Efficient Algorithm for Large-and-Sparse LMI Feasibility Problems

Richard Y. Zhang and Javad Lavaei

***Abstract*—Linear matrix inequalities (LMIs) play a fundamental role in robust and optimal control theory. However, their practical use remains limited, in part because their solution complexities of $O(n^{6.5})$ time and $O(n^4)$ memory limit their applicability to systems containing no more than a few hundred state variables. This paper describes a Newton-PCG algorithm to efficiently solve large-and-sparse LMI feasibility problems, based on efficient log-det barriers for sparse matrices. Assuming that the data matrices share a sparsity pattern that admits a sparse Cholesky factorization, we prove that the algorithm converges in linear $O(n)$ time and memory. The algorithm is highly efficient in practice: we solve LMI feasibility problems over power system models with as many as $n = 5738$ state variables in 2 minutes on a standard workstation running MATLAB.**

I. Introduction

We consider linear matrix inequality (LMI) feasibility problems

$$\text{find } y \text{ such that } \sum_{i=1}^{m} y_i A_i \succ 0,$$

in which the $n \times n$ real symmetric data matrices $A_1, \dots, A_m$ are large-and-sparse. LMI feasibility problems are ubiquitous in robust and optimal control, and are used for Lyapunov stability analysis, LQR / LQG analysis and sythesis, H_∞ and mixed H_2/H_∞ analysis and synthesis, and model order reduction [1], [2]. Large-and-sparse instances of these problems are readily obtained by applying classic robust and optimal control theory to large-scale systems, including electric power systems, transportation networks, communication grids, and multi-agent distributed systems.

LMI problems are efficient to solve in theory, but difficult to solve in practice. The standard approach is to view the problem as a semidefinite program (SDP), and to solve it using a general-purpose interior-point method. These algorithms solve a set of Newton equations at each iteration, usually converging in no more than 20-30 iterations. An important feature of SDPs is that the interior-point Newton subproblem is almost always fully-dense, despite any apparent sparsity in the problem data. Consequently, every iteration forms and factors an $m \times m$ fully-dense matrix, using at least cubic $m^3/3$ arithmetic operations and quadratic $\frac{1}{2}m(m+1)$ units of memory.

The difficulty in solving large-scale LMI problems using interior-point methods has motivated first-order methods, like proximal descent, projected gradient descent, and Douglas–Rachford / ADMM [3]–[5]. By avoiding the dense second-order information, first-order methods have very low per-iteration costs, that can often be custom-tailored to the problem structure of a specific application. On the other hand, first order methods also converge significantly slower than interior-point methods.

A. Contributions

This paper proposes a Newton–PCG algorithm for solving LMI feasibility problems, based on the Projective Method of Gahinet and Nemirovski [6]. At a high level, the algorithm uses Newton's method to solve a determinant maximization problem that either generates a primal feasible point or converges towards a dual Farkas certificate; it solves the inner Newton subproblem using CG and a fast matrix-vector product due to Andersen, Dahl, and Vandenberghe [7]. More specifically, this paper contributes three key insights.

Solve the Newton Subproblem using conjugate gradients (CG). CG is an optimal Krylov subspace iterative method under the Newton metric, making it particularly appropriate as an iterative solver for the Newton subproblem. The most expensive part of CG is a single matrix-vector product with the Hessian matrix at each iteration, typically requiring cubic $\Theta(n^3 + N)$ time and quadratic $\Theta(n^2 + N)$ memory, where N is the number of nonzeros in the data matrices $A_1, \dots, A_m$. However, when the problem is sparse with an aggregate sparsity pattern that factors into a sparse Cholesky factorization pattern, it is possible to reduce these figures to linear $\Theta(n+N)$, by applying the efficient numerical algorithms described in [7].

Step-control to limit the growth of the condition number. If the associated Hessian matrix is sufficiently well-conditioned, then CG converges to machine precision in $O(1)$ iterations, in effect solving the Newton subproblem in linear $O(n + N)$ time and memory. In order to control the rate at which the Newton subproblem becomes ill-conditioned, we modify the Projective Method by limiting its maximum step-size. Despite this modification, we prove in Theorem 2 that Newton's method still makes a fixed amount of progress at every iteration, converging to the optimal solution within $O(1)$ Newton steps.

The Newton decrement is $\Omega(n)$ when the LMI is feasible. We prove in Theorem 3 that Newton's method makes much more progress per iteration when the LMI problem is feasible. In particular, each Newton step achieves a reduction of $\Omega(n)$, which is within a factor of maximum reduction achievable. Consequently, when the LMI problem is feasible, the algorithm converges in just 2-5 Newton steps,

This work was supported by the NSF Award 1808859, DARPA Award D16AP00002, AFOSR Award FA9550-17-1-0163, and ONR YIP.

R.Y. Zhang and J. Lavaei are with the Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA 94720, USA ryz@berkeley.edu and lavaei@berkeley.edu

for a total of 20-50 inner CG steps. If Newton's method does converge in $O(1)$ steps, then our overall algorithm is also linear $O(n+N)$ time and memory.

B. Related work

Second-order algorithms. The outer Newton framework of our algorithm is adopted from the Projective Method of Gahinet and Nemirovski [6]. Our same approach can also be applied to any other Newton framework, including the Phase-I barrier method / big-M method [8] and methods relating to the homogeneous self-dual embedding [9]. As we note in Section III-A, an important advantage of the Projective Method is that the algorithm attempts to directly solve the feasibility / infeasibility problem at every Newton step.

First-order algorithms. First-order methods have have been used extensively to solve convex feasibility problems; see the surveys [3]–[5]. All of these methods have worst-case complexities of $O(1/\epsilon)$ iterations to an ϵ-accurate solution, and in vast majority of cases, the worst-case bound is attained. This paper proposes a Newton-CG algorithm, which can be viewed as embedding a first-order method (i.e. CG) within an outer second-order method (i.e. Newton's method). The advantage here is that CG usually performs significantly better than its worst-case bounds in solving a linear system of equations. While Newton-CG methods have the same worst-case complexity of $O(1/\epsilon)$ iterations to ϵ-accuracy, they tend to converge in significantly fewer iterations in practice.

Newton-CG Algorithms. The idea of using a Newton-CG algorithm is not new; indeed, it was proposed in Karmarkar's original interior-point paper [10], as well as early solvers for LMI feasibility problems [11]. However, the approach is only effective if the cost of each matrix-vector product is small, and if the condition number of the Newton system can be effectively controlled, typically using a preconditioner; see [12], [13]. Indeed, the need for good preconditioners is commonly cited as the primary difficulty for Newton-CG methods in general-purpose interior-point solvers. Our contribution in this paper is to use the efficient matrix-vector products of Andersen, Dahl, and Vandenberghe [7] to reduce the cost of the matrix-vector product, and to use a specialized step-size rule to control the growth of the Newton condition number.

Algorithms for sparse matrices. Andersen, Dahl, and Vandenberghe also proposed a nonsymmetric interior-point method [14] based on the same efficient algorithms for the log-det barrier function [7] that we use extensively in this paper; they applied this method to LMI problems in [15]. Their algorithm directly solves the Newton system via Cholesky factorization, whereas the algorithm in this paper solves the same subproblem iteratively via CG. As a consequence, their algorithm has a much higher complexity of cubic $O(m^3)$ time and $O(m^2)$ memory per Newton step, but tends to be much more robust.

