|R(X)\|_F \leq \frac{1 + \epsilon}{\sqrt{1 - \lambda_k/\mu}} \|\nabla f_\mu(X)\|_F. \quad (18)$$

Proof. Recall that $X = U\Sigma V^T$ where the columns of U form an orthonormal basis for the range space of X . Projecting $\nabla f_\mu(X)$ onto the null space of X^T and using the definition of $R(X)$ in (16), we obtain

$$\begin{aligned} (I - UU^T)\nabla f_\mu(X) &= (I - UU^T)(AX + \mu X(X^T X - I)) \\ &= (I - UU^T)AX = (I - UU^T)AU\Sigma V^T \\ &= R(X)\Sigma V^T. \end{aligned}$$

A rearrangement of the above equality gives

$$R(X) = (I - UU^T)\nabla f_\mu(X)V\Sigma^{-1},$$

which leads to (17) through the following steps,

$$\begin{aligned}
\|R(X)\|_F &= \|(I - UU^T)\nabla f_\mu(X)V\Sigma^{-1}\|_F \\
&\leq \|(I - UU^T)\nabla f_\mu(X)V\|_F \|\Sigma^{-1}\|_2 \\
&= \|(I - UU^T)\nabla f_\mu(X)\|_F \sigma_{\min}^{-1}(X) \\
&\leq \sigma_{\min}^{-1}(X) \|\nabla f_\mu(X)\|_F,
\end{aligned}$$

where the last inequality is due to the fact that $I - UU^T$ is a projection.

To see the second part of this lemma, we only need to recall Theorem 2.4 that gives

$$\sigma_{\min}(\hat{X}) = \sqrt{1 - \lambda_k/\mu},$$

for any global minimizer $\hat{X}$. This completes the proof. $\square$

2.3 Condition Number of the Hessian at Solution

An important quantity for a smooth unconstrained optimization model is the condition number of the Hessian at solution, which can be defined in our case as

$$\kappa(\nabla^2 f_\mu(\hat{X})) = \frac{\lambda_{\max}(\nabla^2 f_\mu(\hat{X}))}{\lambda_{\min}(\nabla^2 f_\mu(\hat{X}))},$$

where $\lambda_{\max}(\cdot)$ (or $\lambda_{\min}(\cdot)$) stands for the largest (or the smallest) eigenvalue of the referred matrix, and $\hat{X}$ is a global minimizer of (2). Obviously, κ is infinity when the involved matrix is singular.

In the next lemma, we calculate the condition number $\kappa(\nabla^2 f_\mu(X))$ for the case $k = 1$, and give a lower bound for the general case.

Lemma 2.7. *Let $\mu > \max(0, \lambda_k)$ and $\hat{X}$ be a global minimizer of f_μ in (2). The condition number of the Hessian of f_μ at $\hat{X}$ satisfies*

$$\kappa(\nabla^2 f_\mu(\hat{X})) \geq \frac{\max(2(\mu - \lambda_1), \lambda_n - \lambda_1)}{\min(2(\mu - \lambda_k), \lambda_{k+1} - \lambda_k)} \geq \frac{\lambda_n - \lambda_1}{\lambda_{k+1} - \lambda_k}. \quad (19)$$

In particular, for $k = 1$ the first inequality above holds as an equality.

Proof. As is well known, the largest or smallest eigenvalues of a symmetric matrix can be obtained by maximizing or minimizing the Rayleigh-Ritz quotient, namely, in our case,

$$\begin{aligned}
\lambda_{\max}(\nabla^2 f_\mu(\hat{X})) &= \max_{\text{tr}(S^T S)=1} \text{tr}(S^T \nabla^2 f_\mu(\hat{X})(S)), \\
\lambda_{\min}(\nabla^2 f_\mu(\hat{X})) &= \min_{\text{tr}(S^T S)=1} \text{tr}(S^T \nabla^2 f_\mu(\hat{X})(S)).
\end{aligned}$$

We first treat the case of $k = 1$ in which X and S refer to vectors, so we simplify them as $\hat{x}$ and s . Also,

the trace constraint $\text{tr}(S^T S) = 1$ becomes $s^T s = 1$. Applying theorem 2.4, $\hat{x}$ takes the form

$$\hat{x} = \pm q_1 \sqrt{1 - \lambda_1/\mu}.$$

Hence, $\mu(\hat{x}^T \hat{x} - 1) = -\lambda_1$. Using (6) with $s = Qy$ and $y^T y = 1$, we have

$$s^T \nabla^2 f_\mu(\hat{x}) s = s^T A s - \lambda_1 + 2\mu(s^T \hat{x})^2 = y^T [\Lambda - \lambda_1 I + 2(\mu - \lambda_1)e_1 e_1^T] y.$$

Therefore,

$$\begin{aligned} \max_{s^T s=1} s^T \nabla^2 f_\mu(\hat{x}) s &= \max(2(\mu - \lambda_1), \lambda_n - \lambda_1), \\ \min_{s^T s=1} s^T \nabla^2 f_\mu(\hat{x}) s &= \min(2(\mu - \lambda_1), \lambda_2 - \lambda_1). \end{aligned}$$

Hence, (19) holds as an equality.

Now we treat the general case. Let us recall theorem 2.4 that $\hat{X}$ takes the form of (12). Since

$$\text{tr}(S^T \nabla^2 f_\mu(\hat{X})(S)) = \text{tr}((SV)^T \nabla^2 f_\mu(\hat{X}V)(SV)),$$

and $\text{tr}((SV)^T SV) = \text{tr}(S^T S)$, the orthogonal matrix V in a solution $\hat{X}$, as is defined in (12), does not change the maximum or minimum of $\text{tr}(S^T \nabla^2 f_\mu(\hat{X})(S))$ under the trace constraint $\text{tr}(S^T S) = 1$. Without loss of generality, in this proof we set $V = I$ and only consider

$$\hat{X} = Q_k(I - \Lambda_k/\mu)^{1/2}.$$

In view of (6), we have

$$\text{tr}(S^T \nabla^2 f_\mu(\hat{X})(S)) = \text{tr}(S^T A S) - \text{tr}(S^T S \Lambda_k) + \mu \text{tr}((S^T \hat{X})^2 + S^T \hat{X} \hat{X}^T S).$$

Let all the columns of S be zero except that the j -th column is the unit eigenvector q_i of A , namely,

$$S = q_i e_j^T, \quad j \leq k, \quad e_j \in \mathbb{R}^k,$$

which satisfies the trace constraint $\text{tr}(S^T S) = 1$. Clearly,

$$S^T \hat{X} = e_j q_i^T Q_k(I - \Lambda_k/\mu)^{1/2} = \begin{cases} e_j e_i^T \sqrt{1 - \lambda_i/\mu}, & i \leq k, \\ 0, & \text{otherwise.} \end{cases}$$

It is not difficult to verify that

$$\text{tr}(S^T \nabla^2 f_\mu(\hat{X})(S)) = \lambda_i - \lambda_j + (\delta_{ij} + \gamma_i)(\mu - \lambda_i). \quad (20)$$

where δ_{ij} is the Kronecker delta and $\gamma_i = 1$ if $i \leq k$ and 0 otherwise. For different choices of i and $j \leq k$, the right-hand side of (20) can take the values of $\lambda_i - \lambda_j$ for $i > k \geq j$, and $2(\mu - \lambda_i)$ for $i = j \leq k$. These values imply the bounds

$$\begin{aligned}\lambda_{\max} \left(\nabla^2 f_{\mu}(\hat{X}) \right) &\geq \max(2(\mu - \lambda_1), \lambda_n - \lambda_1) \\ \lambda_{\min} \left(\nabla^2 f_{\mu}(\hat{X}) \right) &\leq \min(2(\mu - \lambda_k), \lambda_{k+1} - \lambda_k),\end{aligned}$$

from which we obtain (19) and complete the proof. $\square$

We note that in the general case of $k > 1$, the bound in (19) can be further tightened. For example, by letting $i < j = k$, we obtain the values of $\mu - \lambda_k$; hence the factor 2 can be eliminated from the denominator. One can also show that the Hessian becomes singular when there are eigenvalues λ_j , $j < k$, of multiplicity greater than one. In general, this kind of non-uniqueness does not necessarily imply a higher degree of difficulty in solving the optimization problem.

Even though the bound in (19) is not tight, it is good enough to serve the purpose of demonstrating two useful points: (a) the penalty parameter μ should not be chosen excessively close to λ_k ; and (b) one should expect some difficulty arising from the existence of a relatively tiny gap between λ_k and λ_{k+1} . This second difficulty, caused by clustered eigenvalues at a critical location, represents a common challenge to eigenvalue solvers.

