Accelerating Certifiable Estimation with Preconditioned Eigensolvers

David M. Rosen

Abstract—Convex (specifically semidefinite) relaxation provides a powerful approach to constructing robust machine perception systems, enabling the recovery of *certifiably globally optimal* solutions of challenging estimation problems in many practical settings. However, solving the large-scale semidefinite relaxations underpinning this approach remains a formidable computational challenge. A dominant cost in many state-of-the-art (Burer-Monteiro factorization-based) certifiable estimation methods is *solution verification* (testing the global optimality of a given candidate solution), which entails computing a minimum eigenpair of a certain symmetric *certificate matrix*. In this paper, we show how to significantly accelerate this verification step, and thereby the overall speed of certifiable estimation methods. First, we show that the certificate matrices arising in the Burer-Monteiro approach generically possess spectra that make the verification problem expensive to solve using standard iterative eigenvalue methods. We then show how to address this challenge using *preconditioned* eigensolvers; specifically, we design a specialized solution verification algorithm based upon the *locally optimal block preconditioned conjugate gradient* (LOBPCG) method together with a simple yet highly effective algebraic preconditioner. Experimental evaluation on a variety of simulated and real-world examples shows that our proposed verification scheme is very effective in practice, accelerating solution verification by up to 280x, and the overall Burer-Monteiro method by up to 16x, versus the standard Lanczos method when applied to relaxations derived from large-scale SLAM benchmarks.

I. INTRODUCTION

Many fundamental machine perception tasks require the solution of a high-dimensional nonconvex estimation problem; this class includes (for example) the fundamental problems of simultaneous localization and mapping (SLAM) in robotics, 3D reconstruction (in computer vision), and sensor network localization (in distributed sensing), among others. Such problems are known to be computationally hard to solve in general [1], with many local minima that can entrap the smooth local optimization methods commonly applied to solve them [2]. The result is that traditional machine perception algorithms (based upon *local* optimization) can be surprisingly brittle, often returning egregiously wrong answers even when the problem to which they are applied is well-posed.

Recent work has shown that *convex* (specifically *semidefinite*) *relaxation* provides a powerful alternative to (heuristic) local search for building highly-reliable machine perception systems, enabling the recovery of *provably globally optimal* solutions of generally-intractable estimation problems in many practical settings (cf. [1] generally). However, despite the great promise of this approach, solving the resulting semidefinite relaxations remains a significant computational challenge [3].

One of the most successful strategies for solving the large-scale semidefinite relaxations arising in machine perception applications is *Burer-Monteiro factorization* [4]. In brief, this method searches for a solution to a (high-dimensional)

semidefinite program by alternately applying fast *local* optimization to a surrogate nonconvex (but low-dimensional) problem, and then testing whether the recovered critical point is a *global* optimum for the original SDP (which entails calculating a minimum eigenpair of a certain symmetric *certificate matrix* [cf. Sec. II]). Surprisingly, *verifying* a candidate solution is frequently the dominant cost in the Burer-Monteiro approach, often requiring an order of magnitude more computational effort (using standard iterative eigenvalue methods) than the entire nonlinear optimization required to *produce* it.

In this paper we show how to significantly accelerate this rate-limiting verification step, and thereby the overall speed of current state-of-the-art certifiable estimation methods. First, we show that the certificate matrices arising in the Burer-Monteiro approach generically possess spectra that make it expensive to compute a minimum eigenpair using standard iterative eigenvalue methods. Next, we show how to address this challenge using *preconditioned* eigensolvers; specifically, we design a specialized verification algorithm based upon the *locally optimal block preconditioned conjugate gradient* (LOBPCG) [5] method together with a simple yet highly effective algebraic preconditioner. Experimental evaluation on a variety of synthetic and real-world examples demonstrates that our proposed verification approach is very effective in practice, accelerating solution verification by up to 280x, and the overall Burer-Monteiro method by up to 16x, versus a standard Lanczos method when applied to relaxations derived from large-scale SLAM benchmarks.

II. SEMIDEFINITE OPTIMIZATION USING THE BURER-MONTEIRO METHOD

In this section we briefly review the Burer-Monteiro method for semidefinite optimization [4], which forms the algorithmic foundation of many state-of-the-art certifiable estimation methods (cf. [1] and the references therein). We begin with a brief review of semidefinite programs and a discussion of their computational cost. We then describe the Burer-Monteiro method, which provides a scalable approach to solving SDPs with *low-rank* solutions. Finally, we discuss the *verification problem* (Problem 1) arising in the Burer-Monteiro approach, and identify a key feature that makes this problem expensive to solve using standard iterative eigenvalue methods.

A. Semidefinite programs

Recall that a *semidefinite program* (SDP) is a convex optimization problem of the form [6]:

$$\min_{X \in \mathbb{S}_+^n} \operatorname{tr}(CX) \quad \text{s.t.} \quad \mathcal{A}(X) = b, \tag{1}$$

where $C \in \mathbb{S}^n$, $b \in \mathbb{R}^m$, and $\mathcal{A} \colon \mathbb{S}^n \to \mathbb{R}^m$ is a linear map:

$$\mathcal{A}(X)_i \triangleq \operatorname{tr}(A_i X) \quad \forall i \in [m] \tag{2}$$

parameterized by coefficient matrices $\{A_i\}_{i=1}^m \subset \mathbb{S}^n$. Problem (1) can in principle be solved efficiently (i.e. in polynomial time) using general-purpose (e.g. interior point) optimization methods [6, 7]. In practice, however, the high computational cost of storing and manipulating the $O(n^2)$ elements of the decision variable X prevents such straightforward approaches from scaling effectively to problems in which n is larger than a few thousand [3]. Unfortunately, typical instances of many important machine perception problems (e.g. SLAM) are often several orders of magnitude larger, placing them well beyond the reach of such general-purpose techniques.

B. Burer-Monteiro factorization

While problem (1) is expensive to solve *in general*, the *specific instances* of (1) arising as relaxations in certifiable estimation often admit *low-rank* solutions. Any such solution $X^* \in \mathbb{S}_+^n$ can be represented concisely as $X^* = Y^* Y^{*\mathsf{T}}$, where $Y^* \in \mathbb{R}^{n \times r}$ and $r = \operatorname{rank} X^* \ll n$.

