# A Faster Interior Point Method for Semidefinite Programming

Haotian Jiang* Tarun Kathuria† Yin Tat Lee‡ Swati Padmanabhan§
Zhao Song¶

**Abstract**

Semidefinite programs (SDPs) are a fundamental class of optimization problems with important recent applications in approximation algorithms, quantum complexity, robust learning, algorithmic rounding, and adversarial deep learning. This paper presents a faster interior point method to solve generic SDPs with variable size $n \times n$ and $m$ constraints in time

$$\widetilde{O}(\sqrt{n}(mn^2 + m^\omega + n^\omega)\log(1/\epsilon)),$$

where $\omega$ is the exponent of matrix multiplication and $\epsilon$ is the relative accuracy. In the predominant case of $m \geq n$, our runtime outperforms that of the previous fastest SDP solver, which is based on the cutting plane method [JLSW20].

Our algorithm's runtime can be naturally interpreted as follows: $\widetilde{O}(\sqrt{n}\log(1/\epsilon))$ is the number of iterations needed for our interior point method, $mn^2$ is the input size, and $m^\omega + n^\omega$ is the time to invert the Hessian and slack matrix in each iteration. These constitute natural barriers to further improving the runtime of interior point methods for solving generic SDPs.

*jhtdavid@uw.edu. University of Washington.
†tarunkathuria@berkeley.edu. University of California, Berkeley.
‡yintat@uw.edu. University of Washington.
§pswati@uw.edu. University of Washington
¶zhaos@ias.edu. Princeton University and Institute for Advanced Study.

# Contents

# 1 Introduction

Semidefinite programs (SDPs) constitute a class of convex optimization problems that optimize a linear objective over the intersection of the cone of positive semidefinite matrices with an affine space. SDPs generalize linear programs and have a plethora of applications in operations research, control theory, and theoretical computer science [VB96]. Applications in theoretical computer science include improved approximation algorithms for fundamental problems (e.g., Max-Cut [GW95], coloring 3-colorable graphs [KMS94], and sparsest cut [ARV09]), quantum complexity theory [JJUW11], robust learning and estimation [CG18, CDG19, CDGW19], and algorithmic discrepancy and rounding [BDG16, BG17, Ban19]. We formally define SDPs with variable size $n \times n$ and $m$ constraints:

**Definition 1.1** (Semidefinite programming)**.** *Given symmetric*[1] *matrices* $C, A_1, \cdots, A_m \in \mathbb{R}^{n \times n}$ *and* $b_i \in \mathbb{R}$ *for all* $i \in [m]$*, the goal is to solve the convex optimization problem*

$$\max \langle C, X \rangle \text{ subject to } X \succeq 0, \langle A_i, X \rangle = b_i \ \forall i \in [m] \tag{1}$$

*where* $\langle A, B \rangle := \sum_{i,j} A_{i,j} B_{i,j}$ *is the trace product.*

**Cutting plane and interior point methods** Two prominent methods for solving SDPs, with runtimes depending *logarithmically* on the accuracy parameter $\epsilon$, are the *cutting plane method* and the *interior point method*.

The cutting plane method maintains a convex set containing the optimal solution. In each iteration, the algorithm queries a separation oracle, which returns a hyperplane that divides the convex set into two subsets. The convex set is then updated to contain the subset with the optimal solution. This process is repeated until the volume of the maintained set becomes small enough and a near-optimal solution can be found. Since Khachiyan proved [Kha80] that the ellipsoid method solves linear programs in polynomial time, cutting plane methods have played a crucial role in both discrete and continuous optimization [GLS81, GV02].

In contrast, interior point methods add a barrier function to the objective and, by adjusting the weight of this barrier function, solve a different optimization problem in each iteration. The solutions to these successive problems form a well-defined *central path*. Since Karmarkar proved [Kar84] that interior point methods can solve linear programs in polynomial time, these methods have become an active research area. Their number of iterations is usually the square root of the number of dimensions, as opposed to the linear dependence on dimensions in cutting plane methods.

Since cutting plane methods use less structural information than interior point methods, they are slower at solving almost all problems where interior point methods are known to apply. However, SDPs remain one of the most fundamental optimization problems where the state of the art is, in fact, the opposite: the current fastest cutting plane methods[2] of [LSW15, JLSW20] solve a general SDP in time $m(mn^2+m^2+n^\omega)$, while the fastest SDP solvers based on interior point methods in the work of [NN92] and [Ans00] achieve runtimes of $\sqrt{n}(m^2n^2+mn^\omega+m^\omega)$ and $(mn)^{1/4}(m^4n^2+m^3n^\omega)$, respectively, which are slower in the most common regime of $m \in [n, n^2]$ (see Table 1.2). This apparent paradox raises the following natural question:

*How fast can SDPs be solved using interior point methods?*

[1] We can assume that $C, A_1, \cdots, A_m$ are symmetric, since given any $M \in \{C, A_1, \cdots, A_m\}$, we have $\sum_{i,j} M_{ij} X_{ij} = \sum_{i,j} M_{ij} X_{ji} = \sum_{i,j} (M^\top)_{ij} X_{ij}$, and therefore we can replace $M$ with $(M + M^\top)/2$.

[2] [JLSW20] improves upon the runtime of [LSW15] in terms of the dependence on $\log(n/\epsilon)$, while the polynomial factors are the same in both runtimes.

## 1.1 Our results

We present a faster interior point method for solving SDPs. Our main result is the following theorem, the formal version of which is given in Theorem 4.1.

**Theorem 1.2** (Main result, informal)**.** *There is an interior point method that solves a general SDP with variable size $n \times n$ and $m$ constraints in time*[3] $O^*(\sqrt{n}(mn^2 + m^\omega + n^\omega))$.

Our runtime can be roughly interpreted as follows:

- $\sqrt{n}$ is the iteration complexity of the interior point method with the log barrier function.
- $mn^2$ is the input size.
- $m^\omega$ is the cost of inverting the Hessian of the log barrier.
- $n^\omega$ is the cost of inverting the slack matrix.

Thus, the terms in the runtime of our algorithm arise as a natural barrier to further speeding up SDP solvers. See Section 1.2.2, 1.2.3, and 1.2.4 for more detail.

Table 1.1 compares our result with previous SDP solvers. The first takeaway of this table and Theorem 1.2 is that our interior point method always runs faster than that in [NN92] and is faster than that in [NN94] and [Ans00] when $m \geq n^{1/13}$. A second consequence is that whenever $m \geq \sqrt{n}$, our interior point method is faster than the current fastest cutting plane method [LSW15, JLSW20]. We note that $n \leq m \leq n^2$ is satisfied in most SDP applications known to us, such as classical combinatorial optimization problems over graphs, experiment design problems in statistics and machine learning, and sum-of-squares problems. An explicit comparison to previous algorithms in the cases of $m = n$ and $m = n^2$ is shown in Table 1.2.

| Year | References | Method | #Iters | Cost per iter |
|---|---|---|---|---|
| 1979 | [Sho77, YN76, Kha80] | CPM | $m^2$ | $mn^2 + m^2 + n^\omega$ |
| 1988 | [KTE88, NN89] | CPM | $m$ | $mn^2 + m^{3.5} + n^\omega$ |
| 1989 | [Vai89a] | CPM | $m$ | $mn^2 + m^\omega + n^\omega$ |
| 1992 | [NN92] | IPM | $\sqrt{n}$ | $m^2n^2 + mn^\omega + m^\omega$ |
| 1994 | [NN94, Ans00] | IPM | $(mn)^{1/4}$ | $m^4n^2 + m^3n^\omega$ |
| 2003 | [KM03] | CPM | $m$ | $mn^2 + m^\omega + n^\omega$ |
| 2015 | [LSW15] | CPM | $m$ | $mn^2 + m^2 + n^\omega$ |
| 2020 | [JLSW20] | CPM | $m$ | $mn^2 + m^2 + n^\omega$ |
| 2020 | Our result | IPM | $\sqrt{n}$ | $mn^2 + m^\omega + n^\omega$ |

Table 1.1: Summary of key SDP algorithms. CPM stands for cutting plane method, and IPM, interior point method. $n$ is the size of the variable matrix, and $m \leq n^2$ is the number of constraints. Runtimes hide $n^{o(1)}$, $m^{o(1)}$ and $\mathrm{poly}\log(1/\epsilon)$ factors, where $\epsilon$ is the accuracy parameter. [Ans00] simplifies the proofs in [NN94, Section 5.5]. Neither [Ans00] nor [NN94] explicitly analyzed their runtimes, and their runtimes shown here are our best estimates.

Even in the more general case where the SDP might not be dense, where $\mathrm{nnz}(A)$ is the input size (i.e., the total number of non-zeroes in all matrices $A_i$ for $i \in [m]$ and $C$), our interior point method runs faster than the current fastest cutting plane methods[LSW15, JLSW20], which run in time $O^*(m(\mathrm{nnz}(A) + m^2 + n^\omega))$.

[3]We use $O^*$ to hide $n^{o(1)}$ and $\log^{O(1)}(n/\epsilon)$ factors and $\widetilde{O}$ to hide $\log^{O(1)}(n/\epsilon)$ factors, where $\epsilon$ is the accuracy parameter.

| Year | References | Method | Runtime | |
|---|---|---|---|---|
| | | | $m = n$ | $m = n^2$ |
| 1979 | [Sho77, YN76, Kha80] | CPM | $n^5$ | $n^8$ |
| 1988 | [KTE88, NN89] | CPM | $n^{4.5}$ | $n^9$ |
| 1989 | [Vai89a] | CPM | $n^4$ | $n^{6.746}$ |
| 1992 | [NN92] | IPM | $n^{4.5}$ | $n^{6.5}$ |
| 1994 | [NN94, Ans00] | IPM | $n^{6.5}$ | $n^{10.75}$ |
| 2003 | [KM03] | CPM | $n^4$ | $n^{6.746}$ |
| 2015 | [LSW15] | CPM | $n^4$ | $n^6$ |
| 2020 | [JLSW20] | CPM | $n^4$ | $n^6$ |
| 2020 | Our result | IPM | $n^{3.5}$ | $n^{5.246}$ |

Table 1.2: Total runtimes for the algorithms in Table 1.1 for SDPs when $m = n$ and $m = n^2$, where $n$ is the size of matrices, and $m$ is the number of constraints. The runtimes shown in the table hide $n^{o(1)}$, $m^{o(1)}$ and $\text{poly}\log(1/\epsilon)$ factors, where $\epsilon$ is the accuracy parameter and assume $\omega$ to equal its currently best known upper bound of 2.373.

**Theorem 1.3** (Comparison with Cutting Plane Method)**.** *When $m \geq n$, there is an interior point method that solves an SDP with $n \times n$ matrices, $m$ constraints, and $\text{nnz}(A)$ input size, faster than the current best cutting plane method [LSW15, JLSW20], over all regimes of $\text{nnz}(A)$.*

## 1.2 Technique overview

### 1.2.1 Interior point method for solving SDPs

By removing redundant constraints, we can, without loss of generality, assume $m \leq n^2$ in the primal formulation of the SDP (1). Thereafter, instead of solving the primal SDP, which has variable size $n \times n$, we solve its dual formulation, which has dimension $m \leq n^2$:

$$\min b^\top y \text{ subject to } S = \sum_{i=1}^{m} y_i A_i - C, \text{ and } S \succeq 0. \tag{2}$$

Interior point methods solve (2) by minimizing the penalized objective function:

$$\min_{y \in \mathbb{R}^m} f_\eta(y), \text{ where } f_\eta(y) := \eta \cdot b^\top y + \phi(y), \tag{3}$$

where $\eta > 0$ is a parameter and $\phi : \mathbb{R}^m \to \mathbb{R}$ is a *barrier* function that approaches infinity as $y$ approaches the boundary of the feasible set $\{y \in \mathbb{R}^m : \sum_{i=1}^m y_i A_i \succeq C\}$. These methods first obtain an approximate minimizer of $f_\eta$ for some small $\eta > 0$, which they then use as an initial point to minimize $f_{(1+c)\eta}$, for some constant $c > 0$, via the Newton method. This process repeats until the parameter $\eta$ in (3) becomes sufficiently large, at which point the minimizer of $f_\eta$ is provably close to the optimal solution of (2). The iterates $y$ generated by this method follow a *central path*. Different choices of the barrier function $\phi$ lead to different run times in solving (3), as we next describe.

**The log barrier** Nesterov and Nemirovski [NN92] use the *log barrier* function,

$$\phi(y) = g(y) := -\log\det\left(\sum_{i=1}^{m} y_i A_i - C\right), \tag{4}$$

in (3) and, in $O(\sqrt{n}\log(n/\epsilon))$ iterations, obtain a feasible dual solution $y$ that satisfies $b^\top y \leq b^\top y^* + \epsilon$, where $y^* \in \mathbb{R}^m$ is the optimal solution for (2). Within each iteration, the costliest step

is to compute the inverse of the Hessian of the log barrier function for the Newton step. For each $(j,k) \in [m] \times [m]$, the $(j,k)$-th entry of $H$ is given by

$$H_{j,k} = \mathrm{tr}[S^{-1}A_jS^{-1}A_k]. \tag{5}$$

The analysis of [NN92] first computes $S^{-1/2}A_jS^{-1/2}$ for all $j \in [m]$, which takes time $O^*(mn^\omega)$, and then calculates the $m^2$ trace products $\mathrm{tr}[S^{-1}A_jS^{-1}A_k]$ for all $(j,k) \in [m] \times [m]$, each of which takes $O(n^2)$ time. Inverting the Hessian costs $O^*(m^\omega)$, which results in a total runtime of $O^*(\sqrt{n}(m^2n^2 + mn^\omega + m^\omega))$.

**The volumetric barrier** Vaidya [Vai89a] introduced the *volumetric barrier* for a polyhedral set $\{x \in \mathbb{R}^n : Ax \geq c\}$, where $A \in \mathbb{R}^{m\times n}$ and $c \in \mathbb{R}^m$. Nesterov and Nemirovski [NN94] studied the following extension of the volumetric barrier to the convex subset $\{y \in \mathbb{R}^m : \sum_{i=1}^m y_iA_i \succeq C\}$ of the polyhedral cone:

$$V(y) = \frac{1}{2}\log\det(\nabla^2 g(y)),$$

where $g(y)$ is the log barrier function defined in (4). They proved that choosing $\phi(y) = \sqrt{n}V(y)$ in (3) makes the interior point method converge in $\widetilde{O}(\sqrt{m}n^{1/4})$ iterations, which is smaller than the $\widetilde{O}(\sqrt{n})$ iteration complexity of [NN92] when $m \leq \sqrt{n}$. They also studied the *combined volumetric-logarithmic barrier*

$$V_\rho(y) = V(y) + \rho \cdot g(y)$$

and showed that taking $\phi(y) = \sqrt{n/m} \cdot V_\rho(y)$ for $\rho = (m-1)/(n-1)$ yields an iteration complexity of $\widetilde{O}((mn)^{1/4})$. when $m \leq n$, this iteration complexity is lower than $\widetilde{O}(\sqrt{n})$ of [NN92]. We refer readers to the much simpler proofs in [Ans00] for these results.