2.4 Extensions

It is not difficult to see that our analysis in this section, as well as the algorithmic framework described in the next section, can be extended to the generalized eigenvalue problem:

$$\min_{X \in \mathbb{R}^{n \times k}} \text{tr}(X^T A X), \text{ s.t. } X^T B X = I, \quad (21)$$

where B is symmetric and positive definite. In this case, the trace-penalty minimization model is simply

$$\min_{X \in \mathbb{R}^{n \times k}} f_{\mu}(X) := \frac{1}{2} \text{tr}(X^T A X) + \frac{\mu}{4} \|X^T B X - I\|_F^2. \quad (22)$$

In fact, by change of variable $Z = B^{\frac{1}{2}} X$ (where the symmetric matrix $B^{\frac{1}{2}}$ satisfies $B^{\frac{1}{2}} B^{\frac{1}{2}} = B$), the generalized eigenvalue problem (21) can be converted to a standard eigenvalue problem

$$\min_{Z \in \mathbb{R}^{n \times k}} \text{tr}(Z^T \bar{A} Z), \text{ s.t. } Z^T Z = I, \quad (23)$$

where $\bar{A} = B^{-\frac{1}{2}} A B^{-\frac{1}{2}}$. As a result, our model analysis, directly applicable to (23), can be translated to (22) in a straightforward manner. For example, Theorem 2.4 gives the global minimizers of (23) as

$$Z = \bar{Q}_k(I - \Lambda_k/\mu)V^T,$$

where Λ_k is diagonal with k smallest eigenvalues of $\bar{A}$ on its diagonal that also happen to be the generalized eigenvalues of the matrix pair (A, B) , and $\bar{Q}_k$ consists of corresponding eigenvector of $\bar{A}$. By the change of variables $Z = B^{\frac{1}{2}}X$, then

$$X = Q_k(I - \Lambda_k/\mu)V^T, \quad (24)$$

where $Q_k = B^{-\frac{1}{2}}\bar{Q}_k$ consists of the generalized eigenvectors associated with the k smallest generalized eigenvalues in Λ_k . Naturally, the equivalence of the trace-penalty minimization model (22) to the trace minimization model (21) requires that $\mu > \max(0, \lambda_k)$ where λ_k is a k -th smallest generalized eigenvalue of the matrix pair (A, B) .

Another useful extension is to find eigenvectors in the orthogonal complement of the column-space of a given U such that $U^T U = I$, that is:

$$\min_{X \in \mathbb{R}^{n \times k}} \text{tr}(X^T A X), \text{ s.t. } X^T B X = I, \quad U^T X = 0. \quad (25)$$

The variation (25) can arise from a deflation procedure where U is constructed from already converged eigenvectors. The trace-penalty minimization model corresponding to (25) is

$$\min_{X \in \mathbb{R}^{n \times k}} f_\mu(X) := \frac{1}{2} \text{tr}(X^T A X) + \frac{\mu}{4} \|X^T B X - I\|_F^2, \text{ s.t. } U^T X = 0. \quad (26)$$

Starting from X^0 such that $U^T X^0 = 0$, a projected-gradient method for solving (26) has the form $X^{j+1} = X^j - \alpha^j (I - U U^T) \nabla f_\mu(X^j)$ which is just a slight modification of regular gradient methods to be discussed in details in the next section.

The principle of trace-penalty minimization can in fact be applied to other types of eigenvalue problems, but for the sake of space we will leave further extensions to future work.

3 Algorithmic Framework

3.1 Gradient Methods for Trace-Penalty Minimization

The trace-penalty minimization model proposed in the previous section is an unconstrained nonconvex minimization problem. There are many well-studied approaches for this problem, such as the steepest descent gradient, the conjugate gradient, the Newton's and Quasi-Newton methods. Considering the scale of the eigenvalue computation of interest, in this paper we focus on the gradient-type methods of the form:

$$X^{j+1} = X^j - \alpha^j \nabla f_\mu(X^j), \quad (27)$$

where the superscript j denotes the j -th iteration and α^j is the step size.

Although the penalty function may have multiple stationary points, Theorem 2.5 shows that when μ is chosen slightly above $\lambda_k \geq 0$, then rank- k stationary points are likely to be all global minimizers. The following lemma suggests that iterates generated by (27) will most likely remain full rank.

Proposition 3.1. *Let X^{j+1} be generated by (27) from a full-rank iterate X^j . Then X^{j+1} is rank-deficient only if $1/\alpha^j$ is one of the k generalized eigenvalues of the problem:*

$$[(X^j)^T \nabla f_\mu(X^j)]u = \lambda[(X^j)^T(X^j)]u. \quad (28)$$

On the other hand, if $\alpha^j < \sigma_{\min}(X^j)/\|\nabla f_\mu(X^j)\|_2$, then X^{j+1} is of full rank.

Proof. Suppose that X^{j+1} is rank deficient. Then there exists a nonzero vector u such that $X^{j+1}u = 0$. In view of (27), we have

$$X^j u - \alpha^j \nabla f_\mu(X^j)u = 0. \quad (29)$$

Hence, (28) holds under $\lambda = 1/\alpha^j$ after multiplying both sides of (29) by $(X^j)^T/\alpha^j$. Due to the full rank of X^j , $(X^j)^T(X^j)$ is positive definite. The expression of the gradient in (5) implies that $(X^j)^T \nabla f_\mu(X^j)$ is symmetric. Therefore, (28) is a generalized symmetric eigenvalue problem. The second part of the proposition follows directly from (29). $\square$

We next present a few strategies for choosing the step size α^j . Given an arbitrary direction $D \in \mathbb{R}^{n \times k}$, the objective function $f_\mu(X + \alpha D)$ is a quartic function of α ; precisely,

$$\begin{aligned} f_\mu(X + \alpha D) &= \frac{1}{2} \text{tr}(X^T A X) + \frac{\mu}{4} \text{tr}(B^T B) + \left(\text{tr}(D^T A X) + \frac{\mu}{2} \text{tr}(B^T W) \right) \alpha \\ &\quad + \left(\text{tr}(D^T A D) + \frac{\mu}{2} \text{tr}(B^T H) + \frac{\mu}{4} \text{tr}(W^T W) \right) \alpha^2 \\ &\quad + \left(\frac{\mu}{2} \text{tr}(W^T H) \right) \alpha^3 + \left(\frac{\mu}{4} \text{tr}(H^T H) \right) \alpha^4, \end{aligned} \quad (30)$$

where $B = X^T X - I$, $W = D^T X + X^T D$ and $H = D^T D$. The steepest descent gradient method computes the step size by using a one-dimensional exact minimization, i.e., $\alpha^j = \arg\min f_\mu(X^j - \alpha \nabla f(X^j))$, which is determined by a root of the cubic equation $df_\mu(X^j - \alpha \nabla f(X^j))/d\alpha = 0$. Note that $\mu > 0$ and $\text{tr}(H^T H) > 0$ for $\nabla f_\mu(X) \neq 0$, a positive root always exists. Although executing exact line searches along each steepest descent direction often converges slowly, it has been demonstrated in [16, 17] that mixing it with some other step sizes in an alternative fashion can accelerate convergence significantly.

Another successful approach is to use line search with a Barzilai-Borwein (BB) size [3]. Let

$$S^j := X^j - X^{j-1} \quad \text{and} \quad Y^j = \nabla f_\mu(X^j) - \nabla f_\mu(X^{j-1}). \quad (31)$$

The BB step size is

$$\alpha_{\text{BB1}}^j = \frac{\text{tr}((S^j)^T Y^j)}{\|Y^j\|_F^2} \quad \text{or} \quad \alpha_{\text{BB2}}^j = \frac{\|S^j\|_F^2}{\text{tr}((S^j)^T Y^j)}. \quad (32)$$

Since $S^j = \alpha^{j-1} \nabla f_\mu(X^{j-1})$, the computation of the BB step sizes only requires to store one intermediate matrix Y^j in (31). When n and k are huge or when storage becomes a critical factor, one can still compute

a so-called partial BB step size by using (32) but with a pre-selected small subset of columns of both S^j and Y^j , making the storage of an extra Y -matrix unnecessary. A simple heuristic line search scheme that we will use is to shorten the step size, whenever necessary, by back-tracking $\alpha^j = \alpha\delta^h$, where α is one of the BB step sizes in (32), $\delta \in (0, 1)$ and h is the smallest positive integer satisfying the condition

$$f_\mu(X^j - \alpha\delta^h \nabla f_\mu^j) \leq 2f_\mu(X^j). \quad (33)$$

It is known that certain global convergence properties can be guaranteed in theory by more elaborate line search conditions such as non-monotone line search conditions in [7, 9, 18]. We have found, however, that on our trace-penalty function $f_\mu(X)$ condition (33) has performed efficiently and reliably.

At the end of trace-penalty minimization, a Rayleigh-Ritz (RR) step is necessary to compute Ritz-pairs as approximations to eigenpairs. Specifically, in our context the RR step corresponding to a given matrix $X \in \mathbb{R}^{n \times k}$ is defined by the following steps.

1. Orthogonalize and normalize X to obtain U so that $U^T U = I$.
2. Compute the projection $U^T A U$ and its eigenvalue decomposition $V^T \Sigma V$.
3. Assemble the Ritz-pairs into the matrix-pair (Y, Σ) where $Y = UV$.