In their seminal work, Burer and Monteiro [4] proposed to exploit the existence of such low-rank solutions by replacing the high-dimensional decision variable X in (1) with its rank-r symmetric factorization YY^T, producing the following (nonconvex) *rank-restricted* version of the problem:

$$\min_{Y \in \mathbb{R}^{n \times r}} \operatorname{tr}(CYY^\mathsf{T}) \quad \text{s.t.} \quad \mathcal{A}(YY^\mathsf{T}) = b. \tag{3}$$

This substitution has the effect of dramatically reducing the dimensionality of the search space of (3) versus (1), as well as rendering the positive-semidefiniteness constraint in (1) *redundant* (since $YY^\mathsf{T} \succeq 0$ by construction). Burer and Monteiro [4] thus proposed to search for minimizers X^* of (1) by attempting to recover their low-rank factors Y^* from (3) using fast *local* nonlinear programming methods [7].

C. Ensuring global optimality

Since the Burer-Monteiro approach entails searching for a *global* minimizer Y^* of the nonconvex problem (3), one might naturally wonder whether it is possible to *find* such global optima in practice. Remarkably, recent work has shown that this can in fact be done under quite general conditions, despite problem (3)'s nonconvexity [8, 9].

Suppose that we have applied a numerical optimization method to an instance of problem (3), and recovered a first-order stationary point $Y \in \mathbb{R}^{n \times r}$. Then either $X = YY^\mathsf{T}$ is a global minimizer of (1) [which we can determine by checking the necessary and sufficient KKT conditions for the convex program (1)], or X is suboptimal, in which case we would like to identify a direction of improvement for the low-rank factor Y. The following result establishes that checking the optimality of $X = YY^\mathsf{T}$, and constructing a direction of improvement (if necessary), can *both* be accomplished with a *single* minimum-eigenpair calculation:

Theorem 1 (Theorem 4 of [10]). *Assume that the semidefinite program* (1) *satisfies Slater's condition. Let* $Y \in \mathbb{R}^{n \times r}$ *be a KKT point of* (3) *that satisfies the linear independence*

constraint qualification, $\lambda \in \mathbb{R}^m$ *the corresponding Lagrange multiplier, and define:*

$$S \triangleq C + \sum_{i=1}^m \lambda_i A_i. \tag{4}$$

Then exactly one of the following two cases holds:
(i) $S \succeq 0$, *and* $X = YY^\mathsf{T}$ *is a global minimizer of* (1).
(ii) *There exists* $v \in \mathbb{R}^n$ *such that* $v^\mathsf{T} S v < 0$, *and in that case,* $Y_+ = \begin{pmatrix} Y & 0 \end{pmatrix} \in \mathbb{R}^{n \times (r+1)}$ *is a KKT point of* (3) *attaining the same objective value as* Y, *and* $\dot{Y}_+ = \begin{pmatrix} 0 & v \end{pmatrix} \in \mathbb{R}^{n \times (r+1)}$ *is a feasible second-order direction of descent from* Y_+.

In brief, Theorem 1 establishes that the global optimality of $X = YY^\mathsf{T}$ as a solution of (1) can be verified by checking whether the *certificate matrix* S defined in (4) is positive semidefinite, and in the event that it is *not*, that any direction of negative curvature $v \in \mathbb{R}^n$ of S provides a direction of improvement $\dot{Y}_+$ from the embedding Y_+ of the current critical point Y into a higher-dimensional instance of (3).

This theorem of the alternative suggests a natural algorithm for recovering minimizers X^* of (1) via a *sequence* of local optimizations of (3) [11, 12]. Starting at some (small) initial rank r, apply local optimization to (3) to obtain a stationary point Y. If $X = YY^\mathsf{T}$ is optimal for (1), then return Y; otherwise, increment the rank parameter r, and restart the local optimization for the next instance of (3) using the direction of improvement $\dot{Y}_+$ provided by Theorem 1.

D. The solution verification problem

Suprisingly, simply *verifying* the optimality of a candidate solution $X = YY^\mathsf{T}$ (by calculating a minimum eigenpair of the certificate matrix S in (4) and applying Theorem 1) is frequently the dominant cost in the Burer-Monteiro approach, often requiring an order of magnitude more computational effort than the entire nonlinear optimization (3) required to *produce* Y itself (cf. Sec. V). This turns out to be a consequence of a particular feature of the spectra of the certificate matrices S arising in the Burer-Monteiro method.

Remark 1 (The spectra of the certificate matrix). A straightforward calculation shows that the first-order stationarity conditions for (3) can be expressed as (cf. [10, Thm. 2]):

$$SY = 0. \tag{5}$$

This implies that S will have 0 as an eigenvalue of multiplicity *at least* $\operatorname{rank} Y$; that is, S always has a *cluster* of eigenvalues at 0. It is this eigenvalue clustering that makes calculating a minimum eigenpair of S expensive. In detail, if $S \succeq 0$, then $\lambda_{\min}(S) = 0$, so the target eigenvalue is embedded in a tight cluster, while if $\lambda_{\min}(S) < 0$ but has small magnitude, it is difficult to estimate accurately because the cluster of 0 eigenvalues nearby ensures that the relevant eigenvalue gap $\gamma \le |\lambda_{\min}(S) - 0| = |\lambda_{\min}(S)|$ is small. In effect, equation (5) guarantees that the minimum-eigenpair calculation is almost always poorly-conditioned, and therefore expensive to solve using standard iterative eigenvalue methods (cf. Sec. III).

In light of Theorem 1 and Remark 1, we are thus interested in developing more computationally efficient methods of solving the following (*solution*) *verification problem*:

Problem 1 (Verification problem)**.** Given a matrix $S \in \mathbb{S}^n$, determine whether $S \succeq 0$, and if it is not, calculate a vector $x \in \mathbb{R}^n$ such that $x^\mathsf{T} S x < 0$.