However, the volumetric barrier (and thus the combined volumetric-logarithmic barrier) leads to complicated expressions for the gradient and Hessian that make each iteration costly. For instance, the Hessian of the volumetric barrier is

$$\nabla^2 V(y) = 2Q(y) + R(y) - 2T(y),$$

where $Q(y)$, $R(y)$, and $T(y)$ are $m \times m$ matrices such that for each $(j,k) \in [m] \times [m]$,

$$\begin{aligned}
Q(y)_{j,k} &= \mathrm{tr}\left[\mathcal{A}H^{-1}\mathcal{A}^\top\left(\left(S^{-1}A_jS^{-1}A_kS^{-1}\right)\widehat{\otimes}S^{-1}\right)\right], \\
R(y)_{j,k} &= \mathrm{tr}\left[\mathcal{A}H^{-1}\mathcal{A}^\top\left(\left(S^{-1}A_jS^{-1}\right)\widehat{\otimes}\left(S^{-1}A_kS^{-1}\right)\right)\right], \\
T(y)_{j,k} &= \mathrm{tr}\left[\mathcal{A}H^{-1}\mathcal{A}^\top\left(\left(S^{-1}A_jS^{-1}\right)\widehat{\otimes}S^{-1}\right)\mathcal{A}H^{-1}\mathcal{A}^\top\left(\left(S^{-1}A_kS^{-1}\right)\widehat{\otimes}S^{-1}\right)\right].
\end{aligned} \tag{6}$$

Here, $\mathcal{A} \in \mathbb{R}^{n^2\times m}$ is the $n^2 \times m$ matrix whose $i$th column is obtained by flattening $A_i$ into a vector of length $n^2$, and $\widehat{\otimes}$ is the symmetric Kronecker product

$$A\widehat{\otimes}B := \frac{1}{2}(A \otimes B + B \otimes A),$$

where $\otimes$ is the Kronecker product (see Section 2.1for formal definition). Due to the complicated formulas in (6), efficient computation of Newton step in each iteration of the interior point method is difficult; in fact, each iteration runs slower than the Nesterov-Nemirovski interior point method by a factor of $m^2$. Since most applications of SDPs known to us have the number of constraints $m$ be at least linear in $n$, the total runtime of interior point methods based on the volumetric barrier and the combined volumetric-logarithmic barrier is inevitably slow.

### 1.2.2 Our techniques

Given the inefficiency of implementing the volumetric and volumetric-logarithmic barriers discussed above, this paper uses the log barrier in (4). We now describe some of our key techniques that improve the runtime of the Nesterov-Nemirovski interior point method [NN92].

**Hessian computation using fast rectangular matrix multiplication** As noted in Section 1.2.1, the runtime bottleneck in [NN92] is computing the inverse of the Hessian of the log barrier function, where the Hessian is described in (5). In [NN92], each of these $m^2$ entries is computed separately, resulting in a runtime of $O(m^2n^2)$ per iteration.

Instead contrast, we show below how to group these computations using rectangular matrix multiplication. The expression from (5) can be re-written as

$$H_{j,k} = \mathrm{tr}[S^{-1/2}A_jS^{-1/2} \cdot S^{-1/2}A_kS^{-1/2}]. \tag{7}$$

We first compute the key quantity $S^{-1/2}A_jS^{-1/2} \in \mathbb{R}^{n\times n}$ for all $j \in [m]$ by stacking all matrices $A_j \in \mathbb{R}^{n\times n}$ into a tall matrix of size $mn\times n$, and then compute the product of $S^{-1/2} \in \mathbb{R}^{n\times n}$ with this tall matrix. This matrix product can be computed in time $\mathcal{T}_{\mathrm{mat}}(n, mn, n)$[4] using fast rectangular matrix multiplication. We then flatten each $S^{-1/2}A_jS^{-1/2}$ into a row vector of length $n^2$ and stack all $m$ vectors to form a matrix $\mathcal{B}$ of size $m \times n^2$, i.e., the $j$-th row of $\mathcal{B}$ is $\mathcal{B}_j = \mathsf{vec}(S^{-1/2}A_jS^{-1/2})$. It follows that the Hessian can be computed as

$$H = \mathcal{B}\mathcal{B}^\top, \tag{8}$$

which takes time $\mathcal{T}_{\mathrm{mat}}(m, n^2, m)$ by applying fast rectangular matrix multiplication. By leveraging recent developments in this area [GU18], this approach already improves upon the runtime in [NN92].

Thus far, we have reduced the per iteration cost of $O^*(m^2n^2 + mn^\omega)$ for Hessian computation down to

$$\mathcal{T}_{\mathrm{mat}}(n, mn, n) + \mathcal{T}_{\mathrm{mat}}(m, n^2, m).$$

**Low rank update on the slack matrix** The fast rectangular matrix multiplication approach noted above, however, is still not very efficient, because the Hessian must be computed from scratch in each iteration of the interior point method. If there are $T$ iterations in total, it then takes time

$$T \cdot (\mathcal{T}_{\mathrm{mat}}(n, mn, n) + \mathcal{T}_{\mathrm{mat}}(m, n^2, m)).$$

To further improve the runtime, we need to efficiently update the Hessian for the current iteration from the Hessian computed in the previous one. Generally, this is not possible, as the slack matrix $S \in \mathbb{R}^{n\times n}$ in (7) might change arbitrarily in the Nesterov-Nemirovski interior point method.

To overcome this problem, we propose a new interior point method that maintains an approximate slack matrix $\widetilde{S} \in \mathbb{R}^{n\times n}$, which is a spectral approximation of the true slack matrix $S \in \mathbb{R}^{n\times n}$ such that $\widetilde{S}$ admits a *low-rank update* in each iteration. Where needed, we will now use the subscript $t$ to denote a matrix in the $t$-th iteration. Our algorithm updates only the directions in which $\widetilde{S}_t$ deviates too much from $S_{t+1}$; the changes to $S_t$ for the remaining directions are not propagated in $\widetilde{S}_t$. This process of selective update ensures a low-rank change in $\widetilde{S}_t$ even when $S_t$ suffers from a

[4] See Section 3 for the definition.

full-rank update; it also guarantees the proximity of the algorithm's iterates to the central path. Specifically, for each iteration $t \in [T]$, we define the *difference matrix*

$$Z_t = S_t^{-1/2} \widetilde{S}_t S_t^{-1/2} - I \quad \in \mathbb{R}^{n \times n},$$

which intuitively captures how far the approximate slack matrix $\widetilde{S}_t$ is from the true slack matrix $S_t$. We maintain the invariant $\|Z_t\|_{\mathrm{op}} \leq c$ for some sufficiently small constant $c > 0$. In the $(t+1)$-th iteration when $S_t$ gets updated to $S_{t+1}$, our construction of $\widetilde{S}_{t+1}$ involves a novel approach of zeroing out some of the largest eigenvalues of $|Z_t|$ to bound the rank of the update on the approximate slack matrix.

We prove that with this approach, the updates on $\widetilde{S} \in \mathbb{R}^{n \times n}$ over all $T = \widetilde{O}(\sqrt{n})$ iterations satisfy the following *rank inequality* (see Theorem 6.1for the formal statement).

**Theorem 1.4** (Rank inequality, informal version)**.** *Let $\widetilde{S}_1, \widetilde{S}_2, \cdots, \widetilde{S}_T \in \mathbb{R}^{n \times n}$ denote the sequence of approximate slack matrices generated in our interior point method. For each $t \in [T-1]$, denote by $r_t = \mathrm{rank}(\widetilde{S}_{t+1} - \widetilde{S}_t)$ the rank of the update on $\widetilde{S}_t$. Then, the sequence $r_1, r_2, \cdots, r_T$ satisfies*

$$\sum_{t=1}^{T} \sqrt{r_t} = \widetilde{O}(T).$$

The key component to proving Theorem 1.4 is the potential function $\Phi : \mathbb{R}^{n \times n} \to \mathbb{R}_{\geq 0}$

$$\Phi(Z) := \sum_{\ell=1}^{n} \frac{|\lambda(Z)|_{[\ell]}}{\sqrt{\ell}},$$

where $|\lambda(Z)|_{[\ell]}$ is the $\ell$-th in the list of eigenvalues of $Z \in \mathbb{R}^{n \times n}$ sorted in decreasing order of their absolute values. We show an upper bound on the increase in this potential when $S$ is updated, a lower bound on its decrease when $\widetilde{S}$ is updated, and combine the two with non-negativity of the potential to obtain Theorem 1.4.

Specifically, first we prove that whenever $S$ is updated in an iteration, the potential function *increases* by at most $\widetilde{O}(1)$ (see Lemma 6.2). The proof of this statement crucially uses the structural property of interior point method that slack matrices in consecutive steps are sufficiently close to each other. Formally, for any iteration $t \in [T]$, we show in Theorem 5.1 that the consecutive slack matrices $S_t$ and $S_{t+1}$ satisfy

$$\|S_t^{-1/2} S_{t+1} S_t^{-1/2} - I\|_F = O(1) \tag{9}$$

and combine this bound with the Hoffman-Wielandt theorem [HJ12], which relates the $\ell_2$ distance between the spectrum of two matrices with the Frobenius norm of their difference (see Fact 2.2). Next, when $\widetilde{S}$ gets updated, we prove that our method of zeroing out the $r_t$ largest eigenvalues of $|Z_t|$, thereby incurring a rank-$r_t$ update to $\widetilde{S}_t$, results in a potential decrease of at least $\widetilde{O}(\sqrt{r_t})$ (see Lemma 6.3).

**Maintaining rectangular matrix multiplication for Hessian computation.** Given the low-rank update on $\widetilde{S}$ described above, we show how to efficiently update the *approximate Hessian* $\widetilde{H}$, defined as

$$\widetilde{H}_{j,k} = \mathrm{tr}[\widetilde{S}^{-1} A_j \widetilde{S}^{-1} A_k] \tag{10}$$

for each entry $(j,k) \in [m] \times [m]$. The approximate slack matrix $\widetilde{S}$ being a spectral approximation of the true slack matrix $S$ implies that the approximate Hessian $\widetilde{H}$ is also a spectral approximation of the true Hessian $H$ (see Lemma 5.3). This approximate Hessian therefore suffices for our algorithm to approximately follow the central path.

To efficiently update the approximate Hessian $\widetilde{H}$ in (10), we notice that a rank-$r$ update on $\widetilde{S}$ implies a rank-$r$ update on $\widetilde{S}^{-1}$ via the Woodbury matrix identity (see Fact 2.4). The change in $\widetilde{S}^{-1}$ can be expressed as

$$\Delta(\widetilde{S}^{-1}) = V_+ V_+^\top - V_- V_-^\top, \tag{11}$$

where $V_+, V_- \in \mathbb{R}^{n \times r}$. Plugging (11) into (10), we can express $\Delta \widetilde{H}_{j,k}$ as the sum of multiple terms, among the costliest of which are those of the form $\mathrm{tr}[\widetilde{S}^{-1} A_j V V^\top A_k]$, where $V \in \mathbb{R}^{n \times r}$ is either $V_+$ or $V_-$. We compute $\mathrm{tr}[\widetilde{S}^{-1} A_j V V^\top A_k]$ for all $(j,k) \in [m] \times [m]$ in time $\mathcal{T}_{\mathrm{mat}}(r, n, mn)$ by first computing $V^\top A_k$ for all $k \in [m]$ by horizontally concatenating all $A_k$'s into a wide matrix of size $n \times mn$. We then compute the product of $\widetilde{S}^{-1/2}$ with $A_j V$ for all $j \in [m]$, which can be done in time $\mathcal{T}_{\mathrm{mat}}(n, n, mr)$, which equals $\mathcal{T}_{\mathrm{mat}}(n, mr, n)$ (see Lemma 3.3). Finally, by flattening each $\widetilde{S}^{-1/2} A_j V$ into a vector of length $nr$ and stacking all these vectors to form a matrix $\widetilde{\mathcal{B}} \in \mathbb{R}^{m \times nr}$ with $j$-th row

$$\widetilde{\mathcal{B}}_j = \mathsf{vec}(\widetilde{S}^{-1/2} A_j V),$$

the task of computing $\mathrm{tr}[\widetilde{S}^{-1} A_j V V^\top A_k]$ for all $(j,k) \in [m] \times [m]$ reduces to computing $\widetilde{\mathcal{B}} \widetilde{\mathcal{B}}^\top$, which costs $\mathcal{T}_{\mathrm{mat}}(m, nr, m)$.

In this way, we reduce the runtime of $T \cdot (\mathcal{T}_{\mathrm{mat}}(n, mn, n) + \mathcal{T}_{\mathrm{mat}}(m, n^2, m))$ for computing the Hessian using fast rectangular matrix multiplication down to

$$\sum_{t=1}^{T} \left( \mathcal{T}_{\mathrm{mat}}(r_t, n, mn) + \mathcal{T}_{\mathrm{mat}}(n, mr_t, n) + \mathcal{T}_{\mathrm{mat}}(m, nr_t, m) \right), \tag{12}$$

where $r_t$ is the rank of the update on $\widetilde{S}_t$. Applying Theorem 1.4 with several properties of fast rectangular matrix multiplication that we prove in Section 3 , we upper bound the runtime in (12) by

$$O^*(\sqrt{n}(mn^2 + m^\omega + n^\omega)),$$

which implies Theorem 1.2. In Section 1.2.3 and 1.2.4, we discuss bottlenecks to further improving our runtime.

### 1.2.3 Bottlenecks of our interior point method

In most cases, the costliest term in our runtime is the per iteration cost of $mn^2$, which corresponds to reading the entire input in each iteration. Our subsequent discussions therefore focus on the steps in our algorithm that require at least $mn^2$ time per iteration.

**Slack matrix computation.** When $y$ is updated in each iteration of our interior point method, we need to compute the true slack matrix $S$ as

$$S = \sum_{i \in [m]} y_i A_i - C.$$

Computing $S$ is needed to update the approximate slack matrix $\widetilde{S}$ so that $\widetilde{S}$ remains a spectral approximation to $S$. As $S$ might suffer from full-rank changes, it naturally requires $mn^2$ time to compute in each iteration. This is the first appearance of the $mn^2$ cost per iteration.

**Gradient computation.** Recall from (3) that our interior point method follows the central path defined via the penalized objective function

$$\min_{y \in \mathbb{R}^m} f_\eta(y) \quad \text{where} \quad f_\eta(y) := \eta b^\top y + \phi(y),$$

for a parameter $\eta > 0$ and $\phi(y) = -\log\det S$. In each iteration, to perform the Newton step, the gradient of the penalized objective is computed as

$$g_\eta(y)_j = \eta \cdot b_j - \mathrm{tr}[S^{-1} A_j] \tag{13}$$

for each coordinate $j \in [m]$. Even if we are given $S^{-1}$, it still requires $mn^2$ time to compute (13) for all $j \in [m]$. This is the second appearance of the per iteration cost of $mn^2$.

**Approximate Hessian computation.** Recall from Section 1.2.2 that updating the approximate slack matrix $S$ by rank $r$ means the time needed to update the approximate Hessian is dominated by computing the term

$$\Delta_{j,k} = \mathrm{tr}[\widetilde{S}^{-1/2} A_j V \cdot V^\top A_k \widetilde{S}^{-1/2}],$$

where $V \in \mathbb{R}^{n \times r}$ is a tall, skinny matrix that comes from the spectral decomposition of $\Delta \widetilde{S}^{-1}$. Computing $\Delta_{j,k}$ for all $(j,k) \in [m] \times [m]$ requires reading at least $A_j$ for all $j \in [m]$, which takes time $mn^2$. This is the third bottleneck that leads to the $mn^2$ term in the cost per iteration.

### 1.2.4 LP techniques unlikely to improve SDP runtime

The preceeding discussion of bottlenecks suggests that reading the entire input in each iteration, which takes $mn^2$ time per iteration, stands as a natural barrier to further improving the runtime of SDP solvers based on interior point methods.

In the context of linear programming (LP), several recent results [CLS19, BLSS20] yield faster interior point methods that bypass reading the entire input in every iteration. Two techniques crucial to these results are: (1) showing that the Hessian (projection matrix) admits low-rank updates, and (2) speeding computation of the Hessian via sampling.