For convenience, we will refer the above RR procedure as a map $(Y, \Sigma) = \text{RR}(X)$.

In Algorithm 1 below, we specify a basic version of a method for trace-penalty minimization, called ‘‘EigPen-B’’, which uses the first BB step formula in (32), the simple line search condition (33), and a termination rule

$$\|\nabla f_\mu(X^j)\|_F \leq \epsilon, \quad (34)$$

where $\epsilon > 0$ is a prescribed tolerance.

Algorithm 1: Eigenspace by Penalty – basic version (EigPen-B)

Initialize $X^0 \in \mathbb{R}^{n \times k}$ and estimate $\mu \in (\lambda_k, \lambda_n)$. Set $\epsilon, \delta \in (0, 1)$ and $j = 0$.

Compute initial step $\alpha = \|X^0\|_F / \|\nabla f_\mu(X^0)\|_F$.

while $\|\nabla f_\mu(X^j)\|_F > \epsilon$ **do**

- compute the smallest natural number h so that $\alpha\delta^h$ satisfying (33);
- update $X^{j+1} = X^j - \alpha^j \nabla f_\mu(X^j)$;
- compute α using the first formula in (32);
- increment j and continue.

Execute the RR procedure $(X, \Sigma) = \text{RR}(X^j)$.

The memory requirement of Algorithm 1 is summarized as follows. Four n by k matrices, X , AX , $\nabla f_\mu(X)$ and Y defined in (31), are required. As mentioned earlier, the need for storing Y can be essentially eliminated if partial BB step sizes are computed (without obvious performance degradation in our experiments). Using the convention that an $m \times p$ matrix times a $p \times n$ matrix costs $2mnp$ flops, we summarize the computational complexity of various tasks as follows. Let s be the sparsity of the matrix A ,

i.e., $s = |\{A_{ij} | A_{ij} \neq 0\}|/n^2$. The matrix-matrix multiplication AX needs $2sn^2k$ flops while sn^2 is often around $O(n)$. Forming $X(X^T X - I)$ takes $4nk^2$. Each of the inner product $\text{tr}(S^T Y)$ and the calculation of the norms $\|\nabla f_\mu(X)\|_F$ and $\|Y\|_F$ takes $2nk$. Hence, the total cost of each step in trace-penalty minimization is at most $4nk^2 + 8nk + 2sn^2k + O(k^3)$. The orthogonalization step by using Cholesky factorizations needs $2nk^2 + O(k^3)$ since $X^T X$ comes free from trace-penalty minimization steps. The projection $U^T AU$ requires $2nk + 2sn^2k$. The eigenvalue decomposition of the projection in an RR step takes $O(k^3)$. Finally, assembling the Ritz-pairs takes another $2nk^2$. Therefore, the total complexity of the last RR step is $6nk^2 + 2sn^2k + O(k^3)$.

3.2 Enhancement by Restarting

Algorithm EigPen-B often works quite well in practice. However, a typical behavior of gradient methods is that they can reduce the objective function rather rapidly at an initial stage, but the amount of reduction can become extremely small as iterates get closer to a solution. In trace-penalty minimization, it has been observed that restarting the gradient method with a modified X can usually help accelerate convergence and achieve a higher accuracy more quickly. In this subsection, we describe a restarting strategy for trace-penalty minimization that utilizes more than one RR step. In addition to accelerating convergence, the restarting strategy provides a more reliable termination procedure by examining more than one set of Ritz-pairs.

We now demonstrate how RR steps can help speed up trace-penalty minimization. Let

$$Y = \underset{X \in \mathbb{R}^{n \times k}}{\operatorname{argmin}} \{f_\mu(X) : X \in \mathcal{S}\}, \quad (35)$$

where $X \in \mathcal{S}$ means that every column of X is in the subspace $\mathcal{S}$. Let X^J be the iterate generated by the EigPen-B algorithm after J iterations. Clearly, as long as $X^J \in \mathcal{S}$ there holds

$$f_\mu(Y) \leq f_\mu(X^J). \quad (36)$$

On the other hand, consider the subspace trace minimization problem

$$U = \underset{X \in \mathbb{R}^{n \times d}}{\operatorname{argmin}} \{ \text{tr}(X^T A X) : X^T X = I, X \in \mathcal{S} \}, \quad (37)$$

where d is the dimension of the subspace $\mathcal{S}$. Clearly, the RR step $(U, \Sigma) = \text{RR}(X^J)$ is equivalent to solving (37) for $\mathcal{S} = \text{span}\{X^J\}$ (assuming that $X^J \in \mathbb{R}^{n \times k}$ has full rank) so that $U^T A U = \Sigma$ is diagonal (otherwise, replace U by UV where $U^T A U = V \Sigma V^T$).

We now show that a “better point” Y for trace-penalty minimization in (35) and (36) can be explicitly constructed from the RR step output $(U, \Sigma) = \text{RR}(X^J)$. We first consider the simple case $\mathcal{S} = \text{span}\{X^J\}$.

Lemma 3.2. *Let $\mathcal{S} = \text{span}\{X^J\}$ where $X^J \in \mathbb{R}^{n \times k}$ has full rank, and let U be defined in (37) so that $U^T A U = \Sigma$ is diagonal. Then a Y in (35) has the form $Y = U(I - \Sigma/\mu)^{1/2}$, provided that $\mu I \succ \Sigma$.*

Now we prove a more general result that contains the above as a special case.

Lemma 3.3. Let $\mathcal{S} \supseteq \text{span}\{X^j\}$ have dimension $d \geq k$, and U be defined in (37) so that $U^T A U = \Sigma$ is diagonal whose diagonal elements are arranged in an ascending order. Then a matrix Y in (35) has the form $Y = U_k D$ where U_k consists of the first k columns of U , and $D \in \mathbb{R}^{k \times k}$ is a diagonal matrix whose i -th diagonal element is

$$D_{ii} = \max \left(0, 1 - \frac{1}{\mu} \Sigma_{ii} \right)^{1/2}, \quad i = 1, 2, \dots, k. \quad (38)$$

Proof. Since $U \in \mathbb{R}^{n \times d}$ is a basis of $\mathcal{S}$, the solution of (35) can be expressed as $X = UW$ for some $W \in \mathbb{R}^{d \times k}$. Substituting $X = UW$ into (35) and noting that $U^T A U = \Sigma$ and $U^T U = I$, we reduce (35) to

$$\min_{W \in \mathbb{R}^{d \times k}} f_\mu(UW) = \frac{1}{2} \text{tr}(W^T \Sigma W) + \frac{\mu}{4} \|W^T W - I\|_F^2. \quad (39)$$

Using the fact that Σ is a diagonal matrix, it can be verified (see Theorem 2.4) that $W = \begin{pmatrix} D & 0 \end{pmatrix}^T$, with the diagonal matrix D defined as in (38), is indeed a solution of (39). Therefore, $Y = UW = U_k D$. $\square$

In Algorithm 2 below, we present our trace-penalty minimization algorithm with restarting, which is used to perform numerical experiments presented in the next section. The algorithm, called EigPen, contains two loops. The inner loop is stopped once the condition

$$\|\nabla f_\mu(X^j)\|_F \leq \epsilon_i \max(1, \|AX^j\|_F) \quad (40)$$

is met, where $\epsilon_i \in (0, 1)$ is a prescribed tolerance. Then an RR step is executed to construct Ritz-pairs and termination criteria are checked for the outer loop. If the algorithm does not stop, then a smaller tolerance $\epsilon_{i+1} = \delta_\epsilon \epsilon_i$ is set where $\delta_\epsilon \in (0, 1)$, and a better iterate is constructed from which the algorithm restarts the next round of inner iterations by calling EigPen-B.

Algorithm 2: Eigenspace by Penalty – enhanced version (EigPen)

Initialize $\bar{X}^0 \in \mathbb{R}^{n \times k}$ and estimate $\mu \in (\lambda_k, \lambda_n)$. Set $\epsilon_0, \delta, \delta_\epsilon \in (0, 1)$ and $i = j = 0$.

while “not converged” **do**

 Set $X^j = \bar{X}^i$ and compute $\alpha = \|X^j\|_F / \|\nabla f_\mu(X^j)\|_F$.

while $\|\nabla f_\mu(X^j)\|_F > \epsilon_i \cdot \max(1, \|AX^j\|_F)$ **do**

 compute the smallest integer h so that $\alpha^j = \alpha \delta^h$ satisfying (33);

 update $X^{j+1} = X^j - \alpha^j \nabla f_\mu(X^j)$;

 compute α using the first formula in (32);

 increment j and continue.

 Execute the RR procedure $(X, \Sigma) = \text{RR}(X^j)$ and let $\bar{X}^{i+1} = X(I - \Sigma/\mu)^{\frac{1}{2}}$.

 Update the tolerance $\epsilon_{i+1} = \delta_\epsilon \epsilon_i$ and increment i .