III. RELATED WORK

In this section we review prior work pertaining to the solution of the verification problem (Problem 1) from two areas: algorithms for the symmetric eigenvalue problem (Section III-A), and certifiable estimation (Section III-B).

A. Algorithms for the symmetric eigenvalue problem

Standard large-scale eigenvalue algorithms are *iterative projection methods*, meaning that they recover approximate eigenpairs of a matrix $A \in \mathbb{S}^n$ by searching for approximate eigenvectors within a (computationally tractable) low-dimensional subspace of $\mathbb{R}^n$ [13, Chp. 3]. This includes the *power* and *subspace iteration* methods (for computing *dominant* eigenpairs), and the *Lanczos method*, which is the preferred technique for computing a small number of *extremal* eigenpairs of A [13, Chp. 4]. An important feature of these methods is that their convergence rates depend crucially upon the *distribution* of A's eigenvalues, and in particular how *well-separated* a target eigenvalue is from the remainder of A's spectrum (relative to A's spectral norm). For example, letting $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_K$ denote A's distinct eigenvalues, the number of iterations required to estimate a minimum eigenvector x_1 of A using the Lanczos method is approximately $\mathcal{O}(\sqrt{\gamma / \|A\|})$, where $\gamma \triangleq \lambda_2 - \lambda_1$ is the *spectral gap* between A's minimum eigenvalue λ_1 and the next-largest eigenvalue λ_2 (cf. [14, Thm. 6.3] and the discussion in [15, Sec. 6.1]). Consequently, computing a minimum eigenpair of A using the Lanczos method is expensive when $\gamma \ll \|A\|$. Unfortunately, as shown in Section II-D, this case generically holds holds for the verification problems arising in the Burer-Monteiro approach.

Given the sensitivity of iterative eigenvalue methods to the distribution of A's spectrum, a natural idea for improving computational performance is to precondition the eigenvalue problem by applying a *spectral transformation* that sends the desired eigenvalue λ of A to a well-separated *extremal* eigenvalue [13, Sec. 3.3]. The most common such method is the *shift-and-invert* (SI) spectral transformation, in which the matrix A is replaced by the resolvent $R_\sigma \triangleq (A - \sigma I)^{-1}$ for $\sigma \notin \Lambda(A)$. This transformation maps each eigenpair (λ, x) of A to the eigenpair (ν, x) of R_σ, where $\nu = (\lambda - \sigma)^{-1}$; in particular, it sends the eigenvalues of A nearest to the spectral shift σ to well-separated extremal eigenvalues of R_σ, thus enabling their efficient recovery. Unfortunately, basic SI is insufficient to precondition the verification problem, since we do not know *a priori* where the desired minimum eigenvalue $\lambda_{\min}(S)$ is (indeed, determining this is part of the problem!)

Some iterative eigenvalue methods employ what can be interpreted as *adaptive* SI preconditioning schemes. In brief, these methods leverage the updated eigenvalue estimates θ_k

calculated in each iteration to construct an improved SI preconditioner R_{θ_k} for the *next* iteration; well-known examples of this approach include *Rayleigh quotient iteration* (RQI) [13, Sec. 4.3.3] and the *Jacobi-Davidson* (JD) method [13, Sec. 4.7], which can be interpreted as a kind of inexact subspace-accelerated RQI method. While these techniques promote rapid convergence, they require solving sequences of linear systems with *varying* coefficient matrices $A - \theta_k I$, which in practice entails either repeated matrix factorizations or the repeated construction of preconditioners to use with iterative linear system solvers (e.g. MINRES), both of which can be dominant costs in large-scale calculations.

In light of these considerations, we propose to to adopt the *locally optimal block preconditioned conjugate gradient* (LOBPCG) method [5] as the basis of our fast verification approach. In brief, LOBPCG is a preconditioned subspace-accelerated simultaneous iteration scheme for recovering a few *algebraically-smallest* eigenpairs of A. An important distinguishing feature of this method is that it employs a single (*constant*) preconditioner, which enables preconditioning the the verification problem with modest computational overhead.

B. Certifiable estimation

In consequence of the increasing interest in certifiable estimation methods (cf. [1] and the references therein), several recent works have made use of (specific instances of) Theorem 1 to certify the global optimality of candidate solutions to machine perception problems, whether as part of a Burer-Monteiro method [10, 15–19] or as a standalone procedure applied to estimates recovered from more traditional heuristic local search [20–22]. These prior works solve the verification problem by applying Krylov subspace methods (typically the Lanczos method) directly to the certificate matrix S to recover a minimum eigenpair. Reference [23] proposed a simple spectral-shifting strategy to convert the verification problem to the calculation of a *dominant* eigenpair (for which the Lanczos method exhibited better empirical performance), while [15] proposed to employ *accelerated* power iterations, which are more amenable to a distributed implementation.

To the best of our knowledge, this paper is the first to investigate the design of specialized algorithms for solving Problem 1, and in particular the first to propose the use of *preconditioned* eigensolvers to ameliorate the structural ill-conditioning identified in Section II-D whenever the calculation of a vector $x \in \mathbb{R}^n$ satisfying $x^\mathsf{T} S x < 0$ is required.

IV. ACCELERATING SOLUTION VERIFICATION WITH PRECONDITIONED EIGENSOLVERS

In this section we present our proposed fast verification method. We begin in Sec. IV-A with a review of the LOBPCG method, which forms the basis of our approach, and highlight some of the features that make it especially well-suited to solving the verification problem. We then describe our preconditioning strategy for use with LOBPCG in Sec. IV-B. Finally, we present our complete algorithm in Sec. IV-C.

A. The LOBPCG eigensolver

The *locally optimal block preconditioned conjugate gradient* (LOBPCG) method [5, 24] is an iterative eigenvalue method for calculating the k algebraically-smallest eigenpairs (λ, x) of the generalized symmetric eigenvalue problem:

$$Ax = \lambda Bx \tag{6}$$

where $A, B \in \mathbb{S}^n$ and $B \succ 0$. This method exploits the Courant-Fischer variational characterization of a minimum eigenpair (λ, x) of (6) as a solution of the following minimization problem [14, Sec. 9.2.6]:

$$\lambda = \min_{x \in \mathbb{R}^n} x^\mathsf{T} A x \quad \text{s.t.} \quad x^\mathsf{T} B x = 1 \tag{7}$$

to search for x by applying first-order optimization to (7).