II. Problem Description

This paper considers LMI feasibility problem put into *homogenous canonical form*:

$$\text{find } y \text{ such that } A(y) \equiv \sum_{i=1}^{m} y_i A_i \prec 0, \tag{1}$$

where $A : \mathbb{R}^m \to \mathbb{S}^n$ is a *linear* matrix-valued function. The keyword "homogenous" refers to the scale invariance of the linear model. The feasible set of y satisfying $A(y) \prec 0$ is an open convex cone: if $A(x) \prec 0$ and $A(y) \prec 0$, then $A(\alpha x + \beta y) \prec 0$ holds for all $\alpha, \beta > 0$.

Every LMI feasibility problem can be converted into homogenous form. For example, we can find a feasible point x satisfying the inhomogenous relation $A(x) = C - \sum_{i=1}^{m} x_i A_i \prec 0$ by solving the augmented homogenous problem

$$\text{find } (y_0, y) \in \mathbb{R}^{m+1} \text{ s.t. } -Cy_0 - \sum_{i=1}^{m} y_i A_i \prec 0,\ y_0 < 0,$$

and rescaling $x = y/|y_0|$. Also, multiple LMI constraints can be aggregated into a single LMI constraint by concatenating them along the block diagonals of a big matrix, as in

$$\begin{aligned} &A_i(y) \prec 0 \quad \forall i \in \{1, \ldots, \ell\}, \\ \iff &\operatorname{diag}(A_1(y), A_2(y), \ldots, A_m(y)) \prec 0. \end{aligned}$$

Most practical algorithms avoid aggregating constraints this way, because it increases the size of the semidefinite cone by a factor of ℓ. However, the efficient log-det barrier function in Section III-C below treats the aggregated size-$n\ell$ LMI constraint identically as ℓ separate size-n LMI constraints. For this reason, the rest of this paper assumes a single LMI constraint, without either loss of generality or a practical performance penalty.

Problem (1) has Lagrangian dual:

$$\text{find } \begin{array}{c} X \succeq 0 \\ X \neq 0 \end{array} \text{ such that } A^T(X) \equiv \begin{bmatrix} A_1 \bullet X \\ \vdots \\ A_m \bullet X \end{bmatrix} = 0. \tag{2}$$

A choice of X satisfying (2) is known as a *Farkas (infeasibility) certificate* for (1), because it guarantees the inexistence of a feasible point for our original problem (1). Conversely, a feasible point for our original problem guarantees the inexistence of a Farkas certificate satisfying (2).

In this paper, we will actually focus our attention on finding a *strictly feasible* Farkas certificate satisfying

$$\text{find } X \succ 0 \text{ such that } A^T(X) = 0. \tag{3}$$

In other words, we assume that X is never perfectly singular (though we do allow it to be *numerically* singular, i.e. with some eigenvalues on the order of machine epsilon). Feasibility problems that do not satisfy this assumption are *ill-posed* [16], because an arbitrarily perturbation of the data matrices $A_1, \ldots, A_m$ can change its feasibility type. Non-strict feasibility problems are far more difficult to solve, and generally require sophisticated path-following algorithms, which are outside of the scope of this paper.

III. Main Ideas

Before we describe our algorithm in detail, we begin by discussing four key insights that we use to make the method efficient. We keep the discussion in this section at a relatively high level, and defer rigorous details to subsequent sections.

A. The Projective Method

Consider solving the strict feasibility pair

$$\begin{aligned} &\text{find } y \in \mathbb{R}^m \text{ such that } A(y) \prec 0 \\ &\text{or find } X \succ 0 \text{ such that } A^T(X) = 0, \end{aligned} \tag{4}$$

by applying Newton's method (with line search) to the determinant maximization problem

$$\text{maximize } \log\det(I - A(y)) \tag{5}$$

starting from the origin $y = 0$. Intuitively, every feasible point y satisfying $A(y) \prec 0$ is an increasing direction for (5), so attempting to maximize this objective will push y towards feasibility. If y does attain $A(y) \prec 0$ at any point, then we may terminate Newton's method and output y as a feasible point. Otherwise, we proceed with Newton's method as usual. If the method converges to a maximizer $y \to \hat{y}$, then the gradient of the log-det function here $\hat{X} = (I - A(\hat{y}))^{-1}$ is a Farkas certificate. If the method diverges, then setting $\tau = 1/\|y\|$ and $\hat{y} = \tau y$ yields a nonstrict almost-feasible point satisfying $A(\tau) \preceq \tau I$.

Following Gahinet and Nemirovski [6], we use the *Projective Method* to refer to the approach of solving the feasibility problem (4) through the maximum determinant problem (5). This particular name refers to an interesting feature of the method: every Newton iteration makes a direct attempt to solve the original feasibility problem via projection. To explain, let us define the positive definite matrix $S = I - A(y) \succ 0$ at the iterate y. Then Newton search direction Δy at y is defined by projecting S onto the range of A, as in

$$\begin{aligned} &\text{minimize } \|S^{-1/2}(\Delta S - S)S^{-1/2}\|_F^2 \\ &\text{subject to } A(\Delta y) + \Delta S = 0. \end{aligned} \tag{6}$$

If the positive definite matrix S is sufficiently close to the range of A, then we can expect its projection $\Delta S = A(\Delta y)$. If this occurs, then Δy is a strictly feasible point, and we may terminate Newton's method. Simultaneously, the Lagrange dual of (6) is defined by projecting S^{-1} onto the kernel of A, as in

$$\begin{aligned} &\text{minimize } \|S^{1/2}\,(\Delta X - S^{-1})\,S^{1/2}\|_F^2 \\ &\text{subject to } A^T(\Delta X) = 0. \end{aligned} \tag{7}$$

The dual solution (7) is explicitly given in terms of the primal solution (6) via

$$\Delta X = S^{-1} - S^{-1}\,\Delta S\,S^{-1}.$$

If S^{-1} is sufficiently close to the kernel of A, then we can expect its projection to be positive definite. In this case, ΔX is a Farkas certificate, and we may also terminate Newton's method.

More rigorously, the optimal values of (6) and (7) are $n - \delta^2$ and δ^2 respectively, where

$$\delta^2 = -A(\Delta y) \bullet S^{-1} = A(\Delta y) \bullet S^{-1}A(\Delta y)S^{-1} \tag{8}$$

is the *Newton decrement* at y. Classic complexity results for self-concordant functions show that if Newton's method does converge, then δ^2 must go to zero; see [17, Section 9.6.4] or [18, Section 4.1.5]. As soon as $\delta^2 < 1$, the Farkas certificate projection (7) is guaranteed to succeed, because

$$\begin{aligned} &\|S^{1/2}\,(\Delta X - S^{-1})\,S^{1/2}\|_F^2 < 1, \\ \implies &|\lambda_i(S^{1/2}\,\Delta X\,S^{1/2}) - 1| < 1 \qquad \forall i, \\ \implies &\lambda_i(\Delta X) > 0 \qquad \forall i. \end{aligned}$$

On the other hand, if Newton's method diverges and $\lambda_{\min}(S) \to \infty$, then the feasible point projection (6) is guaranteed to succeed for the same reason.

We end our discussing by noting that it is possible for Newton's method to diverge towards a choice of y satisfying nonstrict feasibility $A(y) \preceq 0$. In this case, S and δ^2 diverge but $\lambda_{\min}(S) = 1$, and neither projections (6) and (7) are expected to succeed. As we had discussed earlier in Section II, this scenario corresponds to a nonstrict feasibility problem, and requires more sophisticated path-following methods that are outside of the scope of this paper.