The RR restart approach allows flexibility to integrate other techniques into EigPen. For example, at the j -th iteration, if one chooses the subspace $\mathcal{S}$ in problem (37) to be $\mathcal{S} = \text{span}\{X^{j-1}, X^j, AX^j\}$, then the

RR step would generate a step similar to those in the LOBPCG algorithm [10]. A key difference between LOBPCG and EigPen is that RR steps constitute the main workhorse of the former, but are utilized only a few times in the latter.

3.3 Penalty Parameter Adjustment

We now describe our approach to choosing penalty parameter μ . Theorem 2.4 states that $\mu > \max(0, \lambda_k)$ is necessary and sufficient for the equivalence between (1) and (2). A more restrictive range for μ is given in Theorem 2.5 that eliminates all full-rank stationary points but the global minimizers. However, it requires the extra work of estimating, at the least, λ_{k+1} . On the other hand, both Lemmas 2.6 and 2.7 suggest that μ should not be too close to λ_k , otherwise ill-conditioning could arise in trace-penalty minimization. On balance, we adopt a tractable strategy of choosing $\mu > \lambda_k$ (which is positive after a shifting if necessary) and keeping it reasonably close to λ_k , without attempting to make μ smaller than the next smallest eigenvalue.

Given an initial matrices $X^0 \in \mathbb{R}^{n \times k}$ whose columns are normalized, the k th smallest eigenvalue λ_k can be estimated by the maximal value of the diagonal entries of $(X^0)^T A X^0$, which provides an initial choice

$$\mu = \max(c_1, c_2 \max(\text{diag}((X^0)^T A X^0))), \quad (41)$$

where $c_1 > 0$ and $c_2 > 1$ are two constants for safeguarding. Another estimation of μ comes from the structure of the minimizer $\hat{X}$ given in (12), which yields the eigenvalue decomposition

$$\mu(I - \hat{X}^T \hat{X}) = V \Lambda_k V^T, \quad (42)$$

where V and Λ_k are defined in Theorem 2.4. Once a “good” iterate X^j is at hand after some iterations during trace-penalty minimization, the relationship (42) implies that λ_k can be estimated from the maximum eigenvalues of $I - (X^j)^T X^j$, i.e., $\bar{\lambda}_k^j = \mu \lambda_{\max}(I - (X^j)^T X^j)$. Since the computational cost of approximating the largest eigenvalue of a $k \times k$ matrix is relatively low, the penalty parameter μ can be updated during trace-penalty minimization by the formula

$$\mu = \max(c_1, c_2 \bar{\lambda}_k^j). \quad (43)$$

A more accurate estimate of λ_k becomes available after an RR step is executed and the k -th Ritz-value θ_k is at hand. Then we use the formula

$$\mu = \max(c_1, c_2 \theta_k). \quad (44)$$

In favorable cases where the gap between λ_k and λ_{k+1} is relatively large, our strategy of choosing μ slightly larger than λ_k would have a good chance to satisfy both $\mu > 0$ and $\mu \in (\lambda_k, \lambda_{k+1})$, provided that $\lambda_k > 0$. In order to ensure $\lambda_k > 0$, our current strategy is to first scale the matrix A by $\sigma \approx |\lambda_1|$, assuming

that $\lambda_1 < 0$, and then add a positive shift $\omega > 1$, obtaining

$$\hat{A} = \frac{1}{\sigma} A + \omega I \quad (45)$$

that is at least close to being positive semidefinite. After performing trace-penalty minimization to $\hat{A}$, the above scale and shift can be easily reversed to recover the eigenvalues of A .

To estimate λ_1 , we note that the well-known Gershgorin circle theorem implies that

$$\lambda_1 \geq u_1 := \min_{i=1,\dots,n} \left\{ A_{ii} - \sum_{j \neq i} |A_{ij}| \right\}.$$

In addition, the relationship between matrix norms implies that

$$u_2 := \frac{\max(\|A\|_\infty, \|A\|_F)}{\sqrt{n}} \leq \|A\|_2 = \max(|\lambda_1|, |\lambda_n|).$$

Hence, without too much computation a reasonable value of σ in (45) is taken as

$$\sigma = \max(\min(|u_1|, u_2), 1). \quad (46)$$

As long as σ is not much smaller than $|\lambda_1|$, setting ω to a moderate number between 1 to 10 usually works well in our tests. In our numerical experiments, we always take the safe value of $\omega = 10$.

4 Numerical Experiments

In this section, we test the performance of EigPen as a general solver for computing a set of smallest eigenvalues and their corresponding eigenvectors of sparse matrices.

4.1 Solvers, Test Matrices and Platform

We choose to compare EigPen with the implicitly restarted Lanczos method in ARPACK¹ and the locally optimal preconditioned conjugate gradient (LOBPCG) algorithm. All algorithms are implemented in Fortran and parallelized by using OpenMP except that preconditioning for EigPen is demonstrated in MATLAB. We use an implementation of LOBPCG, previously developed by the second author of this paper, instead of the BLOPEX² package since the performance of BLOPEX seems not quite as stable and efficient as our own version on our test platform.

We select a set of thirteen test matrices arising from the density functional theory (DFT) for electronic structure calculation. They are all sparse matrices³ whose dimension n , the number of nonzero components

¹Downloadable from <http://www.caam.rice.edu/software/ARPACK>

²Downloadable from <http://code.google.com/p/blopex>

³Downloadable from <http://www.cise.ufl.edu/research/sparse/matrices>

nnz and sparsity are listed in Table 1. Many of them are produced by PARSEC [11], a real space DFT based code in which the Hamiltonian is discretized by using finite difference.

Table 1: Problem characteristics

Name	n	nnz	$100 \times \frac{nnz}{n^2} \%$
Andrews	60000	410077	0.01%
C60	17576	212390	0.07%
c_65	48066	204247	0.01%
cfdl	70656	948118	0.02%
finance	74752	335872	0.01%
Ga10As10H30	113081	3114357	0.02%
Ga3As3H12	61349	3016148	0.08%
OPF3754	15435	82231	0.03%
shallow_water1	81920	204800	<0.01%
Si10H16	17077	446500	0.15%
Si5H12	19896	379247	0.10%
SiO	33401	675528	0.06%
wathen100	30401	251001	0.03%

We perform most of our numerical experiments on a single node of Hopper⁴, a Cray XE6 supercomputer maintained at the National Energy Research Scientific Computer Center (NERSC) in Berkeley. The node consists of two twelve-core AMD “MagnyCours” 2.1-GHz processors with a total of 32 gigabyte (GB) shared memory. However, memory access bandwidth and latency are nonuniform across all cores. Each core has its own 64 kilobytes (KB) L1 and 512 KB L2 caches. One 6-MB L3 cache shared among 6 cores on the Magny-Cours processor. There are four DDR3 1333-MHz memory channels per twelve-core “MagnyCours” processor.

We use the multi-threaded version of the Cray Scientific Libraries package, LibSci, which includes multi-threaded versions BLAS and LAPACK subroutines optimized for Cray XE6. Each sparse matrix vector multiplication (SpMV) in ARPACK is parallelized through OpenMP threads. While each individual SpMV is not parallelized in EigPen and LOBPCG, a loop level parallelization is applied to the loop that produces the columns of the product AX where X contains multiple columns. We also implement a block version of the Davidson algorithm. Without using a preconditioner, the algorithm is essentially a steepest descent algorithm directly applied to (1). Because its performance in our tests has been found to be clearly poorer compared to other algorithms discussed in this section, we do not include its timing measurements in the numerical results presented in this section. The block Krylov-Schur algorithm [21] and the (block) Chebyshev-Davidson algorithm [19, 20] are not included in our comparison, partly because suitable Fortran/OpenMP implementations of these algorithms were unavailable for our tests.

4.2 Termination Rules and EigPen Parameters

All tests on the aforementioned machine Hopper are run as batch jobs with a maximum wall clock time limit of 6 hours. We terminate all algorithms when the relative residual norm (defined below) for every Ritz-pair

⁴Detailed information is available at <http://www.nersc.gov/users/computational-systems/hopper/>

(u_i, θ_i) is smaller than a prescribed tolerance tol , that is,

$$\text{res}_i(U) = \frac{\|Au_i - \theta_i u_i\|_2}{\max(1, |\theta_i|)} \leq tol, \quad i = 1, \dots, nev, \quad (47)$$

where nev is the number of smallest eigenvalues to be computed, u_i is the i -th column of U that satisfies $U^T U = I$, and $\theta_i = u_i^T A u_i$ (recall that for EigPen we need to perform an RR step to obtain the Ritz-pair). We also terminate an algorithm when the number of iterations reaches a maximum of 10,000, but this limit was never reached in our experiments.