In detail, the Lagrangian $L: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ of (7) is:

$$L(x, \lambda) \triangleq x^\mathsf{T} A x - \lambda(x^\mathsf{T} B x - 1) \tag{8}$$

and a straightforward calculation shows that the gradient of $L(x, \lambda)$ with respect to x is:

$$\nabla_x L(x, \lambda) = 2Ax - 2\lambda Bx. \tag{9}$$

Comparing (6) and (9) reveals that the KKT points (x, λ) of (7) are precisely the eigenpairs of (6); furthermore, given any *estimate* (θ, x) for a minimum eigenpair, we see that the residual r of the generalized eigenvalue equation (6):

$$r \triangleq Ax - \theta Bx \tag{10}$$

is parallel to the gradient $\nabla_x(L, \theta)$ in (9). A basic first-order optimization method would thus take $-r$ as a search direction to calculate an improved minimum eigenvector estimate [7].

LOBPCG improves upon a basic first-order optimization approach in three important ways. First and foremost, it exploits the fact that the objective in (7) is a quadratic parameterized by A by replacing the residual (gradient) search direction r in (10) with the *preconditioned* gradient search direction:

$$w \triangleq Tr, \tag{11}$$

where $T \in \mathbb{S}_{++}^n$ is a user-defined preconditioning operator. This should be chosen to approximate A^{-1} in the sense that $\kappa(TA) \ll \kappa(A)$; this has the effect of "undistorting" the contours of the quadratic objective in (7), producing significantly higher-quality search directions. This strategy has two major advantages versus the alternative SI preconditioning strategy used in e.g. RQI and JD. First, the figure of merit $\kappa(TA)$ for the preconditioner T is the same one appearing in the design of preconditioners for iterative linear system solvers (such as MINRES), which enables the direct application of this large body of prior work to the generalized eigenvalue problem (6). Second, in contrast to the *varying* resolvents $R_{\theta_k} \triangleq (A - \theta_k I)^{-1}$ used in adaptive SI preconditioning, the preconditioner T used in LOBPCG is designed for the parameter matrix A appearing in (7), and is therefore *constant*. Since constructing a preconditioner is often done via incomplete matrix factorization (which is expensive at large

scale), enabling the use of a constant preconditioner T can provide substantial computational savings.

Second, LOBPCG employs *subspace acceleration*: instead of searching *solely* over the 1-dimensional subspace spanned by w (as in basic preconditioned gradient descent), at each iteration it calculates the next eigenvector estimate x_{i+1} as a linear combination of the preconditioned gradient direction w_i, the current iterate x_i, *and* the previous iterate x_{i-1}:

$$x_{i+1} = \alpha_i w_i + \beta_i x_i + \gamma_i x_{i-1}, \tag{12}$$

where $\alpha_i, \beta_i, \gamma_i \in \mathbb{R}$ are coefficients. The use of a larger (3-dimensional) search space in (12) provides more flexibility in the choice of update x_{i+1}, and therefore the opportunity for enhanced performance. In particular, we remark that standard accelerated gradient methods (such as Nestorov, heavy ball, and the accelerated power method) all correspond to special cases of (12) for specific choices of α_i, β_i, and γ_i. Moreover, LOBPCG further improves upon standard accelerated gradient schemes by exploiting the specific form of (7) to dynamically compute *optimal* choices of α_i, β_i, and γ_i in (12) in each iteration. Specifically, the updated estimate x_{i+1} is chosen to minimize the Rayleigh quotient over the 3-dimensional subspace spanned by w_i, x_i, and x_{i-1}:

$$x_{i+1} \in \underset{x \in \text{span}\{w_i, x_i, x_{i-1}\}}{\text{argmin}} x^\mathsf{T} A x \quad \text{s.t.} \quad x^\mathsf{T} B x = 1. \tag{13}$$

In practice, the minimization (13) amounts to solving a projected generalized eigenvalue problem on the 3-dimensional subspace $\mathcal{S} = \text{span}\{w_i, x_i, x_{i-1}\}$, which can be done simply and efficiently using the Rayleigh-Ritz procedure (Alg. 1).

Finally, LOBPCG extends the single-vector iteration described in (10)–(13) to *simultaneous iteration* on a *block* $X \in \mathbb{R}^{n \times m}$ of m eigenvector estimates. This provides two major advantages. First, it permits the *simultaneous* recovery of the $k \leq m$ algebraically-smallest eigenpairs of (6). Second, a rough analysis suggests (and numerical experience confirms) that the convergence rate of the k smallest eigenpairs in a block of size m depends upon the eigengap $\gamma_{m,k} \triangleq \lambda_{m+1} - \lambda_k$ between the largest desired eigenvalue λ_k and the smallest eigenvalue λ_{m+1} *not* included in the block (cf. the discussion around eq. (5.5) in [5]). It is therefore advantageous to choose the block size m greater than the number of desired eigenpairs k if doing so will significantly enhance the eigengap $\gamma_{m,k}$. This capability is especially useful for addressing clustered eigenvalues, which we have seen are the primary challenge in solving the verification problem.

The complete LOBPCG algorithm is shown as Algorithm 2 (cf. Algorithms 1 and 2 of [25]).

B. Preconditioner design

In this section, we propose a simple yet highly effective algebraic preconditioning strategy for use with LOBPCG, based upon a modified incomplete symmetric indefinite factorization.

Our approach is based upon the following simple argument. Recall that every nonsingular matrix $A \in \mathbb{S}^n$ admits a *Bunch-Kaufman* factorization of the form [26, Sec. 4.4]:

$$PAP^\mathsf{T} = L \, \text{Diag}(D_1, \dots, D_K) L^\mathsf{T}, \tag{14}$$

Algorithm 1 The Rayleigh-Ritz procedure

Input: Symmetric linear operators $A, B \in \mathbb{S}^n$ with $B \succ 0$, full-rank basis matrix $S \in \mathbb{R}^{n \times k}$.