We now describe these techniques in the context of SDP and argue that they are unlikely to improve our runtime.

**Showing that the Hessian admits low-rank updates.** We saw in Section 1.2.2 that constructing an approximate slack matrix $\widetilde{S}$ that admits low-rank updates in each iterations leveraged the fact that the true slack matrix $S$ changes "slowly" throughout our interior point method as described in (9). One natural question that follows is whether a similar upper bound can be obtained for the Hessian. If such a result could be proved, then one could maintain an approximate Hessian that admitted low-rank updates, which would speed up the approximate Hessian computation. Indeed, in the context of LP, such a bound for the Hessian can be proved (e.g., [BLSS20, Lemma 47]).

Unfortunately, it is impossible to prove such a statement for the Hessian in the context of SDP. To show this, it is convenient to express the Hessian using the Kronecker product (Section 2.1)as

$$H = \mathcal{A}^\top \cdot (S^{-1} \otimes S^{-1}) \cdot \mathcal{A},$$

where $\mathcal{A} \in \mathbb{R}^{n^2 \times m}$ is the $n^2 \times m$ matrix whose $i$th column is obtained by flattening $A_i$ into a vector of length $n^2$. By proper scaling, we can assume without loss of generality that the current slack matrix

is $S = I$, and the slack matrix in the next iteration is $S_{\text{new}} = I + \Delta S$, which satisfies $\|\Delta S\|_F = c$ for some tiny constant $c > 0$. Consider the simple example where $\mathcal{A} = I$ (we are assuming here that $m = n^2$ so that $\mathcal{A}$ is a square matrix), which implies that the change in the Hessian can be approximately computed as

$$\begin{aligned}\left\|H^{-1/2}\Delta H H^{-1/2}\right\|_F^2 &\approx \mathrm{tr}\left[((I-\Delta S)\otimes(I-\Delta S) - I\otimes I)^2\right] \\ &\approx \mathrm{tr}\left[(I\otimes\Delta S + \Delta S\otimes I)^2\right] \\ &\geq 2\cdot \mathrm{tr}[I^2]\cdot \mathrm{tr}\left[(\Delta S)^2\right] \\ &= 2n\left\|\Delta S\right\|_F^2 \gg 1.\end{aligned}$$

This large change indicates that we are unlikely to obtain an approximation to the Hessian that admits low-rank updates, which is a key difference between LP and SDP.

**Sampling for faster Hessian computation.** Recall from (8) that the Hessian can be computed as

$$H = \mathcal{B}\cdot\mathcal{B}^\top,$$

where the $j$th row of $\mathcal{B} \in \mathbb{R}^{m\times n^2}$ is $\mathcal{B}_j = \mathsf{vec}(S^{-1/2}A_jS^{-1/2})$ for all $j \in [m]$. We might attempt to approximately compute $H$ faster by sampling a subset of columns of $\mathcal{B}$ indexed by $L \subseteq [n^2]$ and compute the product for only the sampled columns. This could reduce the dimension of the matrix multiplication and speed up the Hessian computation. Indeed, sampling techniques have been successfully used to obtain faster LP solvers [CLS19, BLSS20].

For SDP, however, sampling is unlikely to speed up the Hessian computation. In general, we must sample at least $m$ columns (i.e. $|L| \geq m$) of $\mathcal{B}$ to spectrally approximate $H$ or the computed matrix will not be full rank. However, this requires computing the entries of $S^{-1/2}A_jS^{-1/2}$ that correspond to $L \subseteq [n^2]$ for all $j \in [m]$, which requires reading all $A_j$'s and thus still takes $O(mn^2)$ time.

### 1.3 Related work

**Linear Programming.** Linear Programming is a class of fundamental problems in convex optimization. There is a long list of work focused on fast algorithms for linear programming [Dan47, Kha80, Kar84, Vai87, Vai89b, LS14, LS15, Sid15, Lee16, CLS19, Bra20, BLSS20].

**Cutting Plane Method.** Cutting plane method is a class of optimization methods that iteratively refine a convex set that contains the optimal solution by querying a separation oracle. Since its introduction in the 1950s, there has been a long line of work on obtaining fast cutting plane methods [Sho77, YN76, Kha80, KTE88, NN89, Vai89a, AV95, BV02, LSW15, JLSW20].

**First-Order SDP Algorithms.** As the focus of this paper, cutting plane methods and interior point methods solve SDPs in time that depends *logarithmically* on $1/\epsilon$, where $\epsilon$ is the accuracy parameter. A third class of algorithms, the *first-order methods*, solve SDPs at runtimes that depend *polynomially* on $1/\epsilon$. While having worse dependence on $1/\epsilon$ compared to IPM and CPM, these first-order algorithms usually have better dependence on the dimension. There is a long list of work on first-order methods for general SDP or special classes of SDP (e.g. Max-Cut SDP [AK07, GH16, AZL17, CDST19, LP20, YTF+19], positive SDPs [JY11, PT12, ALO16, JLL+20].)

## 2 Preliminaries

### 2.1 Notation

For any integer $d$, we use $[d]$ to denote the set $\{1, 2, \cdots, d\}$. We use $\mathbb{S}^{n\times n}$ to denote the set of symmetric $n \times n$ matrices, $\mathbb{S}^{n\times n}_{\succeq 0}$ for the set of $n \times n$ positive semidefinite matrices, and $\mathbb{S}^{n\times n}_{\succ 0}$ for the set of $n \times n$ positive definite matrices. For two matrices $A, B \in \mathbb{S}^{n\times n}$, the notation $A \preceq B$ means that $B - A \in \mathbb{S}^{n\times n}_{\succeq 0}$. When clear from the context, we use 0 to denote the all-zeroes matrix (e.g. $A \succeq 0$). For a vector $v \in \mathbb{R}^n$, we use $\mathrm{diag}(v)$ to denote the diagonal $n \times n$ matrix with $\mathrm{diag}(v)_{i,i} = v_i$. For $A, B \in \mathbb{S}^{n\times n}$, we define the inner product to be the trace product of $A$ and $B$, defined as $\langle A, B\rangle := \mathrm{tr}[A^\top B] = \sum_{i,j\in[n]} A_{i,j}B_{i,j}$. For two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{k\times l}$, the *Kronecker product* of $A$ and $B$, denoted as $A \otimes B$, is defined as the $mk \times nl$ block matrix whose $(i, j)$ block is $A_{i,j}B$, for all $(i, j) \in [m] \times [n]$.

Throughout this paper, unless otherwise specified, $m$ denotes the number of constraints for the primal SDP (1), and the variable matrix $X$ is of size $n \times n$. The number of non-zero entries in all the $A_i$ and $C$ of (1) is denoted by $\mathrm{nnz}(A)$.

### 2.2 Useful facts

**Linear algebra.** Some matrix norms we frequently use in this paper are the Frobenius and operator norms, defined as follows. The Frobenius norm of a matrix $A \in \mathbb{R}^{n\times n}$ is defined to be $\|A\|_F := \sqrt{\mathrm{tr}[A^\top A]}$. The operator (or spectral) norm $\|A\|_{\mathrm{op}}$ of $A \in \mathbb{R}^{n\times n}$ is defined to be the largest singular value of $A$. In the case of symmetric matrices (which is what we encounter in this paper), this can be shown to equal the largest absolute eigenvalue of the matrix. A property of trace we frequently use is the following: given matrices $A_1 \in \mathbb{R}^{m\times n_1}, A_2 \in \mathbb{R}^{n_1\times n_2}, \ldots, A_k \in \mathbb{R}^{n_{k-1}\times n_k}$, the trace of their product is invariant under cyclic permutation $\mathrm{tr}[A_1A_2 \ldots A_k] = \mathrm{tr}[A_2A_3 \ldots A_kA_1] = \cdots = \mathrm{tr}[A_kA_1 \ldots A_{k-2}A_{k-1}]$. A matrix $A \in \mathbb{R}^{n\times n}$ is called *normal* if $A$ commutes with its transpose, i.e. $AA^\top = A^\top A$. We note that all symmetric $n\times n$ matrices are normal. Two matrices $A, B \in \mathbb{R}^{n\times n}$ are said to be similar if there exists a nonsingular matrix $S \in \mathbb{R}^{n\times n}$ such that $A = S^{-1}BS$. In particular, if matrices $A$ and $B$ are similar, then they have the same set of eigenvalues. We use the following simple fact involving Loewner ordering: given two invertible matrices $A$ and $B$ satisfying $\frac{1}{\alpha}B \preceq A \preceq \alpha B$ for some $\alpha > 0$, we have $\frac{1}{\alpha}B^{-1} \preceq A^{-1} \preceq \alpha B^{-1}$. We further need the following facts.

**Fact 2.1** (Generalized Lieb-Thirring Inequality [Eld13, ALO16, JLL+20])**.** *Given a symmetric matrix $B$, a positive semi-definite matrix $A$ and $\alpha \in [0, 1]$, we have*

$$\mathrm{tr}[A^\alpha BA^{1-\alpha}B] \le \mathrm{tr}[AB^2].$$

**Fact 2.2** (Hoffman-Wielandt Theorem, [HW53, HJ12])**.** *Let $A, E \in \mathbb{R}^{n\times n}$ such that $A$ and $A + E$ are both normal matrices. Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of $A$, and let $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n$ be the eigenvalues of $A + E$ in any order. There is a permutation $\sigma$ of the integers $1, \ldots, n$ such that $\sum_{i\in[n]} |\widehat{\lambda}_{\sigma(i)} - \lambda_i|^2 \le \|E\|_F^2 = \mathrm{tr}[E^*E]$.*

**Fact 2.3** (Corollary of the Hoffman-Wielandt Theorem, [HJ12])**.** *Let $A, E \in \mathbb{R}^{n\times n}$ such that $A$ is Hermitian and $A+E$ is normal. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $A$ arranged in increasing order $\lambda_1 \le \ldots \le \lambda_n$. Let $\widehat{\lambda}_1, \ldots, \widehat{\lambda}_n$ be the eigenvalues of $A + E$, ordered so that $\mathrm{Re}(\widehat{\lambda}_1) \le \ldots \le \mathrm{Re}(\widehat{\lambda}_n)$. Then, $\sum_{i\in[n]} |\widehat{\lambda}_i - \lambda_i|^2 \le \|E\|_F^2$.*

**Fact 2.4** (Woodbury matrix identity, [Woo49, Woo50])**.** *Given matrices $A \in \mathbb{R}^{n\times n}$, $U \in \mathbb{R}^{n\times k}$, $C \in \mathbb{R}^{k\times k}$, and $V \in \mathbb{R}^{k\times n}$, such that $A$, $C$, and $A + UCV$ are invertible, we have*

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}.$$

# 3 Matrix Multiplication

The main goal of this section is to derive upper bounds on the time to perform the following two rectangular matrix multiplication tasks (Lemma 3.9, 3.10, and 3.11):

- Multiplying a matrix of dimensions $m \times n^2$ with one of dimensions $n^2 \times m$,
- Multiplying a matrix of dimensions $n \times mn$ with one of dimensions $mn \times n$.

Besides being crucial to the runtime analysis of our interior point method in Section 7, these results (as well as several intermediate results) might be of independent interest.

## 3.1 Exponent of matrix multiplication

We need the following definitions to describe the cost of certain fundamental matrix operations we use.

**Definition 3.1.** *Define $\mathcal{T}_{\text{mat}}(n, r, m)$ to be the number of operations needed to compute the product of matrices of dimensions $n \times r$ and $r \times m$.*

**Definition 3.2.** *We define the function $\omega(k)$ to be the minimum value such that $\mathcal{T}_{\text{mat}}(n, n^k, n) = n^{\omega(k)+o(1)}$. We overload notation and use $\omega$ to denote the cost of multiplying two $n \times n$ matrices. Thus, we have $\omega(1) = \omega$.*

The following is a basic property of $\mathcal{T}_{\text{mat}}$ that we frequently use.

**Lemma 3.3** ([BCS97, Blä13])**.** *For any three positive integers $n, m, r$, we have*

$$\mathcal{T}_{\text{mat}}(n, r, m) = O(\mathcal{T}_{\text{mat}}(n, m, r)) = O(\mathcal{T}_{\text{mat}}(m, n, r)).$$

We refer to Table 3 in [GU18] for the latest upper bounds on $\omega(k)$ for different values of $k$. In particular, we need the following upper bounds in our paper.

**Lemma 3.4** ([GU18])**.** *We have:*

- $\omega = \omega(1) \leq 2.372927$,
- $\omega(1.5) \leq 2.79654$,
- $\omega(1.75) \leq 3.02159$,
- $\omega(2) \leq 3.251640$.

## 3.2 Technical results for matrix multiplication

In this section, we derive some technical results on $\mathcal{T}_{\text{mat}}$ and $\omega$ that we extensively use for our runtime analysis. Some of these results can be derived using tensors, and we demonstrate this in Appendix A. We hope that the use of tensors can yield better runtimes for this problem in future.

**Lemma 3.5** (Sub-linearity)**.** *For any $p \geq q \geq 1$, we have*

$$\omega(p) \leq p - q + \omega(q).$$

*Proof.* We assume that $n^p$ and $n^q$ are integers for notational simplicity. Consider multiplying an $n \times n^p$ matrix with an $n^p \times n$ matrix. One can cut the $n \times n^p$ matrix into $n^{p-q}$ rectangular blocks of size $n \times n^q$ and the $n^p \times n$ matrix into $n^{p-q}$ rectangular blocks of size $n^q \times n$, and compute the multiplication of the corresponding blocks. This approach takes time $n^{p-q+\omega(q)+o(1)}$, from which the desired inequality immediately follows. ∎

Key to our analysis is the following lemma, which establishes the convexity of $\omega(k)$.