In EigPen, the initial penalty parameter μ is computed by (41) and it is updated by (43) at most three times in the first outer iteration of EigPen. After an RR step is executed, μ is set according to (44) and is fixed throughout the next round of inner iterations. The constants $c_1 = 0.1$ and $c_2 = 1.1$ are used in (41), (43) and (44). The initial tolerance ϵ^0 is set to tol and the backtracking constant δ is set to 0.25. The parameter δ_ϵ is adjusted dynamically according to the number of the converged eigenvectors (denoted by k_1) that satisfy the condition $\text{res}_i \leq tol$, for $i = 1, \dots, nev$, after each RR step:

$$\delta_\epsilon = \begin{cases} 0.1, & \text{if } k_1 = 0, \\ 0.5, & \text{if } k_1 \leq 0.9 \text{ } nev, \\ 0.6, & \text{if } k_1 \leq 0.95 \text{ } nev, \\ 0.7, & \text{otherwise.} \end{cases}$$

As is already mentioned, in order to facilitate the selection of our penalty parameter μ in EigPen we perform scaling and shifting as in (45) where σ is given by (46) and $\omega = 10$ is always used.

4.3 Overall Performance

We first report the overall performance of ARPACK, LOBPCG and EigPen on the test matrices listed in Table 1. The number of smallest eigenvalues (nev) to be computed is roughly 1% of the dimension of A . All algorithms are run in parallel with 24 cores. No preconditioner is used in these tests.

For LOBPCG and EigPen, we set the dimension of X (denoted by k) to be slightly larger than nev to improve the convergence. Specifically, k is set to $nev \times 1.1$ (round to the nearest integer). For ARPACK, we set the parameter $ncv = nev + 100$, which is the dimension of the Krylov subspace constructed to extract desired eigenvalue approximations. The number 100 is simply the degree of the polynomial constructed implicitly in each restart to filter the unwanted spectral components from the starting vector. Setting nev to a larger value tends to reduce the number of restarts, but making each restart more costly. The optimal value ncv that achieves the best tradeoff between the number of restarts and the amount of work per restart is generally difficult to determine a priori.

Our experiments are performed using two different tolerance values $tol = 10^{-2}$ and $tol = 10^{-4}$. The total wall clock times taken by the three solver are presented in Table 2. Whenever the code terminates abnormally (either the run is stopped prematurely or the maximum wall clock time limit of 6 hours is

reached), the corresponding entry in the table is marked by “--”. From Table 2, we observe that ARPACK did not succeed on matrices c_65, cfd1 and Ga10As10H30 for both tolerance values, and LOBPCG did not succeed on matrices c_65, Ga10As10H30 and Ga3As3H12 for $tol = 10^{-4}$. Even when it converges, on larger problems ARPACK typically takes longer to run than either EigPen or LOBPCG (which may be attributable to a limited parallel scalability in sparse matrix-vector multiplications). In general, EigPen is faster than LOBPCG, especially on larger problems, with only one significant exception on the matrix c_65 for $tol = 10^{-2}$.

Table 2: A comparison of total wall clock time (“--” are abnormal terminations)

Matrix	nev	$tol = 10^{-2}$			$tol = 10^{-4}$		
		ARPACK	LOBPCG	EigPen	ARPACK	LOBPCG	EigPen
Andrews	600	2956	575	159	3344	1160	496
C60	200	57	29	19	59	52	44
c_65	500	--	331	3099	--	--	10030
cfd1	700	--	815	233	--	2883	1547
finance	700	9903	968	472	16120	4629	1122
Ga10As10H30	1000	--	5390	1848	--	--	5531
Ga3As3H12	600	4871	771	563	6587	--	1600
OPF3754	200	23	8	10	23	28	17
shallow_water1	800	2528	642	215	18590	3849	951
Si10H16	200	73	77	24	78	100	86
Si5H12	200	103	86	24	114	114	38
SiO	400	789	265	81	840	1534	287
wathen100	300	828	219	89	869	1103	219

The minimal, average and maximal number of the RR steps performed by EigPen in this set of tests is, respectively, 3, 5 and 9. As will be shown later, the cost of the RR steps only accounts for a very small portion of the total cost of EigPen.

We next show the accuracy of the computed eigenpairs, as well as that of the computed minimum trace values. We should point out that when tol is relatively large, the i -th Ritz value θ_i may be closer to λ_j for $j > i$ than to λ_i . In this case, we may miss some eigenvalues even though the convergence criterion (47) is satisfied for all $i \leq nev$. To measure the accuracy of the computation, we compute the relative difference between θ_i and the true eigenvalue λ_i computed in advance by ScaLAPACK [4]. The maximum relative errors among all eigenvalues, which is measured by

$$\text{err}_\theta = \max_{i=1,\dots,nev} \frac{|\theta_i - \lambda_i|}{\max(1, |\lambda_i|)},$$

are reported in Table 3, and the relative errors between the sum of the nev eigenvalues, defined by

$$\text{err}_{\text{trace}} = \frac{|\sum_{i=1}^{nev} \theta_i - \sum_{i=1}^{nev} \lambda_i|}{\max(1, |\sum_{i=1}^{nev} \lambda_i|)},$$

are presented in Table 4. From these tables, we see that LOBPCG and EigPen achieve the same level of accuracy on most problems. Compared with the other two solvers, the accuracy of ARPACK is worse on most matrices when $tol = 10^{-2}$ and somewhat better on about half of the matrices when $tol = 10^{-4}$.

To measure the accuracy of the approximate eigenvectors, we also report the maximum residual error

Table 3: A comparison of err_θ among different solvers

Matrix	$\text{tol} = 10^{-2}$			$\text{tol} = 10^{-4}$		
	ARPACK	LOBPCG	EigPen	ARPACK	LOBPCG	EigPen
Andrews	1.51e-02	2.78e-04	2.31e-03	4.54e-06	4.58e-05	4.58e-05
C60	4.48e-06	1.78e-04	5.88e-04	4.48e-06	4.34e-05	4.34e-05
c_65	--	2.43e-04	1.52e-03	--	--	4.79e-05
cfdl	--	2.72e-03	6.33e-03	--	5.37e-07	3.90e-06
finance	5.19e-02	1.28e-03	4.78e-03	7.59e-05	4.80e-05	4.80e-05
Ga10As10H30	--	2.98e-04	3.83e-04	--	--	4.97e-05
Ga3As3H12	3.02e-02	1.70e-04	9.15e-04	5.50e-03	--	4.69e-05
OPF3754	3.59e-06	6.74e-04	8.80e-04	3.59e-06	2.22e-05	2.22e-05
shallow_water1	3.96e-01	5.75e-03	5.80e-03	3.85e-04	8.64e-06	8.42e-06
Si10H16	2.83e-02	5.01e-05	9.77e-05	2.53e-02	4.33e-05	4.33e-05
Si5H12	5.52e-02	6.11e-05	3.35e-04	4.38e-06	3.86e-05	3.86e-05
SiO	2.40e-02	9.14e-05	1.42e-03	4.43e-06	4.81e-05	4.81e-05
wathen100	5.27e-03	5.74e-05	9.11e-04	3.93e-06	3.17e-05	3.17e-05

Table 4: A comparison of $\text{err}_{\text{trace}}$ among different solvers

Matrix	$\text{tol} = 10^{-2}$			$\text{tol} = 10^{-4}$		
	ARPACK	LOBPCG	EigPen	ARPACK	LOBPCG	EigPen
Andrews	1.01e-03	1.48e-05	1.06e-04	5.87e-09	1.07e-07	1.07e-07
C60	1.72e-07	4.20e-06	7.48e-06	1.72e-07	9.01e-08	9.01e-08
c_65	--	5.73e-06	2.98e-05	--	--	8.20e-07
cfdl	--	2.36e-01	5.37e-01	--	1.43e-06	1.11e-05
finance	8.13e-03	1.18e-04	4.99e-04	2.35e-07	3.91e-07	3.91e-07
Ga10As10H30	--	1.33e-05	1.50e-05	--	--	3.10e-07
Ga3As3H12	1.82e-03	8.57e-06	5.27e-05	6.38e-05	--	1.01e-06
OPF3754	3.47e-09	4.87e-05	1.77e-05	3.47e-09	8.84e-08	8.84e-08
shallow_water1	1.30e-01	2.38e-03	1.77e-03	2.03e-05	1.70e-07	1.49e-07
Si10H16	2.97e-03	7.60e-06	1.97e-06	1.91e-03	3.16e-06	3.16e-06
Si5H12	1.58e-03	7.62e-06	1.88e-05	2.12e-07	5.50e-07	5.50e-07
SiO	7.61e-04	6.01e-06	3.89e-05	1.72e-07	1.29e-06	1.29e-06
wathen100	2.66e-04	1.18e-05	2.43e-05	6.94e-08	3.05e-07	3.05e-07

defined by

$$\text{err}_{\text{res}} = \max_{i=1, \dots, nev} \text{res}_i$$

in Table 5. We observe that EigPen usually returns a slightly smaller residual error than LOBPCG in this set of tests.