B. Quantifying infeasibility

Problem (5) has Lagrange dual

$$\begin{aligned} &\text{minimize } \operatorname{tr} X - \log\det X - n \\ &\text{subject to } A^T(X) = 0, \end{aligned} \tag{9}$$

whose feasible set coincides with the set of strict Farkas certificates in (4). Since Slater's conditions are satisfied, the two objectives coincide, so we see that (5) is bounded if and only if there exists a Farkas certificate.

Assuming infeasibility, let us write the unique primal solution of (5) as $\hat{y}$ and $\hat{S} = I - A(\hat{y})$. Then, complementary slackness yields $\hat{X} = \hat{S}^{-1}$ as the unique dual solution for (9), and the fact that the objectives coincide implies $\operatorname{tr}\hat{X} = n$. These insights motivate the scalar

$$\phi \equiv -\log\det\hat{X} \geq n - \operatorname{tr}\hat{X} = 0$$

as a quantification of the difficulty of the infeasibility problem. If ϕ is small, then the set of Farkas certificates is large in volume, and it is easy for an interior-point type algorithm like Newton's method to locate its analytic center.

Most feasibility problems in control theory have an upper-bound on the values of ϕ that can be deemed "practically useful". To explain, note that $\hat{X}$ also minimizes the ratio of the arithmetic and geometric means of its eigenvalues over the set of Farkas certificates,

$$\phi = \min_{X \succ 0} \left\{ n \log \left(\frac{\sum_{i=1}^n \lambda_i(X)}{\left[\prod_{i=1}^n \lambda_i(X)\right]^{1/n}} \right) : A^T(X) = 0 \right\},$$

so ϕ is bounded [8, p.76]

$$(n-1)\log\kappa_X \geq \phi \geq \log(\kappa_X/4), \tag{10}$$

where κ_X is the largest condition number over all Farkas certificates

$$\kappa_X = \min_{X \succ 0} \left\{ \frac{\lambda_{\max}(X)}{\lambda_{\min}(X)} : A^T(X) = 0 \right\}. \tag{11}$$

For feasibility problems that arise in control theory, the scalar κ_X has physical interpretations relating to notions of controllability, observability, stability, and passivity. For example, if the matrix $X \succ 0$ represents a quadratic Lyapunov function, then a large value of κ_X results from system modes that are close to being unstable. While the existence of such X would technically prove stability, the conclusion is not robust, and is easily invalidated by nonlinear effects and modeling errors. In practice, it would be more prudent to upper-bound the maximum "useful value" of κ_X, and this in turn places an upper-bound $\phi \leq (n-1)\log \kappa_X$.

On the other hand, Newton's method generates a sequence of increasing lower-bounds:

$$\phi \geq \log\det(I - A(y_j)) \geq \cdots \geq \log\det(I - A(y_0)) \geq 0.$$

Given a maximium "practically useful" value of ϕ, we may terminate the Newton iteration as soon as the best lower-bound exceeds this value. Standard arguments for self-concordant barrier functions show that each bound improves upon the last by at least a constant, so the Projective Method must terminate after at most $O(\phi)$ Newton steps.

C. Determinant maximization for sparse matrices

Given a sparsity pattern V, we define $\mathbb{S}^n_V \subseteq \mathbb{S}^n$ as the set of $n \times n$ real symmetric matrices with this sparsity pattern. Suppose that we view the usual log-det barrier function as a function from $\mathbb{S}^n_V$ to $\mathbb{R}$, as in

$$f(S) = \begin{cases} -\log\det S & S \succ 0,\ S \in \mathbb{S}^n_V \\ +\infty & \text{otherwise.} \end{cases} \tag{12}$$

Using this convention, the gradient of f at S is a function from $\mathbb{S}^n_V$ to $\mathbb{S}^n_V$

$$\nabla f(S) = -P_V(S^{-1}), \tag{13}$$

where $P_V : \mathbb{S}^n \to \mathbb{S}^n_V$ denotes the Euclidean projection onto $\mathbb{S}^n_V$, i.e. we set $[P_V(X)]_{i,j} = X_{i,j}$ if $(i,j) \in V$, and $[P_V(X)]_{i,j} = 0$ otherwise. Similarly, the Hessian matrix-vector product of f at S with Y is a function from $\mathbb{S}^n_V \times \mathbb{S}^n_V$ to $\mathbb{S}^n_V$

$$\begin{aligned} \nabla^2 f(S)[Y] &= \lim_{t \to 0} \frac{1}{t}[\nabla f(S + tY) - \nabla f(S)] \\ &= P_V(S^{-1} Y S^{-1}). \end{aligned} \tag{14}$$

Andersen, Dahl, and Vandenberghe [7] developed sparsity-exploiting algorithms to evaluate $\nabla f(S)$, and $\nabla^2 f(S)[Y]$, with about the same running time as the sparse Cholesky factorization algorithm. The key insight is to use the Cholesky factor as a sparse implicit representation of the dense matrix inverse S^{-1}. Indeed, the standard technique for evaluating $f(S)$ is to compute the Cholesky factor (L, D) satisfying

$$\begin{gathered} LDL^T = S, \quad L \text{ is lower-triangular}, \quad L_{i,i} = 1, \\ D \text{ is diagonal}, \quad D_{i,i} > 0, \end{gathered} \tag{15}$$

and to evaluate the determinant of D, noting that $\det L = 1$ by construction:

$$f(S) = -\log\det D - 2\log\det L = -\sum_{i=1}^n \log D_{i,i}. \tag{16}$$

(If Cholesky factorization fails, then S is not positive definite, and we have $f(S) = -\infty$.) Differentiating the sparse Cholesky factorization algorithm with respect to the nonzero elements of the input matrix yields an algorithm [8, Algorithm 4.1] for the projected inverse onto the *filled sparsity pattern* $\tilde{V} \supseteq V$

$$\tilde{X} = P_{\tilde{V}}(S^{-1}), \qquad \tilde{V} = \{(i,j), L_{i,j} \neq 0\}. \tag{17}$$

Discarding the added elements $\tilde{V} \backslash V$ (known as the fill-in) yields the usual gradient of f

$$-\nabla f(S) = P_V(S^{-1}) = P_V(P_{\tilde{V}}(S^{-1})) = P_V(\tilde{X}). \tag{18}$$

Similarly, differentiating the projected inverse formula yields an algorithm [7, Algorithm 5.1] for the following

$$M = P_{\tilde{V}}(S^{-1} Y S^{-1}) \qquad Y \in \mathbb{S}^n_{\tilde{V}}. \tag{19}$$

Discarding the fill-in $\tilde{V} \backslash V$ after computing M yields the Hessian matrix-vector product

$$\nabla f(S)[Y] = P_V(S^{-1} Y S^{-1}) = P_V(M).$$

Moreover, the algorithms that evaluate ∇f and $\nabla^2 f$ can be mechanically reversed if all matrix elements in the filled sparsity pattern $\tilde{V}$ in (17) are known. For our purposes, this amounts to an algorithm [7, Algorithm 4.2] that solves

$$\text{find } \tilde{S} \in \mathbb{S}^n_{\tilde{V}} \text{ such that } \tilde{X} = P_{\tilde{V}}(\tilde{S}^{-1}), \quad \tilde{S} \succ 0 \tag{20}$$

given $\tilde{X}$, and an algorithm [7, Algorithm 5.2] that solves

$$\text{find } \tilde{Y} \in \mathbb{S}^n_{\tilde{V}} \text{ such that } \tilde{W} = P_{\tilde{V}}(S^{-1} \tilde{Y} S^{-1}) \tag{21}$$

given $\tilde{W}$. These two algorithms also have about the same running time as sparse Cholesky factorization. We emphasize that (20) and (21) can only be solved in closed-form when all matrix elements in the filled pattern $\tilde{V}$ are known (an not just the elements in the original pattern $V \subseteq \tilde{V}$).