**Lemma 3.6** (Convexity)**.** *The fast rectangular matrix multiplication time exponent $\omega(k)$ as defined in Definition 3.2 is convex in $k$.*

*Proof.* Let $k = \alpha \cdot p + (1-\alpha) \cdot q$ for $\alpha \in (0,1)$. For notational simplicity, we assume that $n^p$, $n^q$ and $n^k$ are all integers. Consider a rectangular matrix of dimensions $n \times n^k$. Since $\alpha p \leq k$, we can tile this rectangular matrix with matrices of dimensions $n^\alpha \times n^{\alpha p}$. Then, the product of this tiled matrix with another similarly tiled matrix of dimensions $n^k \times n$ can be obtained by viewing it as a multiplication of a matrix of dimensions $n/n^\alpha \times n^k/n^{\alpha p}$ with one of dimensions $n^k/n^{\alpha p} \times n^{1/\alpha}$, where each "element" of these two matrices is itself a matrix of dimensions $n^\alpha \times n^{\alpha p}$. With this recursion in tow, we obtain the following upper bound.

$$\begin{aligned}
\mathcal{T}_{\mathrm{mat}}(n, n^k, n) \leq & \mathcal{T}_{\mathrm{mat}}(n^\alpha, n^{\alpha p}, n^\alpha) \cdot \mathcal{T}_{\mathrm{mat}}(n/n^\alpha, n^k/n^{\alpha p}, n/n^\alpha) \\
= & \mathcal{T}_{\mathrm{mat}}(n^\alpha, n^{\alpha p}, n^\alpha) \cdot \mathcal{T}_{\mathrm{mat}}(n^{(1-\alpha)}, n^{(1-\alpha)q}, n^{(1-\alpha)}) \\
\leq & n^{\alpha \cdot \omega(p) + o(1)} \cdot n^{(1-\alpha)\cdot\omega(q)+o(1)}.
\end{aligned}$$

The final step above follows from denoting $m = n^\alpha$ and observing that multiplying matrices of dimensions $n^\alpha \times n^{\alpha \cdot p}$ costs, by Definition 3.2, $m^{\omega(p)+o(1)}$, which is exactly $n^{\alpha(\omega(p)+o(1))}$. Applying Definition 3.2 and comparing exponents, this implies that

$$\omega(k) \leq \alpha \cdot \omega(p) + (1-\alpha) \cdot \omega(q),$$

which proves the convexity of the function $\omega(k)$. □

**Claim 3.7.** $\omega(1.68568) \leq 2.96370$.

*Proof.* We can upper bound $\omega(1.68568)$ in the following sense

$$\begin{aligned}
\omega(1.68568) &= \omega(0.25728 \cdot 1.5 + (1 - 0.25728) \cdot 1.75) \\
&\leq 0.25728 \cdot \omega(1.5) + (1 - 0.25728) \cdot \omega(1.75) \\
&\leq 0.25728 \cdot 2.79654 + (1 - 0.25728) \cdot 3.02159 \\
&\leq 2.96370,
\end{aligned}$$

where the first step follows from convexity of $\omega$ (Lemma 3.6), the third step follows from $\omega(1.5) \leq 2.79654$ and $\omega(1.75) \leq 3.02159$ (Lemma 3.4). □

**Lemma 3.8.** *Let $\mathcal{T}_{\mathrm{mat}}$ be defined as in Definition 3.1. Then for any positive integers $h$, $\ell$, and $k$, we have*

$$\mathcal{T}_{\mathrm{mat}}(h, \ell k, h) \leq O(\mathcal{T}_{\mathrm{mat}}(hk, \ell, hk)).$$

*Proof.* Given any matrices $A, B^\top \in \mathbb{R}^{h, \ell k}$, by Definition 3.1, the cost of computing the matrix product $AB$ is $\mathcal{T}_{\mathrm{mat}}(h, \ell k, h)$. We now show how to compute this product in time $O(\mathcal{T}_{\mathrm{mat}}(hk, \ell, hk))$. We cut $A$ and $B^\top$ into $k$ sub-matrices each of size $h \times \ell$, i.e. $A = (A_1, \cdots, A_k)$ and $B^\top = (B_1^\top, \cdots, B_k^\top)$, where each $A_i, B_i^\top \in \mathbb{R}^{h \times \ell}$ for all $i \in [k]$. By performing matrix multiplication blockwise, we can write

$$AB = \sum_{i=1}^{k} A_i B_i.$$

Next, we stack the $k$ matrices $A_1, \cdots, A_k$ vertically to form a matrix $A' \in \mathbb{R}^{hk,\ell}$. Similarly, we stack the $k$ matrices $B_1, \cdots, B_k$ horizontally to form a matrix $B' = (B_1, \cdots, B_k) \in \mathbb{R}^{\ell,hk}$. By Definition 3.1, we can compute $A'B' \in \mathbb{R}^{hk,hk}$ in time $\mathcal{T}_{\text{mat}}(hk, \ell, hk)$. To complete the proof, we note that we can derive $AB$ from $A'B'$ as follows: for each $j \in [k]$, the $j$th diagonal block of $A'B'$ of size $h \times h$ is exactly $A_jB_j$, and summing up the $k$ diagonal $h \times h$ blocks of $A'B'$ gives $AB$. □

### 3.3 General upper bound on $\mathcal{T}_{\text{mat}}(n, mn, n)$ and $\mathcal{T}_{\text{mat}}(m, n^2, m)$

**Lemma 3.9.** *Let $\mathcal{T}_{\text{mat}}$ be defined as in Definition 3.1.*
*If $m \geq n$, then we have*

$$\mathcal{T}_{\text{mat}}(n, mn, n) \leq O(\mathcal{T}_{\text{mat}}(m, n^2, m)).$$

*If $m \leq n$, then we have*

$$\mathcal{T}_{\text{mat}}(m, n^2, m) \leq O(\mathcal{T}_{\text{mat}}(n, mn, n)).$$

*Proof.* We only prove the case of $m \geq n$, as the other case where $m < n$ is similar. This is an immediate consequence of Lemma 3.8 by taking $h = n$, $\ell = n^2$, and $k = \lfloor m/n \rfloor$, where $k$ is a positive integer because $m \geq n$. □

In the next lemma, we derive upper bounds on the term $\mathcal{T}_{\text{mat}}(m, n^2, m)$ when $m \geq n$ and $\mathcal{T}_{\text{mat}}(n, mn, n)$ when $m < n$, which is crucial to our runtime analysis.

**Lemma 3.10.** *Let $\mathcal{T}_{\text{mat}}$ be defined as in Definition 3.1 and $\omega$ be defined as in Definition 3.2.*
*Property I. We have*

$$\mathcal{T}_{\text{mat}}(n, mn, n) \leq O(mn^{\omega+o(1)}).$$

*Property II. We have*

$$\mathcal{T}_{\text{mat}}(m, n^2, m) \leq O\left(\sqrt{n}\left(mn^2 + m^\omega\right)\right).$$

*Proof.* **Property I.**

Recall from Definition 3.1 that $\mathcal{T}_{\text{mat}}(n, mn, n)$ is the cost of multiplying a matrix of size $n \times mn$ with one of size $mn \times n$. We can cut each of the matrices into $m$ sub-matrices of size $n \times n$ each. The product in question then can be obtained by multiplying these sub-matrices. Since there are $m$ of them, and each product of an $n \times n$ submatrix with another $n \times n$ submatrix costs, by definition, $n^{\omega+o(1)}$, we get $\mathcal{T}_{\text{mat}}(n, mn, n) \leq O(mn^{\omega+o(1)})$, as claimed.

**Property II.**

Let $m = n^a$, where $a \in (0, \infty)$. By definition, $\mathcal{T}_{\text{mat}}(m, n^2, m)$ is the cost of multiplying a matrix of size $m \times n^2$ with one of size $n^2 \times m$. Expressing $n^2$ as $m^{2/a}$ then gives, by Definition 3.2, that

$$\mathcal{T}_{\text{mat}}(m, n^2, m) = m^{\omega(2/a)+o(1)} = n^{a \cdot \omega(2/a)+o(1)}$$

Property II is then an immediate consequence of the following inequality, which we prove next:

$$\omega(2/a) < \max(1 + 2.5/a, \omega(1) + 0.5/a) \qquad \forall a \in (0, \infty). \tag{14}$$

Define $b = 2/a \in (0, \infty)$. Then the desired inequality in (14) can be expressed in terms of $b$ as

$$\omega(b) < \max(1 + 5b/4, \omega(1) + b/4) \qquad \forall b \in (0, \infty). \tag{15}$$

Notice that the RHS of (15) is a maximum of two linear functions of $b$ and these intersect at $b^* = \omega(1) - 1$. By the convexity of $\omega(\,\cdot\,)$ as proved in Lemma 3.6, it suffices to verify (15) at the endpoints $b \to 0$, $b \to \infty$ and $b = b^*$. In the case where $b = \delta$ for any $\delta < 1$, (15) follows immediately from the observation that $\omega(\delta) < \omega(1)$. We next argue about the case $b \to \infty$. By Lemma 3.4 we have $\omega(2) \leq 3.252$. Using Lemma 3.5, we have $\omega(b) \leq b - 2 + \omega(2)$. Combining these two facts implies that for any $b > 2$, we have

$$\omega(b) \leq b - 2 + \omega(2) \leq 1 + 5b/4,$$

which again satisfies (15). The final case is $b = b^* = \omega(1) - 1$, for which (15) is equivalent to

$$\omega(\omega(1) - 1) < 5\omega(1)/4 - 1/4. \tag{16}$$

By Lemma 3.4, we have that $\omega(1) - 2 \in [0, 0.372927]$. Then to prove (16), it is sufficient to show that

$$\omega(t+1) < 5t/4 + 9/4 \qquad \forall t \in [0, 0.372927]. \tag{17}$$

By the convexity of $\omega(\,\cdot\,)$ as proved in Lemma 3.6, the upper bound of $\omega(2) \leq 3.251640$ in Lemma 3.4, and recalling that $\omega(1) = t + 2$ for $t \in [0, 0.372927]$, we have for $k \in [1, 2]$,

$$\omega(k) \leq \omega(1) + (k-1) \cdot (3.251640 - (t+2)) = t + 2 + (k-1) \cdot (1.251640 - t).$$

In particular, using this inequality for $k = t + 1$, we have

$$\begin{aligned} \omega(t+1) - 5t/4 - 9/4 &\leq (t+2) + t \cdot (1.251640 - t) - 5t/4 - 9/4 \\ &= -t^2 + 1.00164t - 1/4, \end{aligned}$$

which is negative on the entire interval $[0, 0.372927]$. This establishes (17) and finishes the proof. □

### 3.4 Specific upper bound on $\mathcal{T}_{\text{mat}}(m, n^2, m)$

**Lemma 3.11.** *For any two positive integers $n$ and $m$, we have*

$$\mathcal{T}_{\text{mat}}(m, n^2, m) = o\left(m^3 + mn^{2.37}\right).$$

*Proof.* Let $m = n^a$ where $a \in (0, \infty)$. Recall that $\mathcal{T}_{\text{mat}}(m, n^2, m) = m^{\omega(2/a)+o(1)} = n^{a\omega(2/a)+o(1)}$. We consider the following two cases according to the range of $a$.

**Case 1:** $a \in [1.18647, \infty)$. In this case, we have $\omega(2/a) \leq \omega(2/1.18647) \leq \omega(1.68568) < 3$, where the last inequality follows from Claim 3.7. This implies that

$$\mathcal{T}_{\text{mat}}(m, n^2, m) = o(n^{3a}) = o(m^3). \tag{18}$$

**Case 2:** $a \in (0, 1.18647]$. In this case, we have $2/a \in [1.68567, \infty)$. Consider the linear function

$$y(t) = 1 + 2.37 \cdot \frac{t}{2}. \tag{19}$$

By Claim 3.7, we have

$$\omega(1.68567) < 2.997 \leq y(1.68567). \tag{20}$$

By Lemma 3.4, we have

$$\omega(2) < 3.37 = y(2). \tag{21}$$

An application of Lemma 3.5 then gives, for any $t \geq 2$, the inequality

$$\omega(t) \leq t - 2 + \omega(2) < t - 2 + y(2) \leq y(t), \tag{22}$$

where the last inequality is by definition of $y(t)$ from (19). Therefore, combining the convexity of $\omega(\,\cdot\,)$, as proved in Lemma 3.6, with (20), (21), and (22), we conclude that for any $t \in [1.68567, \infty)$, the function $\omega$ is bounded from above by the affine function $y$, expressed as follows.

$$\omega(t) < y(t) = 1 + 2.37 \cdot \frac{t}{2}.$$

This implies that

$$\mathcal{T}_{\text{mat}}(m, n^2, m) = n^{a \cdot \omega(2/a) + o(1)} = o(n^{a+2.37}) = o(mn^{3.27}). \tag{23}$$

Combining the results from (18) and (23) finishes the proof of the lemma. □

# 4 Main Theorem

In this section, we give the formal statement of our main result.

**Theorem 4.1** (Main result, formal)**.** *Consider a semidefinite program with variable size $n \times n$ and $m$ constraints (assume there are no redundant constraints):*

$$\max \langle C, X \rangle \text{ subject to } X \succeq 0, \langle A_i, X \rangle = b_i \text{ for all } i \in [m]. \tag{24}$$

*Assume that any feasible solution $X \in \mathbb{S}^{n \times n}_{\geq 0}$ satisfies $\|X\|_{\text{op}} \leq R$. Then for any error parameter $0 < \delta \leq 0.01$, there is an interior point method that outputs in time $O^*(\sqrt{n}(mn^2 + m^\omega + n^\omega) \log(n/\delta))$ a positive semidefinite matrix $X \in \mathbb{R}^{n \times n}_{\geq 0}$ such that*

$$\langle C, X \rangle \geq \langle C, X^* \rangle - \delta \cdot \|C\|_{\text{op}} \cdot R \quad \text{and} \quad \sum_{i \in [m]} \left| \langle A_i, \widehat{X} \rangle - b_i \right| \leq 4n\delta \cdot (R \sum_{i \in [m]} \|A_i\|_1 + \|b\|_1),$$

*where $\omega$ is the exponent of matrix multiplication, $X^*$ is any optimal solution to the semidefinite program in (24), and $\|A_i\|_1$ is the Schatten 1-norm of matrix $A_i$.*

The proof of Theorem 4.1 is given in the subsequent sections.

# 5 Approximate Central Path via Approximate Hessian

## 5.1 Main result for approximate central path

Our main result of this section is the following.

**Theorem 5.1** (Approximate central path)**.** *Consider a semidefinite program as in Definition 1.1 with no redundant constraints. Assume that any feasible solution $X \in \mathbb{S}^{n \times n}_{\geq 0}$ satisfies $\|X\|_{\text{op}} \leq R$. Then for any error parameter $0 < \delta \leq 0.01$ and Newton step size $\epsilon_N$ satisfying $\sqrt{\delta} < \epsilon_N \leq 0.1$,*

*Algorithm 1 outputs, in* $T = \frac{40}{\epsilon_N}\sqrt{n}\log(n/\delta)$ *iterations, a positive semidefinite matrix* $X \in \mathbb{R}^{n\times n}_{\succeq 0}$ *that satisfies*

$$\langle C, X\rangle \geq \langle C, X^*\rangle - \delta \cdot \|C\|_{\text{op}} \cdot R \quad \text{and} \quad \sum_{i\in[m]} \left|\langle A_i, \widehat{X}\rangle - b_i\right| \leq 4n\delta \cdot (R \sum_{i\in[m]} \|A_i\|_1 + \|b\|_1), \tag{25}$$

*where* $X^*$ *is any optimal solution to the semidefinite program in Definition 1.1, and* $\|A_i\|_1$ *is the Schatten* 1*-norm of matrix* $A_i$*. Further, in each iteration of Algorithm 1, the following invariant holds for* $\alpha_H = 1.03$*:*

$$\|S^{-1/2} S_{\text{new}} S^{-1/2} - I\|_F \leq \alpha_H \cdot \epsilon_N. \tag{26}$$

*Proof.* At the start of Algorithm 1, Lemma 9.1 is called to modify the semidefinite program to obtain an initial dual solution $y$ for the modified SDP that is close to the dual central path at $\eta = 1/(n+2)$. This ensures that the invariant $g_\eta(y)^\top H(y)^{-1} g_\eta(y) \leq \epsilon_N^2$ holds at the start of the algorithm. Therefore, by Lemma 5.4 and Lemma 5.5, this invariant continues to hold throughout the run of the algorithm. Therefore, after $T = \frac{40}{\epsilon_N}\sqrt{n}\log\left(\frac{n}{\delta}\right)$ iterations, the step size $\eta$ in Algorithm 1 grows to $\eta = (1 + \frac{\epsilon_N}{20\sqrt{n}})^T/(n+2) \geq 2n/\delta^2$. It then follows from Lemma 5.6 that

$$b^\top y \leq b^\top y^* + \frac{n}{\eta} \cdot (1 + 2\epsilon_N) \leq b^\top y^* + \delta^2.$$

Thus when the algorithm stops, the dual solution $y$ has duality gap at most $\delta^2$ for the modified SDP. Lemma 9.1 then shows how to obtain an approximate solution to the original SDP that satisfies the guarantees in (25).