Output: Diagonal matrix $\Theta = \text{Diag}(\theta_1, \ldots, \theta_k) \in \mathbb{S}^k$ of Ritz values with $\theta_1 \leq \cdots \leq \theta_k$, matrix $C \in \text{GL}(k)$ satisfying $C^\mathsf{T}(S^\mathsf{T}BS)C = I_k$ and $C^\mathsf{T}(S^\mathsf{T}AS)C = \Theta$.

1: **function** RAYLEIGHRITZ(A, B, S)
2:     $D = \text{Diag}(S^\mathsf{T}BS)^{-1/2}$.
3:     Cholesky factorize $R^\mathsf{T}R = DS^\mathsf{T}BSD$.
4:     Calculate symmetric eigendecomposition:

$$Q\Theta Q^\mathsf{T} = R^{-\mathsf{T}}DS^\mathsf{T}ASDR^{-1}$$

   with eigenvalues sorted in ascending order.
5:     $C = DR^{-1}Q$.
6:     **return** (Θ, C)
7: **end function**

Algorithm 2 The locally optimal block preconditioned conjugate gradient (LOBPCG) method
---
Input: Symmetric linear operators A, B, T with $B, T \succ 0$, initial eigenvector estimates $X \in \mathbb{R}^{n \times m}$, number of desired eigenpairs $k \leq m$.

Output: Estimates of the k algebraically-smallest eigenpairs of the generalized symmetric eigenvalue problem (6).

1: **function** LOBPCG(A, B, T, X, k)
   // Initialization
2:     $(\Theta, C) = $ RAYLEIGHRITZ(A, B, X)
3:     $X = XC$ ▷ B-orthonormalize initial X
4:     $R = AX - BX\Theta$ ▷ Initial residuals
5:     $P = \emptyset$ ▷ Initialize P
6:     **repeat**
7:         $W = TR$ ▷ Precondition residuals
8:         $S = \begin{pmatrix} X & W & P \end{pmatrix}$ ▷ Search space basis
9:         $(\Theta, C) = $ RAYLEIGHRITZ(A, B, S)
10:        $X = SC_{:,1:m}$ ▷ Update eigenvector estimates
11:        $R = AX - BX\Theta$ ▷ Compute residuals
12:        $P = S_{:,m+1:end} C_{m+1:end, 1:m}$
13:        Determine the number n_c of the k algebraically-smallest eigenpairs (θ_i, x_i) that have converged.
14:    **until** $n_c == k$
15:    **return** $(\Theta_{1:k, 1:k}, X_{:, 1:k})$
16: **end function**

---

where P is a permutation matrix, L is unit lower-triangular, and D_k is a 1×1 or 2×2 symmetric nonsingular matrix for all $k \in [K]$. Given the factorization (14), consider the matrix:

$$T \triangleq P^\mathsf{T}L^{-\mathsf{T}}\text{Diag}\left(D_1^+, \ldots, D_K^+\right)L^{-1}P, \tag{15}$$

where for any nonsingular matrix $M \in \mathbb{S}^p$ with symmetric eigendecomposition $M = Q\,\text{Diag}(\lambda_1, \ldots, \lambda_p)Q^\mathsf{T}$, we define:

$$M^+ \triangleq Q\,\text{Diag}\left(|\lambda_1|^{-1}, \ldots, |\lambda_p|^{-1}\right)Q^\mathsf{T} \in \mathbb{S}_{++}^p \tag{16}$$

to be the positive-definite matrix obtained from M by replacing each of M's eigenvalues with the reciprocal of its absolute

value. Then $T \succ 0$, and a straightforward calculation yields:

$$TA = P^{-1}L^{-\mathsf{T}}\text{Diag}\left(D_1^+ D_1, \ldots, D_K^+ D_K\right)L^\mathsf{T}P. \tag{17}$$

Equation (17) shows that the product TA is similar to a block-diagonal matrix whose blocks $D_k^+ D_k$ all have eigenvalues contained in $\{\pm 1\}$. This proves that $\Lambda(TA) \subseteq \{\pm 1\}$, and therefore $\kappa(TA) = 1$; that is, the matrix T defined in (15) attains the *smallest possible* condition number $\kappa(TA)$, and is therefore an "ideal" preconditioner for use with LOBPCG.

Since computing a matrix decomposition can be expensive when A is large, in practice we propose to construct our preconditioners by replacing the *exact* factorization (14) with a tractable *approximation* $PAP^\mathsf{T} \approx \tilde{L}\,\text{Diag}(\tilde{D}_1, \ldots, \tilde{D}_K)\tilde{L}^\mathsf{T}$. Specifically, we propose to use a *limited-memory incomplete* symmetric indefinite LDL^T factorization [27]; this provides direct control over the number of nonzero elements appearing in the unit lower-triangular factors $\tilde{L}$, and therefore scales gracefully (in both memory and computation) to high-dimensional problems. We construct each diagonal block $\tilde{D}_k^+$ by directly computing and modifying the symmetric eigendecomposition of $\tilde{D}_k$ as in (16); this is again efficient, as each block is at most 2-dimensional. Finally, we avoid *explicitly instantiating* T as a (generically-dense) matrix. Instead, whenever we must compute a product with T, we sequentially apply each of the linear operators on the right-hand side of (15); this requires only 2 permutations, 2 sparse triangular solves, and a set of parallel multiplications with the (very small) diagonal blocks $\tilde{D}_k^+$ per application of T, and is thus again efficient.