All four algorithms mentioned above have the same theoretical complexity and practical running time as Cholesky factorization

$$O(\omega^3 n) \text{ time and } O(\omega^2 n) \text{ memory}, \tag{22}$$

where ω is the maximum number of elements in a column of the Cholesky factor matrix L satisfying (15)

$$\omega = \max_{j \in \{1,\ldots,n\}} |J_j|, \qquad J_j = \{i : L_{i,j} \neq 0\}. \tag{23}$$

It is worth emphasizing that the value of ω depends only on the sparsity pattern V, and not on the exact choice of $S \in \mathbb{S}^n_V$ used to compute L. If L is sparse with $O(n)$ nonzero elements, then ω is guaranteed to be $O(1)$, and all four algorithms mentioned above (as well as sparse Cholesky factorization) are linear time and memory.

In practice, it is usually necessary to use a fill-reducing permutation in order to keep the value of ω small. To be more specific, we compute a permutation matrix Q, and evaluate each f, ∇f, and $\nabla^2 f$ by symmetrically permuting their inputs and reverting the permutation upon output, as in

$$f(S) = f(QSQ^T) \tag{24}$$
$$\nabla f(S) = Q^T \nabla f(QSQ^T)\, Q \tag{25}$$
$$\nabla^2 f(S)[Y] = Q^T \nabla^2 f(QSQ^T)[QYQ^T]\, Q. \tag{26}$$

Computing the optimal Q that minimizes ω is NP-hard, but standard heuristics from numerical linear algebra (like minimum degree and nested dissection) will readily produce high-quality orderings with ω that is within a modest factor of the optimal.

D. Iteratively solving the Newton subproblem

The main computational bottleneck at each iteration of the Projective Method is the solution of an $m \times m$ system of linear equations

$$\mathbf{H}\Delta y \equiv \mathbf{A}^T (S \otimes S)^{-1} \mathbf{A} \Delta y = \mathbf{A}^T \operatorname{vec} S^{-1} \tag{27}$$

for the Newton search direction Δy. Here, $\mathbf{A} = [\operatorname{vec} A_1, \ldots, \operatorname{vec} A_m]$ is the matrix of vectorized data matrices, and $S = I - A(y)$ is the current primal iterate. Considering that the method usually converges in tens of Newton iterations, it is helpful to think of the overall complexity as a modest constant times the cost of computing Δy. The standard approach is to solve (27) directly, by forming the matrix $\mathbf{H}$ explicitly and computing its Cholesky factorization, in $m^3/3 + O(n^3 m + n^2 m^2)$ arithmetic operations and $\Theta(m^2 + n^2)$ memory.

Alternatively, we can solve (27) iteratively, using an iterative Krylov subspace method like conjugate gradients (CG). We defer to standard texts for implementation details[1], and only note that at each iteration, CG performs a single matrix-vector product with $\mathbf{H}$ and some negligible linear algebra operations. Applying the sparsity-exploiting algorithms in Section III-C as

$$\mathbf{A}^T \operatorname{vec} S^{-1} = -A^T(\nabla f(S)),$$
$$\mathbf{A}^T (S \otimes S)^{-1} \mathbf{A} \Delta y = A^T(\nabla^2 f(S)[A(\Delta y)])$$

reduces the cost of each matrix-vector to $O(\omega^3 n + N)$ time and $O(\omega^2 n + N)$ memory, where ω was defined earlier in (23), and N is the total number of nonzero elements in the data matrices $A_1, \ldots, A_m$. Hence, each CG iteration linear $O(n + N)$ time and memory, so long as the value of ω is not too large.

Indeed, CG enjoys several "nice" properties that makes it particularly suitable as an inner iterative method within an outer Newton's method. First, it is *optimal within the Newton metric*, meaning that its k-th iterate p_k solves the following problem

$$\begin{aligned} &\text{minimize } \|S^{-1/2}(A(p_k) + \Delta S)S^{-1/2}\|_F^2 \\ &\text{subject to } p_k \in \operatorname{span}\{\mathbf{H}\Delta y, \ldots, \mathbf{H}^k \Delta y\}, \end{aligned} \tag{28}$$

[1]In our discussions, we will always refer to the variant of CG that choses its initial point to be the origin

where $\Delta S = -A(\Delta y)$ is the primal Newton direction in (6); see [19, Chapter 2] for a detailed discussion of this point. It is worth noting the similarities between (28) and (6): CG projects ΔS onto the Krylov subspace spanned by k matrix-vector products with the Hessian, under the same metric used to obtain ΔS as a projection of S in the first place. Second, CG is able to exploit clustering in the eigenvalues of the Hessian matrix $\mathbf{H}$ to accelerate convergence; its average-case behavior is often considerably better than its worst-case behavior. If the spectrum of $\mathbf{H}$ is sufficiently discrete, the convergence of CG is even superlinear. Third, by virtue of being a solution of the problem (28), every iterate of CG is a direction of descent for the original nonlinear objective.

In the worst-case, CG converges to an ϵ-accurate search direction p satisfying $\|p - \Delta y\|_{\mathbf{H}}^2 \leq \epsilon \|\Delta y\|_{\mathbf{H}}^2$ in

$$\lceil \sqrt{\kappa_H} \log(2/\epsilon) \rceil \text{ CG iterations} \tag{29}$$

where $\kappa_H = \operatorname{cond}(\mathbf{H}) \equiv \|\mathbf{H}\|\|\mathbf{H}^{-1}\|$ is the condition number of the matrix $\mathbf{H}$. Some linear algebra shows that $\sqrt{\kappa_H} \leq \operatorname{cond}(\mathbf{A}) \cdot \operatorname{cond}(S)$, so the worst-case number of CG iterations is actually proportional to $\operatorname{cond}(S)$, the condition number of the current primal matrix S. In most positive definite matrix optimization problems, $\operatorname{cond}(S)$ grows like $O(1/\epsilon)$ within an ϵ-neighborhood of the true solution. In the worst-case, Newton-CG methods require $O(1/\epsilon)$ total inner CG iterations to converge to an ϵ-accurate (outer) solution.

IV. Proposed Algorithm

Given large-and-sparse data matrices $A_1, \ldots, A_m \in \mathbb{S}_V^n$, our algorithm attempts to find a choice of y satisfying $A(y) \prec 0$ or a choice of $X \succ 0$ satisfying $A^T(X) = 0$ by implementing the Projective Method of Section III-A using the efficient barrier function f in Section III-C. In other words, we use Newton's method with back-tracking line search to solve

$$\underset{y \in \mathbb{R}^m}{\text{minimize}}\ g(y) \equiv f(I - A(y))$$

starting from the origin $y = 0$. At each Newton iteration, we compute the Newton direction $\Delta y = -\nabla^2 g(y)^{-1} \nabla g(y)$ using conjugate gradients (CG) as a set of inner iterations within an outer iterative algorithm.

A key feature of our algorithm is a step-size control that bounds the growth of the condition number of the current iterate S. This allows us to bound the growth in the worst-case number of inner CG iterations per outer Newton step; in practice, CG usually converges in far fewer iterations. Our actual algorithm is stated as Algorithm 1. The remainder of this section explains each step in further detail; a complexity analysis is given in Section V.