To prove (26), define $\Delta_S = S_{\text{new}} - S \in \mathbb{R}^{n\times n}$ and $\delta_y = y_{\text{new}} - y \in \mathbb{R}^m$. For each $i \in [n]$, we use $\delta_{y,i}$ to denote the $i$-th coordinate of vector $\delta_y$. We rewrite $\|S^{-1/2} S_{\text{new}} S^{-1/2} - I\|_F^2$ as

$$\begin{aligned}
\|S^{-1/2} S_{\text{new}} S^{-1/2} - I\|_F^2 &= \text{tr}\left[(S^{-1/2}(\Delta_S)S^{-1/2})^2\right] \\
&= \text{tr}\left[S^{-1}\left(\sum_{i=1}^m \delta_{y,i} A_i\right) S^{-1}\left(\sum_{j=1}^m \delta_{y,j} A_j\right)\right] \\
&= \sum_{i,j=1}^m \delta_{y,i}\delta_{y,j}\text{tr}[S^{-1}A_i S^{-1} A_j] \\
&= (\delta_y)^\top H(y) \delta_y \\
&= g_\eta(y)^\top \widetilde{H}(y)^{-1} H(y) \widetilde{H}(y)^{-1} g_\eta(y),
\end{aligned} \tag{27}$$

where we used the fact that $\Delta_S = \sum_{i=1}^m (\delta_y)_i A_i$. It then follows from Lemma 5.4 and the invariant $g_\eta(y)^\top H(y)^{-1} g_\eta(y) \leq \epsilon_N^2$ that

$$g_\eta(y)^\top \widetilde{H}(y)^{-1} H(y) \widetilde{H}(y)^{-1} g_\eta(y) \leq \alpha_H^2 \cdot \epsilon_N^2, \tag{28}$$

where $\alpha_H = 1.03$. Combining Equation (27) with Inequality (28) completes the proof of the theorem. ∎

Table 5.1: Summary of parameters in approxiate central path.

| Notation | Choice | Appearance | Meaning |
|---|---|---|---|
| $\alpha_H$ | 1.03 | Lemma 5.4 | Spectral approximation factor $\alpha_H^{-1} \cdot H \preceq \widetilde{H} \preceq \alpha_H \cdot H$ |
| $\epsilon_N$ | 0.1 | Lemma 5.5 | Upper bound on the Newton step size $(g_\eta^\top H^{-1} g_\eta)^{1/2}$ |
| $\epsilon_S$ | 0.01 | Algorithm 2 | Spectral approximation error $(1-\epsilon_S) \cdot S \preceq \widetilde{S} \preceq (1+\epsilon_S) \cdot S$ |

**Algorithm 1**

1: **procedure** MAIN($n, m, \delta, \epsilon_N, C, A, b$) ▷ $C \in \mathbb{S}^{n\times n}$, $\{A_i\}_{i=1}^m \in \mathbb{S}^{n\times n}$, vector $b \in \mathbb{R}^m$, error parameter $0 < \delta < 0.1$, Newton step size parameter $0 < \epsilon_N < 0.1$
2: Modify the SDP and obtain an initial dual solution $y$ according to Lemma 9.1
3: $\eta \leftarrow 1/(n+2)$
4: $T \leftarrow \frac{40}{\epsilon_N}\sqrt{n}\log\left(\frac{n}{\delta}\right)$
5: $\widetilde{S} \leftarrow S \leftarrow \sum_{i\in[m]} y_i A_i - C$.
6: **for** iter $= 1 \to T$ **do**
7: $\eta_{\text{new}} \leftarrow \eta\left(1 + \frac{\epsilon_N}{20\sqrt{n}}\right)$
8: **for** $j = 1, \cdots, m$ **do** ▷ Gradient computation
9: $g_{\eta_{\text{new}}}(y)_j \leftarrow \eta_{\text{new}} \cdot b_j - \text{tr}[S^{-1} \cdot A_j]$
10: **end for**
11: **for** $j = 1, \cdots, m$ **do** ▷ Hessian computation
12: **for** $k = 1, \cdots, m$ **do**
13: $\widetilde{H}_{j,k}(y) \leftarrow \text{tr}[\widetilde{S}^{-1} \cdot A_j \cdot \widetilde{S}^{-1} \cdot A_k]$
14: **end for**
15: **end for**
16: $\delta_y \leftarrow -\widetilde{H}(y)^{-1} g_{\eta_{\text{new}}}(y)$ ▷ Update on $y$
17: $y_{\text{new}} \leftarrow y + \delta_y$ ▷ Approximate Newton step
18: $S_{\text{new}} \leftarrow \sum_{i\in[m]} (y_{\text{new}})_i A_i - C$
19: $\widetilde{S}_{\text{new}} \leftarrow$ APPROXSLACKUPDATE$(S_{\text{new}}, \widetilde{S})$ ▷ Approximate slack computation
20: $y \leftarrow y_{\text{new}}$, $S \leftarrow S_{\text{new}}$, $\widetilde{S} \leftarrow \widetilde{S}_{\text{new}}$ ▷ Update variables
21: **end for**
22: Return an approximate solution to the original SDP according to Lemma 9.1
23: **end procedure**

## 5.2 Approximate slack update

**Lemma 5.2.** *Given positive definite matrices $S_{\text{new}}, \widetilde{S} \in \mathbb{S}_{>0}^{n\times n}$ and any parameter $0 < \epsilon_S < 0.01$, there is an algorithm (procedure* APPROXSLACKUPDATE *in Algorithm 2) that takes $O(n^{\omega+o(1)})$ time to output a positive definite matrix $\widetilde{S}_{\text{new}} \in \mathbb{S}_{>0}^{n\times n}$ such that*

$$\|S_{\text{new}}^{-1/2} \widetilde{S}_{\text{new}} S_{\text{new}}^{-1/2} - I\|_{\text{op}} \leq \epsilon_S. \tag{29}$$

*Proof.* The runtime of $O(n^{\omega+o(1)})$ is by the spectral decomposition $Z = U \cdot \Lambda \cdot U^\top$, the costliest step in the algorithm. To prove (29), we notice that $\lambda_{\text{new}}$ are the eigenvalues of $S_{\text{new}}^{-1/2} \widetilde{S}_{\text{new}} S_{\text{new}}^{-1/2} - I$ and by the algorithm description (lines 6 - 13), the upper bound $(\lambda_{\text{new}})_i \leq \epsilon_S$ holds for each $i \in [n]$. □

**Algorithm 2** Approximate Slack Update

**procedure** ApproxSlackUpdate($S_{\text{new}}, \widetilde{S}$) ▷ $S_{\text{new}}, \widetilde{S} \in \mathbb{S}^{n\times n}_{\succ 0}$ are positive definite matrices
  $\epsilon_S \leftarrow 0.01$ ▷ Spectral approximation constant
  $Z_{\text{mid}} \leftarrow S_{\text{new}}^{-1/2} \cdot \widetilde{S} \cdot S_{\text{new}}^{-1/2} - I$
  Compute spectral decomposition $Z_{\text{mid}} = U \cdot \Lambda \cdot U^\top$
    ▷ $\Lambda = \text{diag}(\lambda_1, \cdots, \lambda_n)$ are the eigenvalues of $Z_{\text{mid}}$, and $U \in \mathbb{R}^{n\times n}$ is orthogonal
  Let $\pi : [n] \to [n]$ be a sorting permutation such that $|\lambda_{\pi(i)}| \geq |\lambda_{\pi(i+1)}|$
  **if** $|\lambda_{\pi(1)}| \leq \epsilon_S$ **then**
    $\widetilde{S}_{\text{new}} \leftarrow \widetilde{S}$
  **else**
    $r \leftarrow 1$
    **while** $|\lambda_{\pi(2r)}| > \epsilon_S$ or $|\lambda_{\pi(2r)}| > (1 - 1/\log n)|\lambda_{\pi(r)}|$ **do**
      $r \leftarrow r + 1$
    **end while**
    $(\lambda_{\text{new}})_{\pi(i)} \leftarrow \begin{cases} 0 & \text{if } i = 1, 2, \cdots, 2r; \\ \lambda_{\pi(i)} & \text{otherwise.} \end{cases}$
    $\widetilde{S}_{\text{new}} \leftarrow \widetilde{S} + S_{\text{new}}^{1/2} \cdot U \cdot \text{diag}(\lambda_{\text{new}} - \lambda) \cdot U^\top \cdot S_{\text{new}}^{1/2}$
  **end if**
  **return** $\widetilde{S}_{\text{new}}$
**end procedure**

### 5.3 Closeness of slack implies closeness of Hessian

**Lemma 5.3.** *Given symmetric matrices $A_1, \cdots, A_m \in \mathbb{S}^{n\times n}$, and positive definite matrices $\widetilde{S}, S \in \mathbb{S}^{n\times n}_{\succ 0}$, define matrices $\widetilde{H} \in \mathbb{R}^{m\times m}$ and $H \in \mathbb{R}^{m\times m}$ as*

$$\widetilde{H}_{j,k} = \text{tr}[\widetilde{S}^{-1} A_j \widetilde{S}^{-1} A_k] \qquad \text{and} \qquad H_{j,k} = \text{tr}[S^{-1} A_j S^{-1} A_k].$$

*Then both $\widetilde{H}$ and $H$ are positive semidefinite. For any accuracy parameter $\alpha_S \geq 1$, if*

$$\alpha_S^{-1} \cdot S \preceq \widetilde{S} \preceq \alpha_S \cdot S,$$

*then we have that*

$$\alpha_S^{-2} \cdot H \preceq \widetilde{H} \preceq \alpha_S^2 \cdot H.$$

*Proof.* For any vector $v \in \mathbb{R}^n$, we define $A(v) = \sum_{i=1}^m v_i A_i$. We can rewrite $v^\top H v$ as follows.

$$v^\top H v = \sum_{i=1}^m \sum_{j=1}^m v_i v_j H_{i,j} = \sum_{i=1}^m \sum_{j=1}^m v_i v_j \text{tr}[S^{-1} A_i S^{-1} A_j] = \text{tr}[S^{-1/2} A(v) S^{-1} A(v) S^{-1/2}]. \tag{30}$$

Similarly, we have

$$v^\top \widetilde{H} v = \text{tr}[\widetilde{S}^{-1/2} A(v) \widetilde{S}^{-1} A(v) \widetilde{S}^{-1/2}]. \tag{31}$$

As the RHS of (30) and (31) are non-negative, both $\widetilde{H}$ and $H$ are positive semidefinite. Since $\widetilde{S} \preceq \alpha_S \cdot S$, we have $S^{-1} \preceq \alpha_S \cdot \widetilde{S}^{-1}$ (see Section 2.2), which gives the following inequalities

$$\begin{aligned} \text{tr}[S^{-1/2} A(v) S^{-1} A(v) S^{-1/2}] &\leq \alpha_S \cdot \text{tr}[S^{-1/2} A(v) \widetilde{S}^{-1} A(v) S^{-1/2}] \\ &\leq \alpha_S^2 \cdot \text{tr}[\widetilde{S}^{-1/2} A(v) \widetilde{S}^{-1} A(v) \widetilde{S}^{-1/2}], \end{aligned} \tag{32}$$

where the first inequality follows from viewing $\mathrm{tr}[S^{-1/2}A(v)S^{-1}A(v)S^{-1/2}]$ as $\sum_{i=1}^{n} u_i^\top S^{-1}u_i$ for $u_i = A(v)S^{-1/2}e_i$ and the second inequality follows similarly, after using the cyclic permutation property of trace. Similarly, using $\alpha_S^{-1} \cdot S \preceq \widetilde{S}$, we have

$$\mathrm{tr}[S^{-1/2}A(v)S^{-1}A(v)S^{-1/2}] \geq \alpha_S^{-2} \cdot \mathrm{tr}[\widetilde{S}^{-1/2}A(v)\widetilde{S}^{-1}A(v)\widetilde{S}^{-1/2}]. \tag{33}$$

Combining (32) and (33) with (30) and (31) along with the fact that $v$ can be any arbitrary $n$-dimensional vector finishes the proof of the lemma. □

### 5.4 Approximate Hessian maintenance

**Lemma 5.4.** *In each iteration of Algorithm 1, for $\alpha_H = 1.03$, the approximate Hessian $\widetilde{H}(y)$ satisfies that*

$$\alpha_H^{-1}H(y) \preceq \widetilde{H}(y) \preceq \alpha_H \cdot H(y).$$

*Proof.* By Lemma 5.2, given as input two positive definite matrices $S_{\text{new}}$ and $\widetilde{S}$, Algorithm 2 outputs a matrix $\widetilde{S}_{\text{new}}$ such that

$$\|S_{\text{new}}^{-1/2}\widetilde{S}_{\text{new}}S_{\text{new}}^{-1/2} - I\|_{\text{op}} \leq \epsilon_S,$$

where $\epsilon_S = 0.01$ as in Algorithm 2. By definition of operator norm, this implies that in each iteration of Algorithm 1, we have, for $\alpha_S = 1.011$,

$$\alpha_S^{-1} \cdot S \preceq \widetilde{S} \preceq \alpha_S \cdot S.$$

The statement of this lemma then follows from Lemma 5.3. □

### 5.5 Invariance of Newton step size

The following lemma is standard in the theory of interior point methods (e.g. see [Ren01]).

**Lemma 5.5** (Invariance of Newton step [Ren01])**.** *Given any parameters $1 \leq \alpha_H \leq 1.03$ and $0 < \epsilon_N \leq 1/10$, suppose that $g_\eta(y)^\top H(y)^{-1}g_\eta(y) \leq \epsilon_N^2$ holds for some feasible dual solution $y \in \mathbb{R}^m$ and parameter $\eta > 0$, and positive definite matrix $\widetilde{H} \in \mathbb{S}_{>0}^{n\times n}$ satisfies*

$$\alpha_H^{-1}H(y) \preceq \widetilde{H} \preceq \alpha_H H(y)$$

*Then $\eta_{\text{new}} = \eta(1 + \frac{\epsilon_N}{20\sqrt{n}})$ and $y_{\text{new}} = y - \widetilde{H}^{-1}g_{\eta_{\text{new}}}(y)$ satisfy*

$$g_{\eta_{\text{new}}}(y_{\text{new}})^\top H(y_{\text{new}})^{-1}g_{\eta_{\text{new}}}(y_{\text{new}}) \leq \epsilon_N^2.$$

### 5.6 Approximate optimality

The following lemma is also standard in interior point method.

**Lemma 5.6** (Approximate optimality [Ren01])**.** *Suppose $0 < \epsilon_N \leq 1/10$, dual feasible solution $y \in \mathbb{R}^m$, and parameter $\eta \geq 1$ satisfy the following bound on Newton step size:*

$$g_\eta(y)^\top H(y)^{-1}g_\eta(y) \leq \epsilon_N^2.$$

*Let $y^*$ be an optimal solution to the dual formulation (2). Then we have*

$$b^\top y \leq b^\top y^* + \frac{n}{\eta} \cdot (1 + 2\epsilon_N).$$

## 6 Low-rank Update

Crucial to being able to efficiently approximate the Hessian in each iteration is the condition that the rank of the update be not too large. We formalize this idea in the following theorem, essential to the runtime analysis in Section 7.

**Theorem 6.1** (Rank inequality)**.** *Let $r_0 = n$ and $r_i$ be the rank of the update to the approximate slack matrix $\widetilde{S}$ when calling Algorithm 2 in iteration $i$ of Algorithm 1. Then, over $T$ iterations of Algorithm 1, the ranks $r_i$ satisfy the inequality*

$$\sum_{i=0}^{T} \sqrt{r_i} \leq O(T \log^{1.5} n).$$

The rest of this section is devoted to proving Theorem 6.1. To this end, we define the "error" matrix $Z \in \mathbb{R}^{n \times n}$ as follows

$$Z = S^{-1/2} \widetilde{S} S^{-1/2} - I \tag{34}$$

and the potential function $\Phi : \mathbb{R}^{n \times n} \to \mathbb{R}$

$$\Phi(Z) = \sum_{i=1}^{n} \frac{|\lambda(Z)|_{[i]}}{\sqrt{i}}, \tag{35}$$

where $|\lambda(Z)|_{[i]}$ denotes the $i$'th entry in the list of absolute eigenvalues of $Z$ sorted in descending order. The following lemma bounds, from above, the change in the potential described by Equation (35), when $S$ is updated to $S_{\text{new}}$.