C. An accelerated solution verification method

Our complete proposed method for solving the verification problem (Problem 1) is shown as Algorithm 3. Our method first tests the positive-semidefiniteness of the regularized certificate matrix $M \triangleq S + \eta I$ (where $\eta > 0$ is a user-selectable regularization parameter) by attempting to compute its Cholesky decomposition. Note that the use of regularization here is in fact *necessary*: since 0 is always an eigenvalue of S [cf. eq. (5)], then in the event $S \succeq 0$ (corresponding to a *global* minimizer in the Burer-Monteiro method), S's algebraically-smallest eigenvalue will be $\lambda_{\min}(S) = 0$, and therefore the result of *any* numerical procedure for testing S's positive-semidefiniteness [i.e. the sign of $\lambda_{\min}(S)$] will be dominated by finite-precision errors. On the other hand, if $S \succeq 0$ then $M \succ 0$, and therefore the Cholesky factorization attempted in line 3 exists and can be computed numerically stably [26, Sec. 4.2.7]; moreover, the factorization $M = LL^\mathsf{T}$ itself then serves as a certificate that $M = S + \eta I \succeq 0$, or equivalently that $S \succeq -\eta I$. The parameter η thus plays the role of a numerical tolerance for testing S's positive-semidefiniteness.

If the Cholesky factorization attempted in line 3 does *not* succeed, then $S \not\succeq 0$, and we must therefore recover a vector $x \in \mathbb{R}^n$ satisfying $x^\mathsf{T}Sx < 0$. Our algorithm achieves this by applying LOBPCG to compute a minimum eigenvector of the regularized certificate matrix M, using the modified incomplete factorization-based preconditioning strategy of Sec. IV-B. Once again we employ M instead of S because the former

Algorithm 3 Fast solution verification

Input: Certificate matrix $S \in \mathbb{S}^n$, minimum eigenvalue numerical nonnegativity tolerance $\eta > 0$, blocksize $m \geq 1$.
Output: Factorization $S + \eta I = LL^\mathsf{T}$ certifying $S \succeq -\eta I$, or a Ritz pair (λ, x) satisfying $\lambda = x^\mathsf{T} Sx$ and $\lambda < 0$.

1: **function** FASTVERIFICATION(S, η, m)
2:     Construct regularized certificate matrix $M = S + \eta I$.
3:     Attempt Cholesky factorization of M.
4:     **if** $M = LL^\mathsf{T}$ **then**
5:         **return** Cholesky factor L.   ▷ $S \succeq -\eta I$
6:     **end if**
7:     Incomplete symmetric indefinite factorization:

$$PMP^\mathsf{T} \approx L \operatorname{Diag}(D_1, \ldots, D_K) L^\mathsf{T}.$$

8:     Compute D_k^+ via (16) $\forall k \in [K]$.   ▷ Correct inertia
9:     Construct preconditioner T as in (15).
10:    Randomly sample initial block $X \in \mathbb{R}^{n \times m}$.
11:    $(\theta, x) = \text{LOBPCG}(M, I, T, X, 1)$.
12:    **return** $(\theta - \eta, x)$
13: **end function**

Algorithm 4 Test matrix sampler

1: **function** SAMPLETESTMATRIX(N, r, $W_{\max}$, γ).
2:     Sample N vertices: $\mathcal{V} \triangleq \{x_i \sim \mathcal{U}([0,1]^2) \mid i \in [N]\}$.
3:     Set edges $\mathcal{E} \triangleq \{\{i, j\} \mid i \neq j \text{ and } d(x_i, x_j) < r\}$.
4:     Sample $\mathcal{W} \triangleq \{w_{ij} \sim \mathcal{U}([0, W_{\max}]) \mid \{i, j\} \in \mathcal{E}\}$.
5:     Construct weighted graph $G \triangleq (\mathcal{V}, \mathcal{E}, \mathcal{W})$, and set

$$S \triangleq \begin{pmatrix} L(G) & 0 \\ 0 & -\gamma \end{pmatrix}. \tag{18}$$

6:     **return** S
7: **end function**

Finally, since $L(G) \succeq 0$ and $0 \in \Lambda(L(G))$ (as $L(G)$ is a graph Laplacian), it follows from (18) that $\lambda_{\min}(S) = -\gamma$ and the eigenvalue gap is $|-\gamma - 0| = \gamma$; that is, we can directly and *deterministically* control the relevant eigenvalue gap γ of the random test matrices S sampled from Algorithm 4.

For the following experiments we implemented Algorithm 4 in Python, using the `random_geometric_graph` method in the NetworkX library to sample the graphs G, with default parameters[1] $r = 1.25 \sqrt{\log(N)/(\pi N)}$ and $W_{\max} = 10^3$, while varying N and γ. As our baseline for comparison, we apply the Lanczos method to compute a minimum eigenpair of S. Both methods employ the relative-error convergence criterion:

$$\|Sx - \theta x\| \leq \tau |\theta| \tag{19}$$

for eigenpair estimates (θ, x), with $\tau = 10^{-2}$.

In our first set of experiments, we study the effect of varying the eigenvalue gap γ. We sample 50 matrices S from Algorithm 4 for each value of $\gamma = 10^k$, where $k \in \{1, \ldots, -6\}$ (holding $N = 25000$), and then solve the verification problem (Problem 1) using Algorithm 3 (both with and without the preconditioner T described in Sec. IV-B) and the Lanczos method. We record the elapsed computation time for each method, as well as the number of iterations required by LOBPCG in each version of Algorithm 3. Results from this experiment are shown in Figs. 1(a) and 1(b).

These results demonstrate that using LOBPCG together with the preconditioner proposed in Sec. IV-B renders Algorithm 3 essentially insensitive to the eigenvalue gap γ (note that the blue curves in Figs. 1(a) and 1(b) are effectively flat). In fact, we observe that Alg. 3 actually achieves a significant *gain* in performance for $\gamma \leq \eta$, since in that case we can immediately detect S's (numerical) positive-semidefiniteness using the test in lines 3–6. This small-γ regime is especially important because (as discussed in Sec. III) it is the most challenging for standard iterative eigenvalue methods, and in the context of the Burer-Monteiro approach is precisely the case that corresponds to certifying an *optimal solution*.

In our second set of experiments, we study the effect of varying the problem size N. As before, we sample 50 realizations of S from Algorithm 4 for each value of $N = 5000k$, where $k \in \{1, \ldots, 10\}$, and then solve the verification problem

is nonsingular (except in the unlikely case that $-\eta \in \Lambda(S)$), so that the factorization and corresponding preconditioner T computed in lines 7–9 are well-defined.