4.4 Performance profile and dependency on eigenspace dimension

In this subsection, we examine how LOBPCG and EigPen perform when the number of desired eigenpairs (nev) increases. For brevity, we only show results for two matrices Andrews and Ga3As3H12, but similar profiles can be observed for other matrices as well. We exclude ARPACK from this comparison because ARPACK is not as competitive as either LOBPCG or EigPen for relatively large values of nev , as is shown by the results of the previous subsection.

In the following experiments, we set the convergence tolerance to $\text{tol} = 10^{-2}$ and nev to a set of increasing values $\{500, 1000, 1500, 2000, 2500, 3000\}$. We run both solvers on 24 cores. The wall clock time measurements are plotted against nev for both solvers in Figure 1. We observe that in this test EigPen always takes a less amount of time to run than LOBPCG does. The difference in wall clock time increases quickly as nev increases. This observation suggests that the benefit of using EigPen become increasingly greater

Table 5: A comparison of err_{res} among different solvers

Matrix	$\text{tol} = 10^{-2}$			$\text{tol} = 10^{-4}$		
	ARPACK	LOBPCG	EigPen	ARPACK	LOBPCG	EigPen
Andrews	9.10e-03	9.58e-03	9.30e-03	3.96e-05	9.20e-05	3.14e-05
C60	1.77e-04	9.83e-03	5.23e-03	8.67e-07	9.31e-05	7.59e-05
c_65	--	9.60e-03	9.72e-03	--	--	7.22e-05
cfd1	--	9.50e-03	6.22e-03	--	9.80e-05	5.84e-05
finance	9.43e-03	9.94e-03	8.23e-03	5.86e-05	9.96e-05	7.29e-05
Ga10As10H30	--	9.97e-03	5.99e-03	--	--	2.72e-05
Ga3As3H12	7.92e-03	8.61e-03	8.79e-03	7.43e-05	--	3.73e-05
OPF3754	1.19e-04	9.21e-03	5.34e-03	8.21e-05	4.96e-05	7.55e-05
shallow_water1	1.00e-02	9.90e-03	6.70e-03	8.90e-05	9.61e-05	3.96e-05
Si10H16	2.44e-03	8.90e-03	3.44e-03	1.45e-05	9.01e-05	6.54e-05
Si5H12	1.99e-03	9.56e-03	6.51e-03	4.56e-05	9.78e-05	8.98e-05
SiO	1.37e-03	1.00e-02	9.11e-03	3.28e-06	9.46e-05	1.03e-05
wathen100	9.96e-03	9.60e-03	6.22e-03	1.13e-05	7.72e-05	9.95e-05

as nev becomes larger. The key reason that EigPen performs much better than LOBPCG for large nev is that, by performing far fewer RR steps, it is able to leverage BLAS3 operations that are highly optimized for Hopper (and other high performance computers).

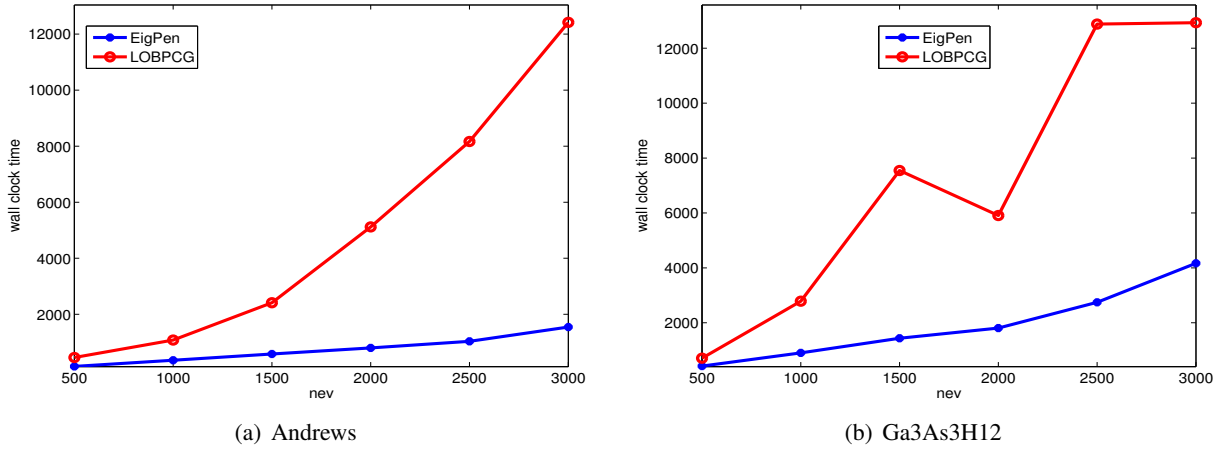


Figure 1: A comparison of wall clock times by LOBPCG and EigPen to compute nev eigenpairs of the matrices Andrews and Ga3As3H12 as nev increases.

In Figure 2, we show run times of four categories: sparse matrix vector multiplications (SpMV), dense matrix-matrix operations (BLAS3, the `DGEMM`, `DSYRK`, `DPOTRF`, `DTRSM` subroutines in BLAS and LAPACK), Rayleigh-Ritz (RR, the `DSYGVD`, `DSYEVD`, `DGESVD` subroutines in LAPACK) calculations, and matrix copying (the `DLACPY` subroutine in LAPACK). These are the major computational components of both EigPen and LOBPCG, albeit in different proportions. Two clarifications are in order here. Firstly, we categorize these subroutines only at the highest solver level. As such, any call to `DGEMM` inside the subroutine `DSYEVD`, for example, is not counted as in the BLAS3 category. Secondly, although the “correctness” of such a classification scheme may be debatable, it does not at alter the overall fact, as is clearly shown by our computational results, that the category BLAS3 is much more scalable than the category RR on our test platform.

The run time of each category is measured in terms of the percentage of wall clock time spent in that

category over the total wall clock time. We can clearly see that for EigPen the run time of BLAS3 dominates the entire computation in almost all cases. The BLAS3 time increases steadily as nev increases from 500 to 3000, while the SpMV time decreases steadily. The run time of RR is negligible. However, since our implementation of EigPen performs extra matrix copying when computing the gradient difference Y^j defined in (31) for the BB step size computation, the cost associated with such data movement is notable. In LOBPCG, the relative cost of SpMV also decreases as nev increase. However, the run time of RR increases rapidly as nev increases. When $nev \geq 1500$, the run time of RR is higher than that of BLAS3. Note that the RR time for LOBPCG seems out of proportion when nev is equal to 1000 and 1500 for the matrix Ga3As3H12. The reason is that we have to perform a few singular value decompositions (SVD) to repair a rank deficient basis in LOBPCG and the cost of SVD is counted in RR. The rank deficiency is caused by eigenvalues clusters nears the 1000th and the 1500th eigenvalues.