**Lemma 6.2** (Potential change when $S$ changes)**.** *Suppose matrices $S$, $S_{\text{new}}$ and $\widetilde{S}$ satisfy the inequalities*

$$\|S^{-1/2} S_{\text{new}} S^{-1/2} - I\|_F \leq 0.02 \qquad \text{and} \qquad \|S^{-1/2} \widetilde{S} S^{-1/2} - I\|_{\text{op}} \leq 0.01. \tag{36}$$

*Define matrices $Z = S^{-1/2} \widetilde{S} S^{-1/2} - I$ and $Z_{\text{mid}} = (S_{\text{new}})^{-1/2} \widetilde{S} (S_{\text{new}})^{-1/2} - I$. Then we have*

$$\Phi(Z_{\text{mid}}) - \Phi(Z) \leq \sqrt{\log n}.$$

*Proof.* Our goal is to prove

$$\sum_{i=1}^{n} (\lambda(Z)_{[i]} - \lambda(Z_{\text{mid}})_{[i]})^2 \leq 10^{-3}. \tag{37}$$

We first show that the lemma statement is implied by (37). We rearrange the order of the eigenvalues of $Z_{\text{mid}}$ and $Z$ so that $\lambda(Z_{\text{mid}})_i$ and $\lambda(Z)_i$ are the $i$th largest eigenvalues of $Z_{\text{mid}}$ and $Z$, respectively. For each $i \in [n]$, denote $\Delta_i = \lambda(Z_{\text{mid}})_i - \lambda(Z)_i$. Then (37) is equivalent to $\|\Delta\|_2^2 \leq 10^{-3}$. Let $\tau$ be the descending order of the magnitudes of eigenvalues of $Z_{\text{mid}}$, i.e. $|\lambda(Z_{\text{mid}})_{\tau(1)}| \geq \cdots \geq |\lambda(Z_{\text{mid}})_{\tau(n)}|$. The potential change $\Phi(Z_{\text{mid}}) - \Phi(Z)$ can be upper bounded as

$$\begin{aligned}
\Phi(Z_{\text{mid}}) &= \sum_{i=1}^{n} \frac{1}{\sqrt{i}} |\lambda(Z_{\text{mid}})_{\tau(i)}| \\
&\leq \sum_{i=1}^{n} \left( \frac{1}{\sqrt{i}} |\lambda(Z)_{\tau(i)}| + \frac{1}{\sqrt{i}} |\Delta_{\tau(i)}| \right) \\
&\leq \Phi(Z) + \left( \sum_{i=1}^{n} \frac{1}{i} \right)^{1/2} \left( \sum_{i=1}^{n} |\Delta_i|^2 \right)^{1/2} \\
&\leq \Phi(Z) + \sqrt{\log n},
\end{aligned}$$

where the third line follows from

$$\sum_i \frac{1}{\sqrt{i}}|\lambda(Z)_{\tau(i)}| \le \sum_i \frac{1}{\sqrt{i}}|\lambda(Z)|_{[i]}$$

and Cauchy-Schwarz inequality. This proves the lemma.

The remaining part of this proof is therefore devoted to proving (37). Define $W = S_{\text{new}}^{-1/2} S^{1/2}$. Then, we can express $Z_{\text{mid}}$ in terms of $Z$ and $W$ in the following way.

$$\begin{aligned} Z_{\text{mid}} &= (S_{\text{new}})^{-1/2} \widetilde{S} (S_{\text{new}})^{-1/2} - I \\ &= (S_{\text{new}})^{-1/2} S^{1/2} S^{-1/2} \widetilde{S} S^{-1/2} S^{1/2} (S_{\text{new}})^{-1/2} - I \\ &= W Z W^\top + W W^\top - I. \end{aligned} \tag{38}$$

Let $\lambda(M)_{[i]}$ denote the $i$'th (ordered) eigenvalue of a matrix $M$. We then have

$$\begin{aligned} \sum_{i=1}^n (\lambda(Z_{\text{mid}})_{[i]} - \lambda(WZW^\top)_{[i]})^2 &\le \|Z_{\text{mid}} - WZW^\top\|_F^2 \\ &= \|W^\top W - I\|_F^2, \end{aligned} \tag{39}$$

where the first inequality is by Fact 2.3 (which is applicable here because $Z_{\text{mid}}$ and $WZW^\top$ are both normal matrices) and the second step is by (38). Denote the eigenvalues of $S^{-1/2} S_{\text{new}} S^{-1/2}$ by $\{\nu_i\}_{i=1}^n$. Then the first assumption in (36) implies that $\sum_{i\in[n]} (\nu_i - 1)^2 \le 4 \times 10^{-4}$. It follows that

$$\|W^\top W - I\|_F^2 = \|S^{1/2} S_{\text{new}}^{-1} S^{1/2} - I\|_F^2 = \sum_{i\in[n]} (1/\nu_i - 1)^2 \le 5 \times 10^{-4}, \tag{40}$$

where the last inequality is because the first assumption from (36) implies $\nu_i \ge 0.98$ for all $i \in [n]$. Plugging (40) into the right hand side of (39), we have

$$\sum_{i=1}^n (\lambda(Z_{\text{mid}})_{[i]} - \lambda(WZW^\top)_{[i]})^2 \le 5 \times 10^{-4}. \tag{41}$$

Let $W = U\Sigma V^\top$ be the singular value decomposition of $W$, with $U$ and $V$ being $n \times n$ unitary matrices. Because of the invariance of the Frobenius norm under unitary transformation, (40) is then equivalent to

$$\|\Sigma^2 - I\|_F = \sum_{i=1}^n (\sigma_i^2 - 1)^2 \le 5 \times 10^{-4}. \tag{42}$$

Since $U$ and $V$ are unitary, the matrix $WZW^\top = U\Sigma V^\top Z V \Sigma U^\top$ is similar to $\Sigma V^\top Z V \Sigma$, and the matrix $Z' = V^\top Z V$ is similar to $Z$. Therefore,

$$\begin{aligned} \sum_{i=1}^n (\lambda(WZW^\top)_{[i]} - \lambda(Z)_{[i]})^2 &= \sum_{i=1}^n (\lambda(\Sigma Z' \Sigma)_{[i]} - \lambda(Z')_{[i]})^2 \\ &\le \|\Sigma Z' \Sigma - Z'\|_F^2, \end{aligned} \tag{43}$$

where the last inequality is by Fact 2.3. We rewrite the Frobenius norm as

$$\begin{aligned}\|\Sigma Z'\Sigma - Z'\|_F &= \|(\Sigma - I)Z'(\Sigma - I) + (\Sigma - I)Z' + Z'(\Sigma - I)\|_F \\ &\le \|(\Sigma - I)Z'(\Sigma - I)\|_F + 2\|(\Sigma - I)Z'\|_F. \end{aligned} \tag{44}$$

The first term can be bounded as:

$$\begin{aligned}\|(\Sigma - I)Z'(\Sigma - I)\|_F^2 &= \mathrm{tr}[(\Sigma - I)Z'(\Sigma - I)^2 Z'(\Sigma - I)] \\ &\le \mathrm{tr}[(\Sigma - I)^4 \cdot (Z')^2] \\ &\le 0.01^2 \cdot \mathrm{tr}[(\Sigma - I)^4] \\ &= \sum_{i=1}^{n} (\sigma_i - 1)^4 \\ &\le 5 \times 10^{-8}, \end{aligned} \tag{45}$$

The first inequality above uses Fact 2.1, the second used the observation that $\|Z'\|_{\mathrm{op}} = \|Z\|_{\mathrm{op}} \le 0.01$, and the last inequality follows from (42) and the fact that $\sum_{i=1}^{n}(\sigma_i - 1)^4 \le \sum_{i=1}^{n}(\sigma_i^2 - 1)^2$. Similarly, we can bound the second term as

$$\begin{aligned}\|(\Sigma - I)Z'\|_F^2 &= \mathrm{tr}[(\Sigma - I)(Z')^2(\Sigma - I)] \\ &\le \mathrm{tr}[(\Sigma - I)^2 (Z')^2] \\ &\le 0.01^2 \cdot \mathrm{tr}[(\Sigma - I)^2] \;\le\; 10^{-7}. \end{aligned} \tag{46}$$

It follows from (43), (44) and (46) that

$$\sum_{i=1}^{n} (\lambda(WZW^\top)_{[i]} - \lambda(Z)_{[i]})^2 \le 10^{-6}. \tag{47}$$

Combining (41) and (47), we get that $\sum_{i=1}^{n}(\lambda(Z)_{[i]} - \lambda(Z_{\mathrm{mid}})_{[i]})^2 \le 10^{-3}$ which establishes (37). This completes the proof of the lemma. □

**Lemma 6.3** (Potential change when $\widetilde{S}$ changes)**.** *Given positive definite matrices $S_{\mathrm{new}}, \widetilde{S} \in \mathbb{S}^n_{>0}$, let $\widetilde{S}_{\mathrm{new}}$ and $r$ be generated during the run of Algorithm 2 when the inputs are $S_{\mathrm{new}}$ and $\widetilde{S}$. Define the matrices $Z_{\mathrm{mid}} = (S_{\mathrm{new}})^{-1/2}\widetilde{S}(S_{\mathrm{new}})^{-1/2} - I$ and $Z_{\mathrm{new}} = (S_{\mathrm{new}})^{-1/2}\widetilde{S}_{\mathrm{new}}(S_{\mathrm{new}})^{-1/2} - I$. Then we have*

$$\Phi(Z_{\mathrm{mid}}) - \Phi(Z_{\mathrm{new}}) \ge \frac{10^{-4}}{\log n}\sqrt{r}.$$

*Proof.* The setup of the lemma considers the eigenvalues of $Z$ when $\widetilde{S}$ changes. For the sake of notational convenience, we define $y = |\lambda(Z_{\mathrm{mid}})|$, the vector of absolute values of eigenvalues of $Z_{\mathrm{mid}} = S_{\mathrm{new}}^{-1/2}\widetilde{S}S_{\mathrm{new}}^{-1/2} - I$. Recall from Table 5.1 that $\epsilon_S = 0.01$. We consider two cases below.

**Case 1.** There does not exist an $i \le n/2$ that satisfies the two conditions $y_{[2i]} < \epsilon_S$ and $y_{[2i]} < (1 - 1/10\log n)y_{[i]}$. In this case, we have $r = n/2$. We consider two sub-cases.

- Case (a). For all $i \in [n]$, we have $y_{[i]} \ge \epsilon_S$. In this case, we change all $n$ coordinates of $y$, and the change in each coordinate contributes to a potential decrease of at least $\epsilon_S/\sqrt{n}$. Therefore, we have $\Phi(Z_{\mathrm{mid}}) - \Phi(Z_{\mathrm{new}}) \ge \epsilon_S\sqrt{n} \ge \frac{10^{-4}}{\log n}\sqrt{r}$.

- Case (b). There exists a minimum index $i \le n/2$ such that $y_{[2j]} < \epsilon_S$ holds for all $j$ in the range $i \le j \le n/2$. In this case, for all $j$ in the above range, we have that $y_{[2j]} \ge (1 - 1/10 \log n) y_{[j]}$. In particular, picking $j = i, 2i, \cdots$ gives

$$y_{[n]} \ge y_{[i]} \cdot (1 - 1/(10 \log n))^{\lceil \log n \rceil} \ge \epsilon_S / 10.$$

  Recalling that our notation $y_{[i]}$ denotes the $i$'th absolute eigenvalue in decreasing order, we use the above inequality and repeat the argument from the previous sub-case to conclude that $\Phi(Z_{\mathrm{mid}}) - \Phi(Z_{\mathrm{new}}) \ge \epsilon_S/10 \cdot \sqrt{n} \ge \frac{10^{-4}}{\log n} \cdot \sqrt{r}$.

**Case 2.** There exists an index $i$ for which both the conditions $y_{[2i]} < \epsilon_S$ and $y_{[2i]} < (1 - 1/10 \log n) y_{[i]}$ are satisfied. By definition, $r \le n/2$ is the smallest such index. Consider the index $j$ such that for all $j' < j$, we have $y_{[j']} \ge \epsilon_S$ and for all $j' \ge j$, we have $y_{[j]} < \epsilon_S$. By the same argument as in Case 1(b), we can prove $y_{[r]} \ge \epsilon_S/10$. Moreover, $y_{[2r]} < (1 - 1/10 \log n) y_{[r]}$ by definition of $r$. Denote by $y^{\mathrm{new}}$ the vector of magnitudes of the eigenvalues of $Z_{\mathrm{new}}$. Since $y^{\mathrm{new}}_{[i]}$ is set to 0 for each $i \in [2r]$, we have $y^{\mathrm{new}}_{[i]} = y_{[i+2r]} \le y_{[i]}$. Further, $y_{[2r]} < (1 - 1/10 \log n) y_{[r]}$ implies that for each $i \in [r]$, we have

$$y_{[i]} - y^{\mathrm{new}}_{[i]} \ge \frac{1}{10 \log n} \cdot y_{[r]} \ge \frac{10^{-2} \epsilon_S}{\log n} = \frac{10^{-4}}{\log n},$$

where $\epsilon_S = 0.01$ by Table 5.1. Therefore, we can bound, from below, the decrease in potential function as

$$\Phi(Z_{\mathrm{mid}}) - \Phi(Z_{\mathrm{new}}) \ge \sum_{i=1}^{r} \frac{y_{[i]} - y^{\mathrm{new}}_{[i]}}{\sqrt{i}} \ge \frac{10^{-4}}{\log n} \sqrt{r}.$$

This finishes the proof of the lemma. □

*Proof of Theorem 6.1.* Recall the definition of the potential function in (35) for an error matrix $Z \in \mathbb{S}^{n \times n}$:

$$\Phi(Z) = \sum_{i=1}^{n} \frac{|\lambda(Z)|_{[i]}}{\sqrt{i}}.$$

Let $S^{(i)}$ and $\widetilde{S}^{(i)}$ be the true and approximate slack matrices in the $i$th iteration of Algorithm 1. Define $Z^{(i)} = (S^{(i)})^{-1/2} \widetilde{S}^{(i)} (S^{(i)})^{-1/2} - I$ and $Z^{(i)}_{\mathrm{mid}} = (S^{(i+1)})^{-1/2} \widetilde{S}^{(i)} (S^{(i+1)})^{-1/2} - I$. By Lemma 6.2, we have that

$$\Phi(Z^{(i)}_{\mathrm{mid}}) - \Phi(Z^{(i)}) \le \sqrt{\log n}.$$

From Lemma 6.3, we have the following potential decrease:

$$\Phi(Z^{(i)}_{\mathrm{mid}}) - \Phi(Z^{(i+1)}) \ge \frac{10^{-4}}{\log n} \sqrt{r_i}.$$

These together imply that

$$\Phi(Z^{(i+1)}) - \Phi(Z^{(i)}) \le \sqrt{\log n} - \frac{10^{-4}}{\log n} \sqrt{r_i}. \tag{48}$$

We note that $\Phi(Z^{(0)}) = 0$ as we initialized $\widetilde{S} = S$ in the beginning of the algorithm, and that the potential function $\Phi(Z)$ is always non-negative. The theorem then follows by summing up (48) over all $T$ iterations. □

# 7 Runtime Analysis

Our main result of this section is the following bound on the runtime of Algorithm 1.