V. EXPERIMENTAL RESULTS

In this section we evaluate the performance of our fast verification method (Algorithm 3) on a variety of synthetic and real-world examples. All experiments are performed on a Lenovo T480 laptop with an Intel Core i7-8650U 1.90 GHz processor and 16 Gb of RAM running Ubuntu 22.04. Our experimental implementation of Algorithm 3 is written in C++, using CHOLMOD [28] to perform the Cholesky factorization in line 3, and SYM-ILDL [27] to compute the incomplete symmetric indefinite factorization in line 7.

A. Simulation studies

Our first set of experiments is designed to study how the performance of Algorithm 3 depends upon (i) the eigenvalue gap γ associated with the minimum eigenvalue $\lambda_{\min}(S)$ (cf. Sec. III-A) and (ii) the dimension of S.

To that end, we generate a set of suitable test instances using Algorithm 4. In brief, this method constructs a matrix S by randomly sampling a weighted *geometric random graph* G on the unit square $[0, 1]^2 \subset \mathbb{R}^2$ [29], forming its graph Laplacian $L(G)$, and then extending this matrix by a single row and column whose only nonzero value is the final element $-\gamma$ on the diagonal. This generative procedure is designed to capture several important properties of the certificate matrices arising in real-world instances of Problem 1, including:

- $S = L(G) + E$, where $L(G)$ is the Laplacian of a weighted graph G and E is a sparse matrix;
- A (spatial) *locality*-based sparsity pattern typical of real-world measurement networks in robotics and CV [1];
- Variation in the measurement precisions in the network.

---

[1]This choice of r ensures that the sampled graphs G are asymptotically almost surely connected as $N \to \infty$ [29, Thm. 13.2].
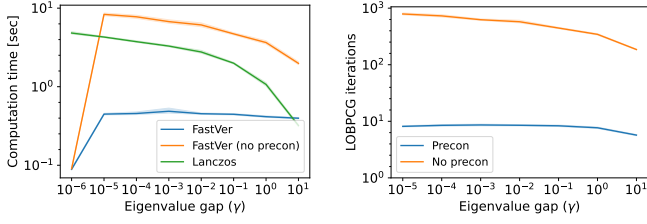
(a) Computation time vs. eigengap γ (b) LOBPCG iterations vs. eigengap γ
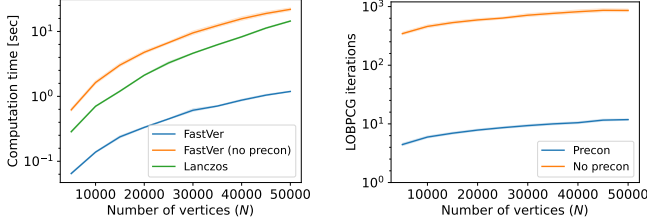
(c) Computation time vs. size N (d) LOBPCG iterations vs. size N

Fig. 1. Results for the simulation experiments. Figs. (a) and (c) plot the means and 95% confidence interval shadings for the elapsed computation times of our proposed fast verification method (both with and without preconditioning) and the Lanczos method applied to 50 random realizations of the verification problem sampled from Algorithm 4 while varying the eigenvalue gap γ and problem size N (respectively). Figs. (b) and (d) plot the means and 95% confidence interval shadings for the number of LOBPCG iterations required in Algorithm 3 with and without preconditioning.

(Problem 1) using Algorithm 3 (again both with and without preconditioning) and the Lanczos method. Results from this experiment are shown in Figs. 1(c) and 1(d).

These results show that the computational cost of Algorithm 3 (as measured in both elapsed time and LOBPCG iterations) scales gracefully with the size of the verification problem. In particular, Fig. 1(c) shows that Alg. 3's running time increases approximately *linearly* with N (note the logarithmic axis), which is consistent with the (approximately linearly) increasing cost of performing sparse matrix-vector multiplications and triangular solves with the operators A and T.

B. SLAM benchmarks

In this experiment we evaluate the impact of our fast verification approach (Algorithm 3) on the total computational cost of the Burer-Monteiro method. To do so, we solve a sequence of large-scale semidefinite relaxations obtained from a standard suite of SLAM benchmarks (see [16] for details). For each problem instance, we record the total time required to perform the local nonlinear optimizations (3), as well as the total time required to solve the subsequent verification problems (Problem 1) using (i) our proposed fast verification method (Alg. 3) and (ii) the spectrally-shifted Lanczos approach proposed in [23, Sec. III-C]. Results of this experiment are shown in Fig. 2.

These results clearly show that the verification step is the dominant cost of the Burer-Monteiro approach when using an unpreconditioned Krylov-subspace method, often requiring an order of magnitude more computational effort than nonlinear
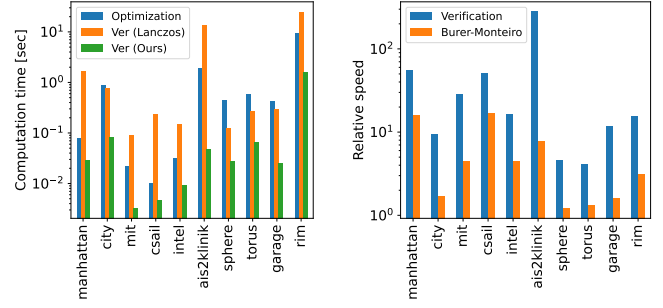
(a) Computational cost of the Burer-Monteiro method   (b) Relative speed using Algorithm 3 versus spectrally-shifted Lanczos

Fig. 2. Results for the large-scale SLAM benchmarks. (a): The total computation time required to perform local optimization (3) in the Burer-Monteiro method and solve the verification problem (Problem 1) using (i) the spectrally-shifted Lanczos method proposed in [23] and (ii) our proposed fast verification procedure (Algorithm 3). (b): The relative gain in computational speed achieved for both verification and the overall Burer-Monteiro method using our proposed fast verification scheme (Algorithm 3) versus the spectrally-shifted Lanczos approach of [23].

optimization. Our fast verification procedure significantly accelerates this rate-limiting step in all tested cases, achieving a relative gain in speed of between 4.08 (on `torus`) and 280 (on `ais2klinik`) for the verification problems, and a corresponding gain of between 1.21 (on `sphere`) and 16.9 (on `csail`) for the end-to-end Burer-Monteiro method. Crucially, solution verification is no longer the dominant cost when using our fast verification scheme [note that the blue bars are uniformly taller than the green in Fig. 2(b)].