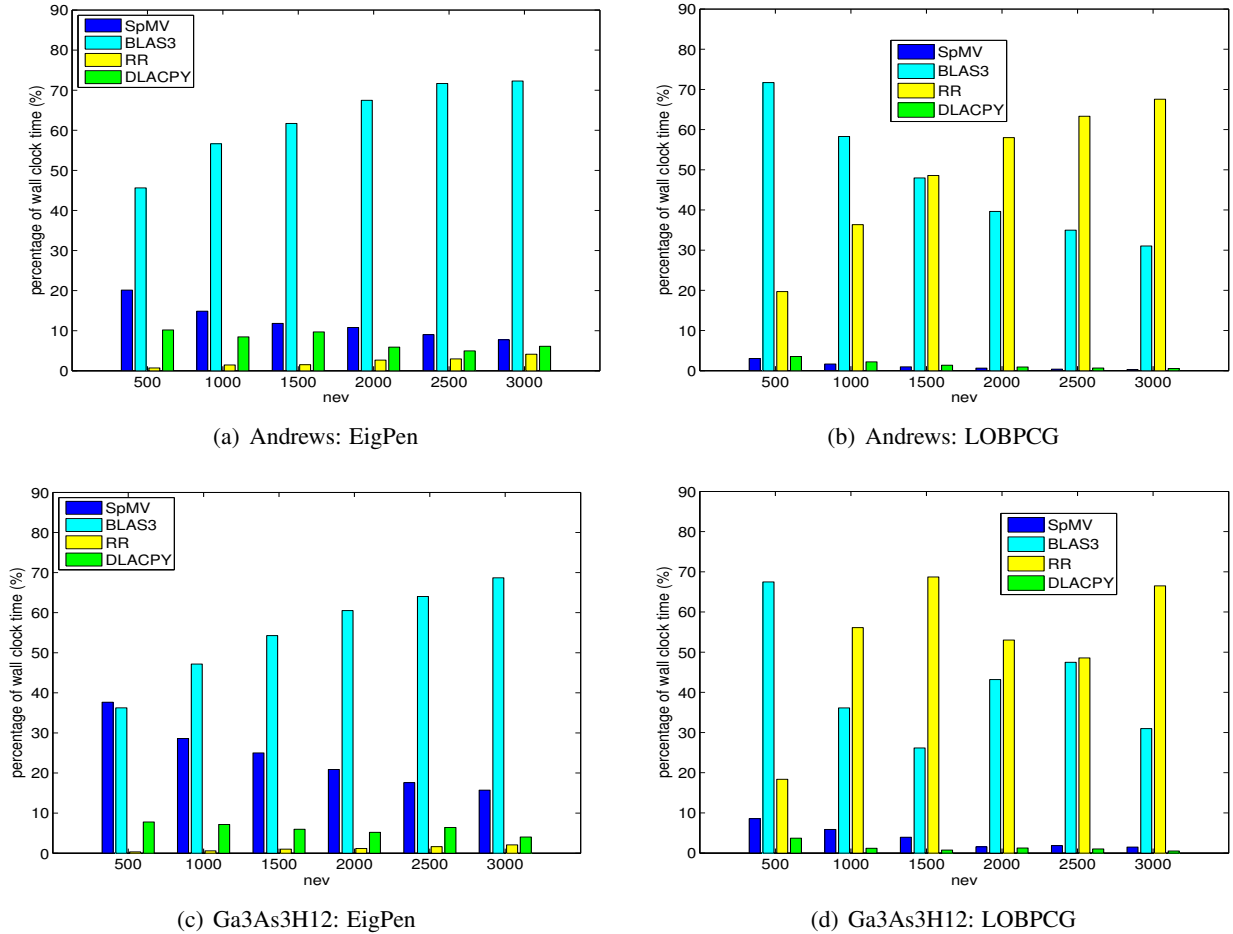


Figure 2: A comparison of timing profile between EigPen and LOBPCG.

Figures 1 and 2 clearly demonstrate that the advantages of the EigPen algorithm are due to fewer Rayleigh-Ritz calculations. This advantage is more pronounced when the number of eigenpairs to be computed (nev) is large because the cost of Rayleigh-Ritz calculation grows rapidly with respect to nev (and

k). Although the complexity of the Rayleigh-Ritz calculation is the same as that associated with the dense matrix-matrix operations required for updating the approximate solution in EigPen, dense matrix-matrix operations can be implemented efficiently on modern high performance parallel computers whereas it is more difficult to achieve the same level of efficiency for RR calculations. As a result, by keeping the number of Rayleigh-Ritz calculations small in EigPen and making use of more BLAS3 operations, we can make it more efficient than LOBPCG for large nev values.

4.5 Parallel scalability

In this subsection, we examine parallel scalability of LOBPCG and EigPen. For brevity, we again only show results for the Andrews and Ga3As3H12 matrices, although similar results can be seen for other test problems as well. We define the speedup factor for running a code on p cores as

$$\text{speedup-factor}(p) = \frac{\text{wall clock time for a single core run}}{\text{wall clock time for a } p\text{-core run}}.$$

We set $nev = 1500$ and only run 5 iterations for each solver since the speedup factor does not change if more iterations are performed.

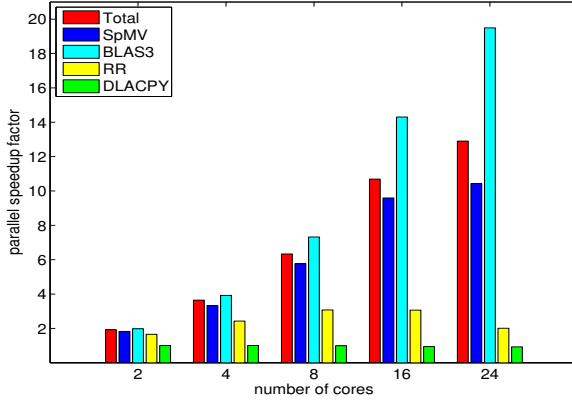
Figure 3 shows the speedup factors associated with SpMV, BLAS3, RR and DLACPY, as well as the overall computation, when the parallelized Fortran codes are run with 2, 4, 8, 16 and 24 cores. As we can clearly see from the figure that the speedup factors for BLAS3 are nearly perfect when these codes are run on as many as 24 cores. The scalability of SpMV is almost as good for the Ga3As3H12 problem. But it is slightly worse beyond 8 cores for the Andrews matrix, which we believe is due to a higher sparsity of the Andrews matrix that makes the effect of thread overhead more prominent in parallel SpMV calculations. However, the speedup factor for RR increases slowly with respect to the number of cores up to 8 cores, then it starts to decrease. Because computation in EigPen is heavily dominated by BLAS3 (and to a lesser extent by SpMV) that scales much better than RR, the overall scalability of EigPen is better than that of LOBPCG.

4.6 Preconditioning for EigPen

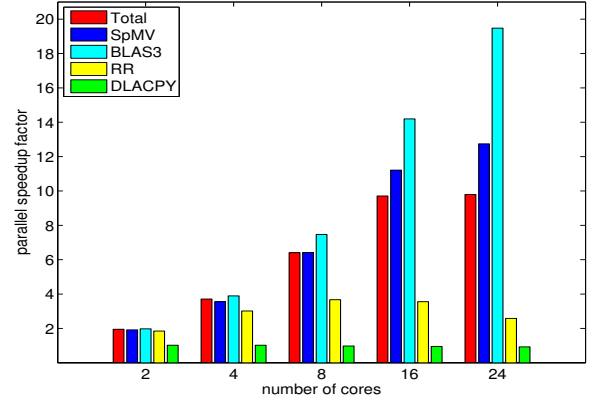
One can also introduce a preconditioner in EigPen similar to LOBPCG. The use of a preconditioner essentially amounts to a change of variable in the form of $Y = LX$. If an appropriate nonsingular L is chosen, substituting $X = L^{-1}Y$ into (2) yields a problem with a better conditioned Hessian. Let $M = L^T L$ be a preconditioner. It is not difficult to show that a preconditioned gradient method can be described by

$$X^{j+1} = X^j - \alpha^j M^{-1} \nabla f_\mu(X^j). \quad (48)$$

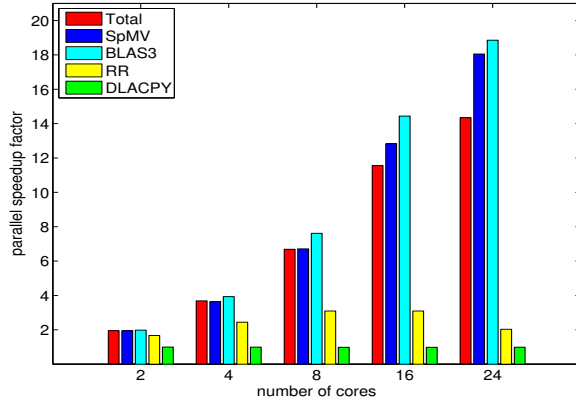
We now demonstrate that the performance of EigPen can be improved by preconditioning using the matrix “c_65” as an example. Recall from Table 2 this problem is relatively difficult to be solved by EigPen without using a preconditioner. The following experiment is performed in MATLAB on a Dell Precision M4700 workstation with Intel i7-3720QM CPU at 2.60GHz ($\times 8$) and 16GB of memory running



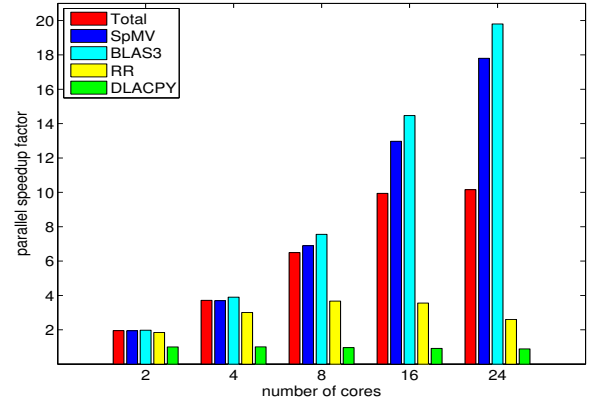
(a) Andrews: EigPen



(b) Andrews: LOBPCG



(c) Ga3As3H12: EigPen



(d) Ga3As3H12: LOBPCG

Figure 3: A comparison of timing profile between LOBPCG and EigPen.

Ubuntu 12.04 and MATLAB 2011b. We choose to use MATLAB because it is relatively easy to construct a preconditioner and perform $M^{-1}\nabla f_\mu(X^j)$ in MATLAB.

We computed $nev = 500$ eigenpairs with the tolerance $tol = 10^{-2}$ by using the same RR restart Algorithm 2 as in subsection 4.3. The preconditioner is computed by performing an incomplete Cholesky factorization of the matrix “c_65” using the command

$$L = \text{ichol}(A, \text{opts}),$$

where $\text{opts} = \text{struct}('type', 'ict', 'droptol', 1e-2, 'michol', 'on')$. Without preconditioning, EigPen consumed 4700 seconds and performed a total number of 3389 gradient evaluations. With the preconditioner, the wall clock time went down to 2339 seconds and the total number of gradient evaluations decreased to 1375.