A. Computing the Newton direction

We begin by computing the Newton direction $\Delta y = \nabla^2 g(y)^{-1} \nabla g(y)$ using the framework outlined in Sec-

Algorithm 1 Newton-PCG for LMI feasibility

Input. Data matrices $A_1, \ldots, A_m \in \mathbb{S}^n_V$. Strict feasibility tolerance $\tau > 0$. Residual tolerance $\epsilon_r \in (0, 1)$. Condition number bound $\kappa > 1$. Backtracking line search parameters $\gamma \in (0, 1/2)$ and $\rho \in (0, 1)$. (Optional) Preconditioner matrix $\tilde{\mathbf{H}}$.

Output. A feasible y satisfying $A(y) \prec 0$, a Farkas certificate $X \succ 0$ satisfying $A^T(X) = 0$, or a nonstrict almost-feasible point y satisfying $A(y) \preceq \tau I$.

Algorithm. Set $y = 0$ and do:

1) (Newton direction) Use preconditioned conjugate gradients with preconditioner $\tilde{\mathbf{H}}$ to solve

$$\nabla^2 g(y)\Delta y = -\nabla g(y)$$

for the Newton directions Δy and$\Delta S = -A(\Delta y)$ to ϵ_r residual tolerance. Here, we define $S = I - A(y)$ and evaluate the gradient and the Hessian matrix-vector product at each iteration of PCG as

$$\nabla g(y) = A^T \nabla f(S),$$
$$\nabla^2 g(y)\Delta y = A^T \nabla^2 f(S)[A(\Delta y)].$$

2) (Feasibility / infeasibility test) Find α and β satisfying

$$\text{minimize } \alpha + \beta \text{ such that } -\alpha\Delta S \preceq S \preceq \beta \Delta S.$$

If $\alpha \leq -\tau$, then terminate and output Δy as a feasible point satisfying $A(\Delta y) \preceq -\tau I$. If $\beta \leq 1 - \tau$, then terminate and output the sparse matrix Z whose matrix inverse $\Delta X = Z^{-1}$ is a Farkas certificate satisfying $\Delta X \succeq \tau I$ and $A^T(\Delta X) = 0$.

3) (Line search) Determine the maximum step-size

$$t_{\max} = \min\left\{1, \frac{\kappa - 1}{\alpha\kappa + \beta}\right\},$$

and find $t/t_{\max} \in \{1, \rho, \rho^2, \ldots\}$ satisfying

$$g(y + t\Delta y) \leq g(y) + t\gamma \nabla g(y)^T \Delta y,$$

while evaluating each $g(y) = f(A(y))$. Make step $y \leftarrow y + t\Delta y$.

4) (Termination test) If $A(y) \preceq -\tau I$, then exit and return y as a feasible point. If y is not feasible but $g(y) > n\log(1/\tau)$, then exit and mark $y \leftarrow \tau y$ as a nonstrict almost-feasible point satisfying $A(y) \preceq \tau I$. If neither conditions hold, go to Step 1.

tion III-D with one minor modification: we optionally adopt the following preconditioner

$$\tilde{\mathbf{H}} = \mathbf{A}^T\mathbf{A} = \begin{bmatrix} \|A_1\|_F^2 & A_1 \bullet A_2 & \cdots & A_1 \bullet A_m \\ A_2 \bullet A_1 & \|A_2\|_F^2 & \cdots & A_2 \bullet A_m \\ \vdots & \vdots & \ddots & \vdots \\ A_m \bullet A_1 & A_m \bullet A_2 & \cdots & \|A_m\|_F^2 \end{bmatrix}$$

and use the preconditioned conjugate gradients (PCG) algorithm in lieu of the standard CG algorithm. We again defer the implementation details to standard texts, and note that each PCG iteration makes a single matrix-vector product with $\nabla^2 g(y)$, and a single *linear solve* with the preconditioner given a right-hand side $r \in \mathbb{R}^m$:

$$\text{find } x \in \mathbb{R}^m \text{ such that } \tilde{\mathbf{H}}x = r. \tag{30}$$

For many applications, the $m \times m$ matrix $\tilde{\mathbf{H}}$ is large-and-sparse, and can be factored into a sparse Cholesky factor. In these cases, (30) can be efficiently solved in $O(m)$ time, and the number of PCG iterations in (29) becomes independent of the condition number of the data $\mathrm{cond}(\mathbf{A})$. This is particularly important in controls applications, which often contain borderline-stable matrices that are near-singular. Even in cases where $\tilde{\mathbf{H}}$ cannot be efficiently factored, it is still usually worthwhile to use some sort of preconditioner, such as an incomplete Cholesky factor of $\tilde{\mathbf{H}}$, or its (block) diagonal elements.

Indeed, the matrix $\tilde{\mathbf{H}}$ implements the Euclidean projection onto the range of the operator A:

$$\tilde{\mathbf{H}}^{-1} A^T(U) = \arg\min_y \|A(y) - U\|_F^2,$$

a crucial building block for first-order feasibility algorithms, including classical projection methods like alternating projections and Dykstra's algorithm, as well as proximal point methods like the Douglas–Rachford iterations, Peaceman–Rachford iterations, and ADMM. As a consequence, a body of literature has been developed to classify the controls problems for which (30) can be efficiently solved.

B. Feasibility / infeasibility tests

After the Newton direction $\Delta y = \nabla^2 g(y)^{-1} \nabla g(y)$ has been computed, we evaluate the primal search direction $\Delta S = -A(\Delta y)$, and attempt to find the smallest value of α and β satisfying

$$\alpha \geq -\lambda_{\min}(S^{-1}\Delta S), \qquad \beta \geq \lambda_{\max}(S^{-1}\Delta S). \tag{31}$$

This can be done by a simple line search, noting that positive definiteness can be checked in $O(\omega^3 n)$ time and $O(\omega^2 n)$ memory by attempting to compute the sparse Cholesky factorization. We emphasize that α and β do not have to be particularly accurate, so long as they satisfy (31).

Now, if $\alpha < 0$ or if $\beta < 1$, then the "projection" part of the Projective Method has found a feasible point or infeasibility certificate, and may proceed to terminate. To explain why this is the case, note that by their definition in (6) and (7), the matrices

$$\Delta S = -A(\Delta y), \quad \Delta X = S^{-1} - S^{-1}\,\Delta S\, S^{-1} \tag{32}$$

satisfy primal feasibility $\exists w : S = A(w)$ and dual feasibility $A^T(\Delta X) = 0$. If either matrix is also positive definite, then they would constitute a valid certificate.