**Theorem 7.1** (Runtime bound)**.** *The total runtime of Algorithm 1 for solving an SDP with variable size $n \times n$ and $m$ constraints is at most $O^*\left(\sqrt{n}\left(mn^2 + \max(m,n)^\omega\right)\right)$, where $\omega$ is the matrix multiplication exponent as defined in Definition 3.2.*

To prove Theorem 7.1, we first upper bound the runtime in terms of fast rectangular matrix multiplication times. The iteration complexity of Algorithm 1 is $T = \widetilde{O}(\sqrt{n})$.

**Lemma 7.2** (Total cost)**.** *The total runtime of Algorithm 1 over $T$ iterations is upper bounded as*

$$\mathcal{T}_{\text{Total}} \leq O^*\left(\min\left(n \cdot \text{nnz}(A), mn^{2.5}\right) + \sqrt{n}\max(m,n)^\omega + \sum_{i=0}^{T}\left(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \mathcal{T}_{\text{mat}}(m, nr_i, m)\right)\right), \tag{49}$$

*where $\text{nnz}(A)$ is the total number of non-zero entries in all the constraint matrices, $r_i$, as defined in Theorem 6.1, is the rank of the update to the approximation slack matrix $\widetilde{S}$ in iteration $i$, and $\omega$ and $\mathcal{T}_{\text{mat}}$ are defined in Definitions 3.2 and 3.1, respectively.*

**Remark 7.3.** *A more careful analysis can improve the first term in the RHS of (49) to $\sqrt{n} \cdot \text{nnz}(A)^{1-\gamma} \cdot (mn^2)^\gamma$ for $\gamma = \frac{1}{2(3-\omega(1))}$. For the purpose of this paper, however, we will only need the simpler bound given in Lemma 7.2.*

*Proof.* The total runtime of Algorithm 1 consists of two parts:

- **Part 1.** The time to compute the approximate Hessian $\widetilde{H}(y)$ (which we abbreviate as $\widetilde{H}$) in Line 11 - 15.
- **Part 2.** The total cost of operations other than computing the approximate Hessian.

**Part 1.**

We analyze the cost of computing the approximate Hessian $\widetilde{H}$.

**Part 1a. Initialization.**

We start with computing $\widetilde{H}$ in the first iteration of the algorithm. Each entry of $\widetilde{H}$ involves the computation

$$\widetilde{H}_{j,k} = \text{tr}\left[(\widetilde{S}^{-1/2}A_j\widetilde{S}^{-1/2})(\widetilde{S}^{-1/2}A_k\widetilde{S}^{-1/2})\right].$$

It first costs $O^*(n^\omega)$ to invert $\widetilde{S}$. Then the cost of computing the key module of the approximate Hessian, $\widetilde{S}^{-1/2}A_j\widetilde{S}^{-1/2}$ for all $j \in [m]$, is obtained by stacking the matrices $A_j$ together:

$$\mathcal{T}_{\widetilde{S}^{-1/2}A_j\widetilde{S}^{-1/2} \text{ for all } j\in[m]} \leq O(\mathcal{T}_{\text{mat}}(n, mn, n)). \tag{50}$$

Vectorizing the matrices $\widetilde{S}^{-1/2}A_j\widetilde{S}^{-1/2}$ into row vectors of length $n^2$, for each $j \in [m]$, and stacking these rows vertically to form a matrix $B$ of dimensions $m \times n^2$, one observes that $\widetilde{H} = BB^\top$. We therefore have,

$$\mathcal{T}_{\text{computing } \widetilde{H} \text{ from } B} \leq O(\mathcal{T}_{\text{mat}}(m, n^2, m)). \tag{51}$$

Combining (50), (51), and the initial cost of inverting $\widetilde{S}$ gives the following cost for computing $\widetilde{H}$ for the first iteration:

$$\mathcal{T}_{\text{part 1a}} \leq O^*(\mathcal{T}_{\text{mat}}(m, n^2, m) + \mathcal{T}_{\text{mat}}(n, mn, n) + n^\omega). \tag{52}$$

**Part 1b. Accumulating low-rank changes over all the iterations**

Once the approximate Hessian in the first iteration has been computed, every next iteration has the approximate Hessian computed using a rank $r_i$ update to the approximate slack matrix $\widetilde{S}$ (see Line 15 of Algorithm 2). If the update from $\widetilde{S}$ to $\widetilde{S}_{\text{new}}$ has rank $r_i$, Fact 2.4 implies that we can compute, in time $O(n^{\omega+o(1)})$, the $n \times r_i$ matrices $V_+$ and $V_-$ satisfying $\widetilde{S}_{\text{new}}^{-1} = \widetilde{S}^{-1} + V_+V_+^\top - V_-V_-^\top$. The cost of updating $\widetilde{H}$ is then dominated by the computation of $\text{tr}[\widetilde{S}^{-1/2}A_jVV^\top A_k\widetilde{S}^{-1/2}]$, where $V \in \mathbb{R}^{n\times r_i}$ is either $V_+$ or $V_-$. We note that

$$\mathcal{T}_{A_jV \text{ for all } j\in[m]} \leq O^*\left(\min\left(r_i \cdot \text{nnz}(A), mn^2 r_i^{\omega-2+o(1)}\right)\right), \tag{53}$$

where $\text{nnz}(A)$ is the total number of non-zero entries in all the constraint matrices, and the second term in the minimum is obtained by stacking the matrices $A_j$ together and splitting it and $V$ into matrices of dimensions $r_i \times r_i$. Further, pre-multiplying $\widetilde{S}^{-1/2}$ with $A_jV$ for all $j \in [m]$ essentially involves computing the matrix product of an $n \times n$ matrix and an $n \times mr_i$ matrix, which, by Definition 3.1, costs $\mathcal{T}_{\text{mat}}(n, mr_i, n)$. This, together with (53), gives

$$\mathcal{T}_{\widetilde{S}^{-1/2}A_jV \text{ for all } j\in[m]} \leq O^*\left(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \min\left(r_i \cdot \text{nnz}(A), mn^2 r_i^{\omega-2+o(1)}\right)\right). \tag{54}$$

The final step is to vectorize all the matrices $\widetilde{S}^{-1/2}A_jV$, for each $j \in [m]$, and stack these vertically to get an $m \times nr_i$ matrix $B$, which gives the update to Hessian to be computed as $BB^\top$. This costs, by definition, $\mathcal{T}_{\text{mat}}(m, nr_i, m)$. Combining this with (54) gives the following run time for one update to the approximate Hessian:

$$\mathcal{T}_{\text{rank } r_i \text{ Hessian update}} \leq O^*\left(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \min\left(r_i \cdot \text{nnz}(A), mn^2 r_i^{\omega-2}\right) + \mathcal{T}_{\text{mat}}(m, nr_i, m) + n^\omega\right). \tag{55}$$

Using this bound over all $T = \widetilde{O}(\sqrt{n})$ iterations, and applying $\sum_{i=0}^{T}\sqrt{r_i} \leq \widetilde{O}(\sqrt{n})$ from Theorem 6.1, gives

$$\mathcal{T}_{\text{part 1b}} \leq O^*\left(\min(n \cdot \text{nnz}(A), mn^{2.5}) + \sqrt{n} \cdot n^\omega + \sum_{i=1}^{T}(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \mathcal{T}_{\text{mat}}(m, nr_i, m))\right). \tag{56}$$

**Combining Part 1a and 1b.**

Combining (52) and (56), we have

$$\begin{aligned}\mathcal{T}_{\text{part 1}} &\leq \mathcal{T}_{\text{part 1a}} + \mathcal{T}_{\text{part 1b}} \\ &\leq O^*\left(\min(n \cdot \text{nnz}(A), mn^{2.5}) + \sqrt{n} \cdot n^\omega + \sum_{i=0}^{T}(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \mathcal{T}_{\text{mat}}(m, nr_i, m))\right),\end{aligned} \tag{57}$$

where we incorporated the bound from (52) into the $i = 0$ case.

**Part 2.**

Observe that there are four operations performed in Algorithm 1 other than computing $\widetilde{H}$:

- **Part 2a.** computing the gradient $g_\eta(y)$
- **Part 2b.** inverting the approximate Hessian $\widetilde{H}$
- **Part 2c.** updating the dual variables $y_{\mathrm{new}}$ and $S(y_{\mathrm{new}})$
- **Part 2d.** computing the new approximate slack matrix $\widetilde{S}(y_{\mathrm{new}})$

**Part 2a.** The $i$'th coordinate of the gradient is expressed as $g_\eta(y)_i = \eta b_i - \mathrm{tr}[S^{-1}A_i]$. The cost per iteration of computing this quantity equals $O(\mathrm{nnz}(A) + n^{\omega+o(1)})$, where the second term comes from inverting the matrix $S$.

**Part 2b.** The cost of inverting the approximate Hessian $\widetilde{H}$ is $O(m^{\omega+o(1)})$ per iteration.

**Part 2c.** The cost of updating the dual variable $y_{\mathrm{new}} = y - \widetilde{H}^{-1} g_{\eta_{\mathrm{new}}}(y)$, given $\widetilde{H}^{-1}$ and $g_{\eta_{\mathrm{new}}}(y)$, is $O(m^2)$ per iteration. The cost of computing the new slack matrix $S_{\mathrm{new}} = \sum_{i\in[m]} (y_{\mathrm{new}})_i A_i - C$ is $O(\mathrm{nnz}(A))$ per iteration.

**Part 2d.** The per iteration cost of updating the approximate slack matrix $\widetilde{S}_{\mathrm{new}}$ is $O(n^{\omega+o(1)})$ by Lemma 5.2.

**Combining Part 2a, 2b, 2c and 2d.**

The total cost of operations other than computing the Hessian over the $T = \widetilde{O}(\sqrt{n})$ iterations is therefore bounded by

$$\begin{aligned} \mathcal{T}_{\text{part 2}} &\leq \mathcal{T}_{\text{part 2a}} + \mathcal{T}_{\text{part 2b}} + \mathcal{T}_{\text{part 2c}} + \mathcal{T}_{\text{part 2d}} \\ &\leq O^*(\sqrt{n}(\mathrm{nnz}(A) + \max(m,n)^\omega)). \end{aligned} \tag{58}$$

**Combining Part 1 and Part 2.**

Combining (57) and (58) and using $r_0 = n$ finishes the proof of the lemma.

$$\begin{aligned} \mathcal{T}_{\text{total}} &\leq \mathcal{T}_{\text{part 1}} + \mathcal{T}_{\text{part 2}} \\ &\leq O^* \left( \min\left(n \cdot \mathrm{nnz}(A), mn^{2.5}\right) + \sqrt{n}\max(m,n)^\omega + \sum_{i=0}^{T} \left(\mathcal{T}_{\mathrm{mat}}(n, mr_i, n) + \mathcal{T}_{\mathrm{mat}}(m, nr_i, m)\right) \right). \end{aligned}$$

□

**Lemma 7.4.** *Let $\mathcal{T}_{\mathrm{mat}}$ be as defined in Definition 3.1. Let $T = \widetilde{O}(\sqrt{n})$ and $\{r_1, \cdots, r_T\}$ be a sequence that satisfies*

$$\sum_{i=1}^{T} \sqrt{r_i} \leq O(T \log^{1.5} n)$$

*Property I. We have*

$$\sum_{i=1}^{T} \mathcal{T}_{\mathrm{mat}}(m, nr_i, m) \leq O^*(\sqrt{n}\max(m^\omega, n^\omega) + \mathcal{T}_{\mathrm{mat}}(m, n^2, m)),$$

*Property II. We have*

$$\sum_{i=1}^{T} \mathcal{T}_{\mathrm{mat}}(n, mr_i, n) \leq O^*(\sqrt{n}\max(m^\omega, n^\omega) + \mathcal{T}_{\mathrm{mat}}(n, mn, n)).$$

*Proof.* We give only the proof of Property I, as the proof of Property II is similar. Let $m = n^a$. For each $i \in [T]$, let $r_i = n^{b_i}$, where $b_i \in [0,1]$. Then

$$\mathcal{T}_{\text{mat}}(m, nr_i, m) = \mathcal{T}_{\text{mat}}(n^a, n^{1+b_i}, n^a) = n^{a\omega((1+b_i)/a)+o(1)}. \tag{59}$$

For each number $k \in \{0, 1, \cdots, \log n\}$, define the set of iterations

$$I_k = \{i \in [T] \ : \ 2^k \leq r_i \leq 2^{k+1}\}.$$

Then our assumption on the sequence $\{r_1, \cdots, r_T\}$ can be expressed as $\sum_{k=0}^{\log n} |I_k| \cdot 2^{k/2} \leq O(T \log^{1.5} n)$. This implies that for each $k\{0, 1, \cdots, \log n\}$, we have $|I_k| \leq O(T \log^{1.5} n/2^{k/2})$. Next, taking the summation of Eq. (59) over all $i \in [T]$, we have

$$\begin{aligned}
\sum_{i=1}^{T} \mathcal{T}_{\text{mat}}(m, nr_i, m) &= \sum_{i=1}^{T} n^{a\cdot\omega((1+b_i)/a)} \\
&= \sum_{k=0}^{\log n} \sum_{i \in I_k} n^{a\cdot\omega((1+b_i)/a)} \\
&\leq O(\log n) \cdot \max_k \max_{i \in I_k} \frac{T \log^{1.5} n}{2^{k/2}} \cdot n^{a\cdot\omega((1+b_i)/a)} \\
&\leq \widetilde{O}(1) \cdot \max_k \max_{2^k \leq n^{b_i} \leq 2^{k+1}} \frac{\sqrt{n}}{2^{k/2}} \cdot n^{a\cdot\omega((1+b_i)/a)} \\
&\leq \widetilde{O}(1) \cdot \max_{b_i \in [0,1]} n^{1/2 - b_i/2 + a\cdot\omega((1+b_i)/a)},
\end{aligned}$$

where the fourth step follows from $T = \widetilde{O}(\sqrt{n})$. To bound the exponent on $n$ above, we define the function $g$,

$$g(b_i) = 1/2 - b_i/2 + a \cdot \omega((1 + b_i)/a). \tag{60}$$

This function is convex in $b_i$ due to the convexity of the function $\omega$ (Lemma 3.6). Therefore, over the interval $b_i \in [0, 1]$, the maximum of $g$ is attained at one of the end points. We simply evaluate this function at the end points.