In order to see the effect of preconditioning clearly, we show the iteration history of the gradient norm $\{\|\nabla f_\mu(X^j)\|_F\}$ computed by Algorithm 1 using the unpreconditioned scheme (27) and the preconditioned scheme (48), respectively. The results are depicted in Figure 4. It is clear that preconditioning can improve

the performance of EigPen significantly.

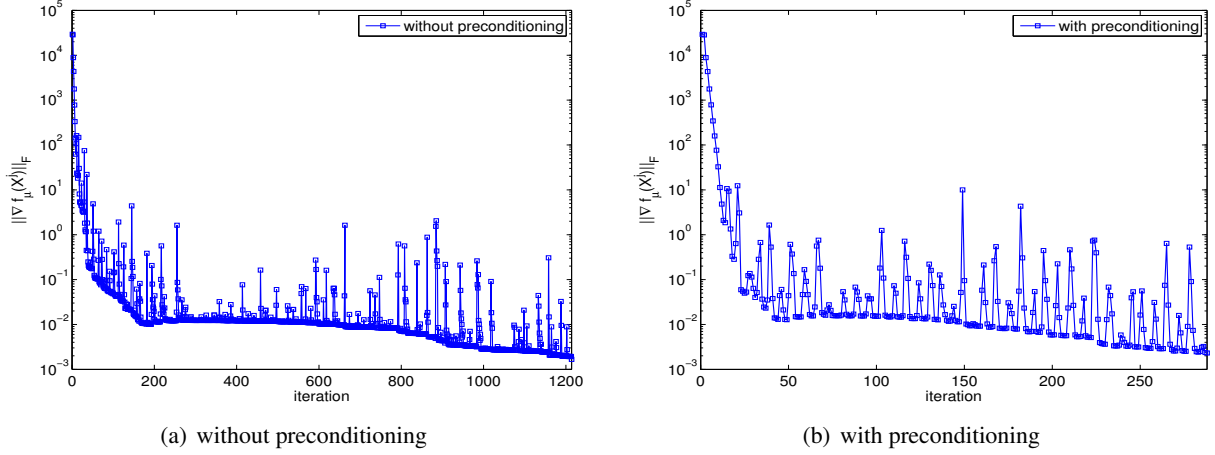


Figure 4: The gradient norm $\|\nabla f_\mu(X^j)\|_F$ versus iteration on c_{65} matrix without/with preconditioning.

We should point out that it is generally not easy to identify an effective and efficient preconditioner for an eigenvalue problem. An ideal preconditioner should be close to A^{-1} so that the first term of the Hessian (6) associated with the transformed problem becomes well conditioned. Yet, the pre-conditioner itself should not be too ill-conditioned; otherwise, the subsequent terms in (6) may become ill-conditioned, possibly making the entire Hessian ill-conditioned. The purpose of presenting the above example is not to promote a particular pre-conditioner, but rather to demonstrate the fact that EigPen can indeed take advantage of a good pre-conditioner whenever it is available.

5 Conclusion

We propose and study a quadratic penalty approach for large-scale eigenspace computation. Our analysis on stationarity of the penalty function establishes, under suitable conditions, not only an equivalence to the original eigenvalue problem in terms of the optimal invariant subspace, but also a desirable property of having fewer saddle points. It appears that our analysis and usage of quadratic penalty functions for eigenspace computation is new. A gradient-type method is then developed in which the number of Rayleigh-Ritz (RR) steps, including orthogonalization, is greatly reduced in exchange for BLAS3-rich operations. Hence, our method has the potential to take advantage of tens or hundreds of thousands-way concurrency on modern multi/many-core systems.

Numerical experiments, based on a Fortran implementation using OpenMP, are conducted on a parallel computer to evaluate the performance of our new eigensolver EigPen in comparison to a few state-of-the-art solvers such as LOBPCG. In our numerical experiments (where preconditioning is not used to avoid complications), EigPen generally outperforms LOBPCG in wall clock time, especially as the dimension of the computed eigenspace increases. This trend in favor of EigPen can be explained as follows. As the eigenspace dimension increases, the computation in EigPen is increasingly dominated by dense matrix-

matrix operations, but in LOBPCG it is increasingly dominated by the Rayleigh-Ritz procedure including orthogonalization. Clearly, the former is more scalable than the latter on modern high performance computers.

The performance of EigPen can certainly be improved in several aspects, such as speeding up convergence or improving accuracy, with the help of a number of techniques or of Hessian information. Candidate techniques may include subspace optimization, preconditioning, deflation and polynimail filtering (in fact we have tried some of them but did not include them in our numerical comparison for simplicity). A verification of the parallel efficiency of EigPen using the Message Passing Interface (MPI) is also highly expected.

Acknowledgements

Z. Wen would like to thank Prof. Michael Ulbrich for hosting his visit at Technische Universität München. X. Liu would like to thank Prof. Yuhong Dai for discussing nonlinear programming techniques for eigenvalue computation.

References

- [1] H. M. AKTULGA, L. LIN, C. HAINE, E. G. NG, AND C. YANG, *Parallel Eigenvalue Calculation based on Multiple Shift-invert Lanczos and Contour Integral based Spectral Projection Method*, LBNL Tech Report, Nov 2012.
- [2] E. ANDERSON, Z. BAI, J. DONGARRA, A. GREENBAUM, A. MCKENNEY, J. DU CROZ, S. HAMMERLING, J. DEMMEL, C. BISCHOF, AND D. SORENSEN, *Lapack: a portable linear algebra library for high-performance computers*, in Proceedings of the 1990 ACM/IEEE conference on Supercomputing, Supercomputing '90, IEEE Computer Society Press, 1990, pp. 2–11.
- [3] J. BARZILAI AND J. M. BORWEIN, *Two-point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.
- [4] L. S. BLACKFORD, J. CHOI, A. CLEARY, E. D’AZEVEDO, J. DEMMEL, I. DHILLON, S. HAMMARLING, G. HENRY, A. PETITET, K. STANLEY, D. WALKER, AND R. C. WHALEY, *ScaLAPACK user’s guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.
- [5] R. CHELIKOWSKY AND Y. SAAD, *A spectrum slicing method for the Kohn-Sham problem*, Comp. Phys. Comm., 183 (2012), p. 497.
- [6] R. COURANT, *Variational methods for the solution of problems of equilibrium and vibrations*, Bull. Amer. Math. Soc., 49 (1943), pp. 1–23.
- [7] Y. H. DAI, *On the nonmonotone line search*, J. Optim. Theory Appl., 112 (2002), pp. 315–330.

- [8] H. R. FANG AND Y. SAAD, *A filtered Lanczos procedure for extreme and interior eigenvalue problems*, SIAM J. Sci. Comp., 34 (2013), pp. A2220–A2246.
- [9] L. GRIPPO, F. LAMPARIELLO, AND S. LUCIDI, *A nonmonotone line search technique for Newton's method*, SIAM J. Numer. Anal., 23 (1986), pp. 707–716.
- [10] A. V. KNYAZEV, *Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.
- [11] L. KRONIK, A. MAKMAL, M. TIAGO, M. M. G. ALEMANY, X. HUANG, Y. SAAD, AND J. R. CHELIKOWSKY, *PARSEC – the pseudopotential algorithm for real-space electronic structure calculations: recent advances and novel applications to nanostructures*, Phys. Stat. Solidi. (b), 243 (2006), pp. 1063–1079.
- [12] J. NOCEDAL AND S. J. WRIGHT, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer, New York, second ed., 2006.
- [13] B. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, 1980.
- [14] Y. SAAD, J. R. CHELIKOWSKY, AND S. M. SHONTZ, *Numerical methods for electronic structure calculations of materials*, SIAM Rev., 52 (2010), pp. 3–54.
- [15] W. SUN AND Y. YUAN, *Optimization Theory and Methods: Nonlinear Programming*, Springer, New York, 2006.
- [16] Y.-X. YUAN, *A new stepsize for the steepest descent method*, J. Comput. Math., 24 (2006), pp. 149–156.
- [17] ———, *Step-sizes for the gradient method*, in Third International Congress of Chinese Mathematicians. Part 1, 2, vol. 2 of AMS/IP Stud. Adv. Math., 42, pt. 1, Amer. Math. Soc., 2008, pp. 785–796.
- [18] H. ZHANG AND W. W. HAGER, *A nonmonotone line search technique and its application to unconstrained optimization*, SIAM J. Optim., 14 (2004), pp. 1043–1056.
- [19] Y. ZHOU, *A block Chebyshev-Davidson method with inner-outer restart for large eigenvalue problems*, J. Comput. Phys., 229 (2010), pp. 9188–9200.
- [20] Y. ZHOU AND Y. SAAD, *A ChebyshevDavidson algorithm for large symmetric eigenproblems*, SIAM J. Matrix Anal. and Appl., 29 (2007), pp. 954–971.
- [21] ———, *Block krylovschur method for large symmetric eigenvalue problems*, Numerical Algorithms, 47 (2008), pp. 341–359.