Proposition 1. *Define ΔS and ΔX as in (32) and let α and β be the smallest values satisfying (31). Then, we have $\alpha < 0$ if and only if $\Delta S \succ 0$ and $\beta < 1$ if and only if $\Delta X \succ 0$.*

Proof: Equation (32) implies the following eigenvalue relationship

$$1 = \lambda_{n-i+1}(S\Delta X) + \lambda_i(S^{-1}\Delta S) \quad \forall i \in \{1, \ldots, n\}$$

where the eigenvalues are ordered $\lambda_1 \geq \cdots \geq \lambda_n$. If $\lambda_{\min}(S^{-1}\Delta S) > 0$, then $\Delta S \succ 0$ because S^{-1} is positive definite. If $\lambda_{\max}(S^{-1}\Delta S) < 1$, then $\lambda_{\min}(S\Delta X) > 0$ and hence $\Delta X \succ 0$ because S is positive definite. ∎

Note that the matrix ΔX is dense, and should not be explicitly formed. Instead, we use the efficient algorithms in Section III-C to compute its projection onto the filled sparsity pattern $P_{\tilde{V}}(\Delta X) = P_{\tilde{V}}(S^{-1}) - P_{\tilde{V}}(S^{-1}\,\Delta S\,S^{-1})$, i.e. as instances of (17) and (19). We then look for a sparse matrix $Z \in \mathbb{S}^n_{\tilde{V}}$ satisfying $P_{\tilde{V}}(Z^{-1}) = P_{\tilde{V}}(\Delta X)$, and $Z \succ 0$, i.e. as an instance of (20). The fact that $P_V(Z^{-1}) = P_V(\Delta X)$ implies $A^T(Z^{-1}) = 0$, so $Z^{-1} \succ 0$ is a valid Farkas certificate. By representing the dense Farkas certificate ΔX using its sparse inverse Z, we have reduced the associated cost to $O(\omega^3 n)$ time and $O(\omega^2 n)$ memory.

C. Backtracking line search

In Step 3, we perform a backtracking line search with the step-size $t \geq 0$ limited to $t_{\max} = (\kappa - 1)/(\alpha\kappa + \beta)$. This way, the condition number of the new iterate is limited to be at most a factor of κ larger than the current iterate, as in

$$\mathrm{cond}(S + t\Delta S) \leq \left(\frac{1 + t\beta}{1 - t\alpha}\right)\mathrm{cond}(S) \leq \kappa \mathrm{cond}(S). \quad (33)$$

The following result shows that backtracking line search will always find a sufficiently large step-size that achieves the Armijo sufficient decrement condition

$$g(y + t\Delta y) \leq g(y) + \gamma t \nabla g(y)^T \Delta y, \quad (34)$$

with some $\rho \in (0, 1)$ and $\gamma \in (0, 1/2)$.

Theorem 2. *Backtracking line-search finds a step-size t satisfying (34) within the following range*

$$\frac{\rho t_{\max}}{t_{\max} + 1} \leq t \leq t_{\max},$$

Hence, Newton's method makes a decrement of at least

$$g(y + t\Delta y) - g(y) \leq -\gamma\rho\frac{\kappa - 1}{2\kappa}\delta^2,$$

where $\delta^2 = -\nabla g(y)^T \Delta y$ is the Newton decrement. Moreover, if $\alpha \leq 1/2 - \gamma$, then backtracking line-search takes a unit-step with $t = 1$.

Proof: The proof is given in Appendix A. ∎

We may always assume that $\delta \geq 1$, because $\delta < 1$ implies $\beta < 1$, and our algorithm would have terminated at the feasibility / infeasibility test. Consequently, the algorithm reduces the objective by at least $\gamma\rho(\kappa - 1)/2\kappa = O(1)$ at every Newton step.

V. Complexity

The value of a current iterate y can be measured by its ability to bound the maximum determinant of all Farkas certificates with trace n:

$$\epsilon = \max_{X \succ 0}\left\{\det(X) : \begin{array}{c} A^T(X) = 0 \\ \mathrm{tr}\, X = n \end{array}\right\} \ll 1.$$

Note that this quantity is related to the scalar ϕ defined in Section III-B by $\phi = \log(1/\epsilon) \geq -g(y) > 0$. It is helpful to interpret ϵ as an *absolute measure of accuracy*, and ϕ as the *number of accurate digits*.

Let us assume infeasibility, so that both ϵ and ϕ attain finite values. In the previous section, we proved that Newton's method decreases $g(y)$ by a constant factor after every Newton step (Theorem 2). Accordingly, we achieve an accuracy of $-g(y) = \phi = \log(1/\epsilon)$ in at most $O(\log(1/\epsilon))$ Newton steps. On the other hand, every Newton step increases the condition number of S by a factor of κ. At the j-th Newton step, PCG requires at most $O(\kappa^j)$ iterations to compute a sufficiently accurate Newton direction in order for further progress to be made. Taking the summation and applying the upper-bound $\log(1+t) \leq t$ yields a total of $O(1/\epsilon)$ PCG iterations.

However, the above analysis breaks down when the problem is feasible. Even though Newton's method diverges, we can compute the number of PCG iterations required to attain a value of y satisfying $-g(y) = \log(1/\epsilon)$. The following result is our key estimate.

Theorem 3. *Define κ_S is the largest condition number over all feasible points*

$$\kappa_S = \min_{S \succ 0}\left\{\frac{\lambda_{\max}(S)}{\lambda_{\min}(S)} : S = A(y)\right\}.$$

Then at an iterate y and $S = I - A(y)$, the Newton decrement $\delta^2 = -\nabla g(y)^T \Delta y$ is bounded by

$$\frac{n}{\kappa_S[\mathrm{cond}(S)]^2} \leq \delta^2 \leq n.$$

Proof: The proof is given in Appendix B. ∎

Combining this result with Theorem 2 shows that the algorithm makes a massive $\Omega(n)$ decrement at each Newton step, within a constant factor of the maximum achievable. If the value of κ_S is not too small (and hence the feasibility problem is not too difficult to solve), then our algorithm should quickly diverge to a strict or nonstrict feasible point.

VI. Numerical Example

Finally, we benchmark the performance of our algorithm by solving the *structured* Lyapunov feasibility problem:

$$\text{find } P \in \mathbb{S}^n_V \text{ such that } A^T P + PA \prec 0, \quad (35)$$

where V is an *a priori* imposed sparsity pattern. (Note that Hurwitz stability of A guarantees $P \succ 0$, so positive definiteness does not need to be explicitly enforced.) Problem (35) arises as a sparse and scalable surrogate for the fully-dense quadratic Lyapunov function in stability analysis and model reduction applications. The sparse Cholesky factor of P provides important insights on the block-wise coupling of different modes in A, and can be used in decoupled and decentralized control. The problem is also a basic building block for sparse optimal control, including LQR and H_∞.

Our numerical experiments source their A matrices from standard power system test cases in the MATPOWER suite, and enforce P to have the same sparsity pattern as $A + A^T$. For each test case, we compute the (nearly) complex symmetric bus admittance matrix $Y_{bus} \approx Y^T_{bus}$, and embed

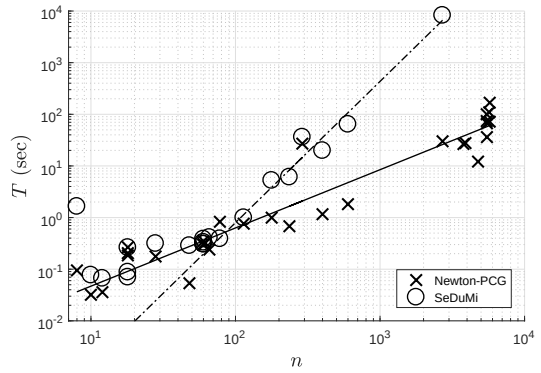

Figure 1. Comparison of our algorithm vs SeDuMi over 31 structured Lyapunov problems. Regression lines show $O(n^{1.12})$ and $O(n^{2.73})$ respectively.

Table I
SOME SELECT STRUCTURED LYAPUNOV PROBLEMS.