VI. CONCLUSION

In this paper we showed how to significantly accelerate solution verification in the Burer-Monteiro method, the rate-limiting step in many state-of-the-art certifiable perception algorithms. We first established that the certificate matrices arising in the verification problem generically possess spectra that make it expensive to solve using standard iterative eigenvalue methods. Next, we showed how to overcome this challenge using *preconditioned* eigensolvers; specifically, we proposed a specialized solution verification algorithm based upon applying LOBPCG together with a simple yet highly effective incomplete factorization-based preconditioner. Experimental evaluation confirmed that our proposed verification scheme is very effective in practice, accelerating solution verification by up to 280x, and the overall Burer-Monteiro method by up to 16x, versus the Lanczos method when applied to relaxations derived from large-scale SLAM benchmarks.

REFERENCES

[1] D. Rosen, K. Doherty, A. Terán Espinoza, and J. Leonard, "Advances in inference and representation for simultaneous localization and mapping," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 4, pp. 215–242, 2021.

[2] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[3] A. Majumdar, G. Hall, and A. Ahmadi, "Recent scalability improvements for semidefinite programming with applications in machine learning, control and robotics," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 3, pp. 331–360, 2020.

[4] S. Burer and R. Monteiro, "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization," *Math. Program.*, vol. 95, pp. 329–357, 2003.

[5] A. Knyazev, "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method," *SIAM J. Sci. Comput.*, vol. 23, no. 2, pp. 517–541, 2001.

[6] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[7] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed. New York: Springer Science+Business Media, 2006.

[8] N. Boumal, V. Voroninski, and A. Bandeira, "The non-convex Burer-Monteiro approach works on smooth semidefinite programs," in *Advances in Neural Information Processing Systems (NeurIPS)*, Dec. 2016, pp. 2757–2765.

[9] D. Cifuentes and A. Moitra, "Polynomial time guarantees for the Burer-Monteiro method," *arXiv preprint arXiv:1912.01745*, 2019.

[10] D. Rosen, "Scalable low-rank semidefinite programming for certifiably correct machine perception," in *Intl. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Jun. 2020.

[11] N. Boumal, "A Riemannian low-rank method for optimization over semidefinite matrices with block-diagonal constraints," 2015, arXiv preprint: arXiv:1506.00575v2.

[12] M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre, "Low-rank optimization on the cone of positive semidefinite matrices," *SIAM J. Optim.*, vol. 20, no. 5, pp. 2327–2351, 2010.

[13] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, Eds., *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Philadelphia: The Society for Industrial and Applied Mathematics (SIAM), 2000.

[14] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, 2nd ed. The Society for Industrial and Applied Mathematics (SIAM), 2011.

[15] Y. Tian, K. Khosoussi, D. Rosen, and J. How, "Distributed certifiably correct pose-graph optimization," *IEEE Trans. on Robotics*, vol. 37, no. 6, pp. 2137–2156, Dec. 2021.

[16] D. Rosen, L. Carlone, A. Bandeira, and J. Leonard, "SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group," *Intl. J. of Robotics Research*, vol. 38, no. 2–3, pp. 95–125, Mar. 2019.

[17] J. Briales and J. Gonzalez-Jimenez, "Cartan-Sync: Fast and global SE(d)-synchronization," *Robotics and Automation Letters*, vol. 2, no. 4, pp. 2127–2134, Oct. 2017.

[18] T. Fan, H. Wang, M. Rubenstein, and T. Murphey, "Efficient and guaranteed planar pose graph optimization using the complex number representation," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Macau, China, Nov. 2019, pp. 1904–1911.

[19] F. Dellaert, D. Rosen, J. Wu, R. Mahony, and L. Carlone, "Shonan rotation averaging: Global optimality by surfing $SO(p)^n$," in *European Conf. on Computer Vision*, 2020.

[20] L. Carlone, D. Rosen, G. Calafiore, J. Leonard, and F. Dellaert, "Lagrangian duality in 3D SLAM: Verification techniques and optimal solutions," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, Sep. 2015.

[21] L. Carlone and F. Dellaert, "Duality-based verification techniques for 2D SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Seattle, WA, May 2015, pp. 4589–4596.

[22] J. Briales and J. Gonzalez-Jimenez, "Fast global optimality verification in 3d SLAM," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Daejon, Korea, Oct. 2016, pp. 4630–4636.

[23] D. Rosen and L. Carlone, "Computational enhancements for certifiably correct SLAM," Sep. 2017, presented at the International Conference on Intelligent Robots and Systems (IROS) in the workshop "Introspective Methods for Reliable Autonomy".

[24] A. Knyazev, M. Argentati, I. Lashuk, and E. Ovtchinnikov, "Block locally optimal preconditioned eigenvalue xolvers (BLOPEX) in hypre and PETSc," *SIAM J. Sci. Comput.*, vol. 29, no. 5, pp. 2224–2239, 2007.

[25] J. Duersch, M. Shao, C. Yang, and M. Gu, "A robust and efficient implementation of LOBPCG," *SIAM J. Sci. Comput.*, vol. 40, no. 5, pp. C655–C676, Oct. 2018.

[26] G. Golub and C. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.

[27] C. Greif, S. He, and P. Liu, "SYM-ILDL: Incomplete LDL^T factorization of symmetric indefinite and skew-symmetric matrices," *ACM Trans. Math. Softw.*, vol. 44, no. 1, Apr. 2017.

[28] Y. Chen, T. Davis, W. Hager, and S. Rajamanickam, "Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate," *ACM Trans. Math. Softw.*, vol. 35, no. 3, pp. 22:1–22:14, Oct. 2008.

[29] M. Penrose, *Random Geometric Graphs*, ser. Oxford Studies in Probability. Oxford University Press, 2003.