Problem Description				Newton-PCG			SeDuMi
#	n	m	ω	Newt	PCG	sec	sec
19	400	1580	42	3	18	1.2	19
20	600	2408	38	3	21	1.8	64
21	2708	10902	64	5	57	30	8255
22	3776	14400	84	4	32	26	(17912)
23	3902	14675	86	4	32	28	(19617)
25	5472	21240	112	3	18	103	(50048)
31	5738	24211	88	5	65	169	(57080)

it into a real matrix with a slight diagonal shift, as in $A \equiv \tau I_n + \left[\begin{smallmatrix} G & -C \\ C & G \end{smallmatrix}\right]$, where $G = \mathrm{Re}\, Y_{bus}$ and $C = \mathrm{Re}\, Y_{bus}$. It is worth emphasizing that the Hermitian projection $Y_{bus} + Y_{bus}^H$ is positive semidefinite. We set the diagonal shift τ to be 0.2% of the spectral radius of Y_{bus}, in order to give all matrices a similar asymptotic decay rate. Our algorithm parameters are set to $\tau = 10^{-3}$, $\epsilon_r = 10^{-3}$, $\kappa = 3$, $\gamma = 0.01$ and $\rho = 0.5$. All numerical experiments are performed on an Intel Xeon E3-1230 quad-core 3.30GHz CPU with 16 GB of RAM.

Figure 1 compares the running times between our algorithm and SeDuMi, an open-source general-purpose interior-point solver for semidefinite programs. Table I gives the associated details for some select examples: ω is the complexity parameter defined in (23), "PCG" is the *total* number of inner PCG iterations, and estimated running times are shown in parantheses. Our algorithm manages to find feasible points for problems as large as $n = 5738$ in under two minutes, while the largest problem solved by SeDuMi has just $n = 2708$ and took 2.5 hours. Logarithmic regression (weighted against $\log n$) results in empirical time complexities of $O(n^{1.12})$ for our algorithm and $O(n^{2.73})$ for SeDuMi. As we explained in Section III-D, the approximate cubic time complexity of SeDuMi comes from factoring a fully-dense matrix of size-m, while our near-linear time complexity comes from a fast matrix-vector product within PCG.

VII. CONCLUSIONS

This paper describes an efficient algorithm that solves large-and-sparse LMI feasibility problems. The key insight is to solve the dense $m \times m$ Newton subproblem using an iterative method like conjugate gradients, because each (inner) iteration can be performed in linear $O(n)$ time and memory. We also use a step-size control to bound the rate at which the number of inner iterations per outer iteration can grow. In practice, the algorithm usually converges in no more than 100 inner iterations, and this allows us to solve large-scale LMI feasibility problems with as many as $n = 5738$ state variables in less than two minutes, on a standard workstation computer.

REFERENCES

[1] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*. Siam, 1994, vol. 15.

[2] K. Zhou, J. C. Doyle, K. Glover *et al.*, *Robust and optimal control*. Prentice hall New Jersey, 1996, vol. 40.

[3] H. H. Bauschke and J. M. Borwein, "On projection algorithms for solving convex feasibility problems," *SIAM Rev.*, vol. 38, no. 3, pp. 367–426, 1996.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[5] N. Parikh, S. Boyd *et al.*, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[6] P. Gahinet and A. Nemirovski, "The projective method for solving linear matrix inequalities," *Math. Program.*, vol. 77, no. 1, pp. 163–190, 1997.

[7] M. S. Andersen, J. Dahl, and L. Vandenberghe, "Logarithmic barriers for sparse matrix cones," *Optim. Method. Softw.*, vol. 28, no. 3, pp. 396–423, 2013.

[8] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Rev.*, vol. 38, no. 1, pp. 49–95, 1996.

[9] Y. Ye, M. J. Todd, and S. Mizuno, "An $o(\sqrt{nL})$-iteration homogeneous and self-dual linear programming algorithm," *Math. Oper. Res.*, vol. 19, no. 1, pp. 53–67, 1994.

[10] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. 16th ACM Symp. Theory of computing*. ACM, 1984, pp. 302–311.

[11] L. Vandenberghe and S. Boyd, "A primal-dual potential reduction method for problems involving matrix inequalities," *Math. Program.*, vol. 69, no. 1-3, pp. 205–236, 1995.

[12] K.-C. Toh and M. Kojima, "Solving some large scale semidefinite programs via the conjugate residual method," *SIAM Journal on Optimization*, vol. 12, no. 3, pp. 669–691, 2002.

[13] J. Gondzio, "Matrix-free interior point method," *Computational Optimization and Applications*, vol. 51, no. 2, pp. 457–480, 2012.

[14] M. S. Andersen, J. Dahl, and L. Vandenberghe, "Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones," *Math. Program. Comp.*, vol. 2, no. 3-4, pp. 167–201, 2010.

[15] M. S. Andersen, L. Vandenberghe, and J. Dahl, "Linear matrix inequalities with chordal sparsity patterns and applications to robust quadratic optimization," in *2010 IEEE Int. Symp. Computer-Aided Control System Design (CACSD)*. IEEE, 2010, pp. 7–12.

[16] J. Renegar, "Incorporating condition measures into the complexity theory of linear programming," *SIAM J. Optim.*, vol. 5, no. 3, pp. 506–524, 1995.

[17] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[18] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2013, vol. 87.

[19] A. Greenbaum, *Iterative methods for solving linear systems*. Siam, 1997, vol. 17.

APPENDIX

A. Proof of Theorem 2

We begin by proving a technical lemma and an associated lower-bound on the log-det function.

Lemma 4. *The function $f(t) = t^{-2}(t - \log(1+t))$ is monotonously decreasing for all $t > -1$, i.e. we have $\frac{\mathrm{d}}{\mathrm{d}t} f(t) < 0$.*

Proof: This follows from the fact that

$$f(t) = t^{-2} \int_0^t \frac{\tau}{1+\tau}\, d\tau = \int_0^1 \frac{s}{1+ts}\, ds.$$

Both the integrand and its partial derivative are continuous for $t > -1$, so we can commute differentiation with integration and yield

$$\frac{\mathrm{d}}{\mathrm{d}t} f(t) = \int_0^1 \left(\frac{\partial}{\partial t} \frac{s}{1+ts}\right) ds = \int_0^1 \frac{-s^2}{(1+ts)^2}\, ds,$$

showing that the gradient is always negative. ∎

Corollary 5. *For any $D \in \mathbb{S}^n$ satisfying $I + D \succ 0$, we have*

$$\log\det(I+D) \geq \operatorname{tr} D - \frac{\mu - \log(1+\mu)}{\mu} \|D\|_F^2$$

where $\lambda_{\min}(D) \geq \mu > -1$.

Proof: Write $\lambda_1 \geq \cdots \geq \lambda_n$ as the n eigenvalues of D, and use the monotonicity of f the bound the sum

$$\operatorname{tr} D - \log\det(I+D) = \sum_{i=1}^n \lambda_i^2 f(\lambda_i) \leq f(\lambda_n) \sum_{i=1}^n \lambda_i^2.$$

∎

Following Nesterov [18, p.186], we define the auxillary functions $\omega(t) = t - \log(1+t)$ and $\omega_*(t) = -t - \log(1-t)$. Note that these two functions are convex conjugates, and so satisfy [18, Lemma 4.1.4]

$$\begin{aligned} \omega(\xi) &= \max_{0 \leq t < 1} \{\xi t - \omega_*(t)\}, \qquad \forall \xi \geq 0, \\ \omega_*(t) &= \max_{\xi \geq 0} \{\xi t - \omega(\xi)\}, \qquad \forall t \in [0, 1). \end{aligned} \tag{36}$$

It is helpful to view both functions as approximately $t^2/2$, since

$$\frac{t^2}{2(t+1)} \leq \omega(t) \leq \frac{t^2}{2} \leq \omega_*(t) \leq \frac{t^2}{2} + t^3, \tag{37}$$

where the first three bounds hold for all $t \geq 0$, and the final upper-bound holds only for $t \leq 0.81$. Finally, note that the Newton decrement, defined as

$$\begin{aligned} \delta^2 &= -\nabla g(y)^T \Delta y = \Delta y^T \nabla^2 g(y) \Delta y \\ &= \sum_{i=1}^n \lambda_i^2 (S^{-1} \Delta S), \end{aligned}$$

gives upper-bounds $\alpha \leq \delta$ and $\beta \leq \delta$, since

$$\alpha = |\lambda_{\min}(S^{-1}\Delta S)|, \quad \beta = \lambda_{\max}(S^{-1}\Delta S),$$

and the Frobenius norm is bigger than any individual eigenvalue.

Proof of Theorem 2: Corollary 5 implies the following bound

$$g(y + t\Delta y) - g(y) \leq \frac{\delta^2}{\mu^2}[-t\mu^2 + \omega_*(t\mu)]$$

for any $\mu \geq \alpha > 0$ in (31). To prove the minimum step-size $t \geq t_{\max}/(1+t_{\max})$, we chose $\mu = 1/t_{\max}$, noting that positive-definiteness at step-size $t_{\max}$ implies $1 - \alpha t_{\max} > 0$ and hence $\mu \geq \alpha$. Using (36) to minimize the bound over t yields an optimal step-size of $t_* = 1/(1+\mu) = t_{\max}/(1+t_{\max})$, with an associated decrement of

$$g(y + t\Delta y) - g(y) \leq -\frac{\delta^2}{\mu^2}\omega(\mu) \leq -t_* \frac{\delta^2}{2}.$$

The second bound here follows from the first lower-bound in (37). Hence, the Armijo sufficient decrement condition (34) is satisfied at the step-size $t = \rho t_*$. To prove the minimum decrement, we lower-bound t_* in terms of κ, as in

$$t_* = \frac{\kappa - 1}{(\alpha+1)\kappa + (\beta - 1)} \geq \left(\frac{\kappa - 1}{\kappa + 1}\right) \frac{1}{\delta + 1},$$

noting that $\alpha \leq \delta$ and $\beta \leq \delta$. Substituting this into the Armijo condition (34) yields the desired decrement. Finally, to prove that unit-step is achieved at $\alpha \leq 1/2 - \gamma$, we set $t = 1$ and observe the upper-bound

$$g(y + \Delta y) - g(y) \leq \frac{\delta^2}{\alpha^2}[-\alpha^2 + \omega_*(\alpha)].$$

Suppose that $\alpha \leq 0.81$ (which is guaranteed by $\alpha \leq 1/2 - \gamma$). Then, by the upper-bound in (37), we have

$$\begin{aligned} \frac{\delta^2}{\alpha^2}[-\alpha^2 + \omega_*(\alpha)] &\leq \frac{\delta^2}{\alpha^2}[-\alpha^2 + \frac{\alpha^2}{2} + \alpha^3] \\ &= -\left(\frac{1}{2} - \alpha\right)\delta^2. \end{aligned}$$

Hence, if $1/2 - \alpha \geq \gamma$, or equivalently if $\alpha \leq 1/2 - \gamma$, then the Armijo sufficient decrement condition (34) is satisfied at $t = 1$. ∎

B. Proof of Theorem 3

Proof: The upper-bound is a classic result for the log-det barrier; see Nesterov [18, Theorem 4.3.3]. To derive the lower-bound, write $L = \lambda_{\max}(S)$ and $\mu = \lambda_{\min}(S)$ and observe that the Newton decrement satisfy

$$\begin{aligned} \frac{\delta^2}{2} &= \min_{\Delta X} \{\frac{1}{2}\|S^{\frac{1}{2}}(\Delta X - S^{-1})S^{\frac{1}{2}}\|_F^2 : A^T(\Delta X) = 0\}, \\ &\geq \mu^2 \min_{\Delta X} \{\frac{1}{2}\|\Delta X - S^{-1}\|_F^2 : A^T(\Delta X) = 0\}, \\ &= \mu^2 \min_{U, \Delta X} \{\frac{1}{2}\|\Delta X - U\|_F^2 : A^T(\Delta X) = 0,\ U = S^{-1}\}, \\ &\geq \mu^2 \min_{U, \Delta X} \{\frac{1}{2}\|\Delta X - U\|_F^2 : A^T(\Delta X) = 0,\ U \succeq S^{-1}\}, \\ &\geq \mu^2 \min_{U, \Delta X} \{\frac{1}{2}\|\Delta X - U\|_F^2 : A^T(\Delta X) = 0,\ U \succeq \frac{1}{L}\}, \\ &= \frac{\mu^2}{L^2} \min_{U, \Delta X} \{\frac{1}{2}\|\Delta X - U\|_F^2 : A^T(\Delta X) = 0,\ U \succeq I\}. \end{aligned}$$

The first bound follows from $\|S^{\frac{1}{2}} X S^{\frac{1}{2}}\|_F \geq \mu \|X\|_F$, and the second bound follows by relaxing the equality into an

inequality, and the third bound follows because $S^{-1} \succeq \frac{1}{L} I$. Now, to bound the last term, we take the Lagrangian dual

$$\begin{aligned}
&\min_{\Delta X} \{\frac{1}{2}\|\Delta X - U\|_F^2 : A^T(\Delta X) = 0,\ U \succeq I\},\\
=&\max_{A(\Delta y)\succeq 0} \{\operatorname{tr} A(\Delta y) - \frac{1}{2}\|A(\Delta y)\|_F^2\},\\
=&\max_{A(\Delta y)\succeq 0} \{\frac{n}{2} - \frac{1}{2}\|A(\Delta y) - I\|_F^2\},\\
=&\frac{n}{2} - \min_{A(\Delta y)\succeq 0} \frac{1}{2}\|A(\Delta y) - I\|_F^2,\\
\geq&\frac{n}{2}\left(1 - \min_{A(\Delta y)\succeq 0} \|A(\Delta y) - I\|^2\right).
\end{aligned}$$

The final line upper-bounds the Frobenius with the spectral norm, as in $\|X\|_F^2 \leq n\|X\|$. Finally, we derive an upper-bound to the minimization by adding $A(\Delta y) \preceq I$ as a constraint, and solving

$$\begin{aligned}
&\min_{A(\Delta y)\succeq 0} \|A(\Delta y) - I\|^2\\
\leq&\min_{A(\Delta y)\succeq 0} \{\|A(\Delta y) - I\|^2 : A(\Delta y) \preceq I\}\\
=&\min_{A(\Delta y)\succeq 0} \left(\frac{1}{\operatorname{cond}(A(\Delta y))} - 1\right)^2,\\
=&(\kappa_S^{-1} - 1)^2.
\end{aligned}$$

Combined, we have the proved the following bound

$$\begin{aligned}
\delta^2 &\geq n\,\frac{\mu^2}{L^2}\,(1 - (\kappa_S^{-1} - 1)^2)\\
&\geq n\,\frac{\mu^2}{L^2}\,(2\kappa_S^{-1} - \kappa_S^{-2}) \geq n\,\frac{\mu^2}{L^2}\,\kappa_S^{-1},
\end{aligned}$$

as desired. ■