**Case 1.** Consider the case $b_i = 0$. In this case, we have $g(0) = 1/2 + a\omega(1/a)$. We consider the following two subcases. **Case 1a.** If $a \geq 1$, then we have

$$g(0) = 1/2 + a \cdot \omega(1/a) \leq 1/2 + a\omega(1) = 1/2 + a\omega$$

**Case 1b.** If $a \in (0, 1)$, then we define $k = 1/a > 1$. It follows from Lemma 3.5 and $\omega > 1$, that

$$g(0) = 1/2 + a \cdot \omega(1/a) = 1/2 + \omega(k)/k \leq 1/2 + (k - 1 + \omega)/k \leq 1/2 + \omega.$$

Combining both Case 1a and Case 1b, we have that

$$n^{g(0)} \leq \max(n^{1/2+a\omega}, n^{1/2+\omega}) \leq \sqrt{n} \cdot \max(m^\omega, n^\omega).$$

**Case 2** Consider the other case of $b_i = 1$. In this case, $g(1) = 1/2 - 1/2 + a\omega(2/a) = a\omega(2/a)$. We now finish the proof by combining Case 1 and Case 2 as follows.

$$\max_{b_i \in [0,1]} n^{1/2 - b_i + a\cdot\omega((1+b_i)/a)} \leq \sqrt{n} \max(m^\omega, n^\omega) + n^{a\cdot\omega(2/a)}.$$

□

*Proof of Theorem 7.1.* In light of Lemma 7.4, the upper bound on runtime given in Lemma 7.2 can be written as

$$\mathcal{T}_{\text{Total}} \leq O^*\left(\min\left\{n \cdot \text{nnz}(A), mn^{2.5}\right\} + \sqrt{n}\max(m,n)^\omega + \mathcal{T}_{\text{mat}}(n, mn, n) + \mathcal{T}_{\text{mat}}(m, n^2, m)\right). \quad (61)$$

Combining this with 3.10, we have the following upper bound on the total runtime of Algorithm 1:

$$\begin{aligned}\mathcal{T}_{\text{Total}} &\leq O^*\left(\min\left\{n \cdot \text{nnz}(A), mn^{2.5}\right\} + \sqrt{n}\max(m,n)^\omega + \sqrt{n}\left(mn^2 + m^\omega\right)\right) \\ &\leq O^*\left(\sqrt{n}\left(mn^2 + \max(m,n)^\omega\right)\right).\end{aligned}$$

This finishes the proof of the theorem. □

# 8 Comparison with Cutting Plane Method

In this section, we prove Theorem 1.3, restated below.

**Theorem 1.3** (Comparison with Cutting Plane Method)**.** *When $m \geq n$, there is an interior point method that solves an SDP with $n \times n$ matrices, $m$ constraints, and $\text{nnz}(A)$ input size, faster than the current best cutting plane method [LSW15, JLSW20], over all regimes of $\text{nnz}(A)$.*

**Remark 8.1.** *In the dense case with $\text{nnz}(A) = \Theta(mn^2)$, Algorithm 1 is faster than the cutting plane method whenever $m \geq \sqrt{n}$.*

*Proof of Theorem 1.3.* Recall that the current best runtime of the cutting plane method for solving an SDP (1) is $\mathcal{T}_{\text{CP}} = O^*(m \cdot \text{nnz}(A) + mn^{2.372927} + m^3)$ [LSW15, JLSW20], where 2.372927 is the current best upper bound on the exponent of matrix multiplication $\omega$. By Lemma 7.2 and 7.4, we have the following upper bound on the total runtime of Algorithm 1:

$$\mathcal{T}_{\text{Total}} \leq O^*\left(\min\left\{n \cdot \text{nnz}(A), mn^{2.5}\right\} + \sqrt{n}\max(m,n)^\omega + \mathcal{T}_{\text{mat}}(n, mn, n) + \mathcal{T}_{\text{mat}}(m, n^2, m)\right)$$

Since $m \geq n$ by assumption, Lemma 3.9 and 3.9 further simplify the runtime to

$$\mathcal{T}_{\text{Total}} \leq O^*\left(\min\left\{n \cdot \text{nnz}(A), mn^{2.5}\right\} + \sqrt{n}m^\omega + \mathcal{T}_{\text{mat}}(m, n^2, m)\right) \quad (62)$$

Note that $\min\left\{n \cdot \text{nnz}(A), mn^{2.5}\right\} \leq m \cdot \text{nnz}(A) \leq O(\mathcal{T}_{\text{CP}})$ and that $\sqrt{n}m^\omega = o(m^3) \leq o(\mathcal{T}_{\text{CP}})$ since $m \geq n$. Furthermore, Lemma 3.11 states that $\mathcal{T}_{\text{mat}}(m, n^2, m) = o(m^3 + mn^{2.37}) \leq o(\mathcal{T}_{\text{CP}})$. Since each term on the RHS of (62) is upper bounded by $\mathcal{T}_{\text{CP}}$, we make the stated conclusion. □

# 9 Initialization

**Lemma 9.1** (Initialization)**.** *Consider a semidefinite program as in Definition 1.1 of dimension $n \times n$ with $m$ constraints, and assume that it has the following properties.*

1. *Bounded diameter: for any $X \succeq 0$ with $\langle A_i, X\rangle = b_i$ for all $i \in [m]$, we have $\|X\|_{\text{op}} \leq R$.*
2. *Lipschitz objective: $\|C\|_{\text{op}} \leq L$.*

*For any $0 < \delta \leq 1$, the following* modified *semidefinite program*

$$\begin{aligned} \max_{\overline{X} \succeq 0} \ & \langle \overline{C}, \overline{X} \rangle \\ \text{s.t.} \ & \langle \overline{A}_i, \overline{X} \rangle = \overline{b}_i, \forall i \in [m+1], \end{aligned}$$

*where*

$$\overline{A}_i = \begin{bmatrix} A_i & 0_n & 0_n \\ 0_n^\top & 0 & 0 \\ 0_n^\top & 0 & \frac{b_i}{R} - \mathrm{tr}[A_i] \end{bmatrix}, \quad \forall i \in [m],$$

$$\overline{A}_{m+1} = \begin{bmatrix} I_n & 0_n & 0_n \\ 0_n^\top & 1 & 0 \\ 0_n^\top & 0 & 0 \end{bmatrix} \ , \ \overline{b} = \begin{bmatrix} \frac{1}{R} b \\ n+1 \end{bmatrix} \ , \ \overline{C} = \begin{bmatrix} C \cdot \frac{\delta}{L} & 0_n & 0_n \\ 0_n^\top & 0 & 0 \\ 0_n^\top & 0 & -1 \end{bmatrix},$$

*satisfies the following statements.*

1. *The following are feasible primal and dual solutions:*

$$\overline{X} = I_{n+2} \ , \ \overline{y} = \begin{bmatrix} 0_m \\ 1 \end{bmatrix} \ , \ \overline{S} = \begin{bmatrix} I_n - C \cdot \frac{\delta}{L} & 0_n & 0 \\ 0_n^\top & 1 & 0 \\ 0_n^\top & 0 & 1 \end{bmatrix}.$$

2. *For any feasible primal and dual solutions $(\overline{X}, \overline{y}, \overline{S})$ with duality gap at most $\delta^2$, the matrix $\widehat{X} = R \cdot \overline{X}_{[n] \times [n]}$, where $\overline{X}_{[n] \times [n]}$ is the top-left $n \times n$ block submatrix of $\overline{X}$, is an approximate solution to the original semidefinite program in the following sense:*

$$\begin{aligned} \langle C, \widehat{X} \rangle &\geq \langle C, X^* \rangle - LR \cdot \delta, \\ \widehat{X} &\succeq 0, \\ \sum_{i \in [m]} \left| \langle A_i, \widehat{X} \rangle - b_i \right| &\leq 4n\delta \cdot (R \sum_{i \in [m]} \|A_i\|_1 + \|b\|_1), \end{aligned}$$

*where $X^*$ is any optimal solution to the original SDP and $\|A\|_1$ denotes the Schatten 1-norm of a matrix $A$.*

*Proof.* For the first result, straightforward calculations show that $\langle \overline{A}_i, \overline{X} \rangle = \overline{b}_i$ for all $i \in [m+1]$, and that $\sum_{i \in [m+1]} \overline{y}_i \overline{A}_i - \overline{S} = \overline{C}$. Now we prove the second result. Denote $\mathsf{OPT}$ and $\overline{\mathsf{OPT}}$ the optimal values of the original and modified SDP respectively. Our first goal is to establish a lower bound for $\overline{\mathsf{OPT}}$ in terms of $\mathsf{OPT}$. For any optimal solution $X \in \mathbb{S}^{n \times n}$ of the original SDP, consider the following matrix $\overline{X} \in \mathbb{R}^{(n+2) \times (n+2)}$

$$\overline{X} = \begin{bmatrix} \frac{1}{R} X & 0_n & 0_n \\ 0_n^\top & n + 1 - \frac{1}{R} \mathrm{tr}[X] & 0 \\ 0_n^\top & 0 & 0 \end{bmatrix}.$$

Notice that $\overline{X}$ is a feasible primal solution to the modified SDP, and that

$$\overline{\mathsf{OPT}} \geq \langle \overline{C}, \overline{X} \rangle = \frac{\delta}{LR} \cdot \langle C, X \rangle = \frac{\delta}{LR} \cdot \mathsf{OPT},$$

where the first step follows because the modified SDP is a maximization problem, and the final step is because $X$ is an optimal solution to the original SDP.

Given a feasible primal solution $\overline{X} \in \mathbb{R}^{(n+2)\times(n+2)}$ of the modified SDP with duality gap $\delta^2$, we could assume $\overline{X} = \begin{bmatrix} \overline{X}_{[n]\times[n]} & 0_n & 0_n \\ 0_n^\top & \tau & 0 \\ 0_n^\top & 0 & \theta \end{bmatrix}$ without loss of generality, where $\tau, \theta \geq 0$. This is because if the entries of $\overline{X}$ other than the diagonal and the top-left $n \times n$ block are not 0, then we could zero these entries out and the matrix remains feasible and positive semidefinite. We thus immediately have $\widehat{X} \succeq 0$. Notice that

$$\frac{\delta}{L} \cdot \langle C, \overline{X}_{[n]\times[n]} \rangle - \theta = \langle \overline{C}, \overline{X} \rangle \geq \overline{\mathsf{OPT}} - \delta^2 \geq \frac{\delta}{LR} \cdot \mathsf{OPT} - \delta^2. \tag{63}$$

Therefore, we can lower bound the objective value for $\overline{X}_{[n]\times[n]}$ in the original SDP as

$$\langle C, \widehat{X} \rangle = R \cdot \langle C, \overline{X}_{[n]\times[n]} \rangle \geq \mathsf{OPT} - LR \cdot \delta,$$

where the last inequality follows from (63). By matrix Hölder inequality, we have

$$\begin{aligned} \frac{\delta}{L} \cdot \langle C, \overline{X}_{[n]\times[n]} \rangle &\leq \frac{\delta}{L} \cdot \|C\|_{\mathrm{op}} \cdot \mathrm{tr}\left[\overline{X}_{[n]\times[n]}\right] \\ &\leq \frac{\delta}{L} \cdot \|C\|_{\mathrm{op}} \cdot \langle \overline{A}_{m+1}, \overline{X} \rangle \\ &\leq (n+1)\delta, \end{aligned}$$

where in the last step follows from $\|C\|_{\mathrm{op}} \leq L$ and $b_{m+1} = n + 1$. We can thus upper bound $\theta$ as

$$\theta \leq \frac{\delta}{L} \cdot \langle C, \overline{X}_{[n]\times[n]} \rangle + \delta^2 - \frac{\delta}{LR} \cdot \mathsf{OPT} \leq (2n+1)\delta + \delta^2 \leq 4n\delta, \tag{64}$$

where the first step follows from (63), the second step follows from $\mathsf{OPT} \geq -\|C\|_{\mathrm{op}} \cdot \|X^*\|_1 \geq -nLR$ where $\|\cdot\|_1$ is the Schatten 1-norm, and the last step follows from $\delta \leq 1 \leq n$. Notice that by the feasiblity of $\overline{X}$ for the modified SDP, we have

$$\langle A_i, \overline{X}_{[n]\times[n]} \rangle + (\frac{1}{R} \cdot b_i - \mathrm{tr}[A_i])\theta = \frac{1}{R} \cdot b_i.$$

This implies that

$$\left| \langle A_i, \widehat{X} \rangle - b_i \right| = |(b_i - R \cdot \mathrm{tr}[A_i])\theta| \leq 4n\delta \cdot (R\,\|A_i\|_1 + |b_i|),$$

where the final step follows from the upper bound of $\theta$ in (64). Summing the above inequality up over all $i \in [m]$ finishes the proof of the lemma. □

## Acknowledgment

We thank Aaron Sidford for many helpful discussions and Deeksha Adil, Sally Dong, Sandy Kaplan, and Kevin Tian for useful feedback on the writing. We gratefully acknowledge funding from CCF-1749609, CCF-1740551, DMS-1839116, Microsoft Research Faculty Fellowship, and Sloan Research Fellowship. Zhao Song is partially supported by Ma Huateng Foundation, Schmidt Foundation, Simons Foundation, NSF, DARPA/SRC, Google and Amazon.

## References


[AK07] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, 2007.

[ALO16] Zeyuan Allen Zhu, Yin Tat Lee, and Lorenzo Orecchia. Using optimization to obtain a width-independent, parallel, simpler, and faster positive SDP solver. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms(SODA)*, 2016.

[Ans00] Kurt M Anstreicher. The volumetric barrier for semidefinite programming. *Mathematics of Operations Research*, 2000.

[ARV09] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 2009.

[AV95] David S Atkinson and Pravin M Vaidya. A cutting plane algorithm for convex programming that uses analytic centers. *Mathematical Programming*, 69(1-3):1–43, 1995.

[AZL17] Zeyuan Allen-Zhu and Yuanzhi Li. Follow the compressed leader: faster online learning of eigenvectors and faster mmwu. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017.

[Ban19] Nikhil Bansal. On a generalization of iterated and randomized rounding. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2019.

[BCS97] Peter Bürgisser, Michael Clausen, and Mohammad A Shokrollahi. *Algebraic complexity theory*, volume 315. Springer Science & Business Media, 1997.

[BDG16] Nikhil Bansal, Daniel Dadush, and Shashwat Garg. An algorithm for komlós conjecture matching banaszczyk. In *57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2016.

[BG17] Nikhil Bansal and Shashwat Garg. Algorithmic discrepancy beyond partial coloring. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2017.

[Blä13] Markus Bläser. Fast matrix multiplication. *Theory of Computing*, pages 1–60, 2013.

[BLSS20] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2020.

[Bra20] Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2020.

[BV02] Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (STOC)*, pages 109–115. ACM, 2002.

[CDG19] Yu Cheng, Ilias Diakonikolas, and Rong Ge. High-dimensional robust mean estimation in nearly-linear time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2019.

[CDGW19] Yu Cheng, Ilias Diakonikolas, Rong Ge, and David Woodruff. Faster algorithms for high-dimensional robust covariance estimation. In *Conference on Learning Theory (COLT)*, 2019.

[CDST19] Yair Carmon, John C. Duchi, Aaron Sidford, and Kevin Tian. A rank-1 sketch for matrix multiplicative weights. In *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, pages 589–623, 2019.

[CG18] Yu Cheng and Rong Ge. Non-convex matrix completion against a semi-random adversary. In *Conference On Learning Theory (COLT)*, 2018.

[CLS19] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, 2019.

[Dan47] George B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, 13:339–347, 1947.

[Eld13] Ronen Eldan. Thin shell implies spectral gap up to polylog via a stochastic localization scheme. *Geometric and Functional Analysis*, 2013.

[GH16] Dan Garber and Elad Hazan. Sublinear time algorithms for approximate semidefinite programming. *Mathematical Programming*, 158(1-2):329–361, 2016.

[GLS81] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1981.

[GU18] François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, 2018.

[GV02] Jean-Louis Goffin and Jean-Philippe Vial. Convex nondifferentiable optimization: A survey focused on the analytic center cutting plane method. *Optimization methods and software*, 2002.

[GW95] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 1995.

[HJ12] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, NY, USA, 2nd edition, 2012.

[HW53] A. J. Hoffman and H. W. Wielandt. The variation of the spectrum of a normal matrix. *Duke Math. J.*, 20(1):37–39, 03 1953.

[JJUW11] Rahul Jain, Zhengfeng Ji, Sarvagya Upadhyay, and John Watrous. QIP = PSPACE. *Journal of the ACM (JACM)*, 2011.

[JLL+20] Arun Jambulapati, Yin Tat Lee, Jerry Li, Swati Padmanabhan, and Kevin Tian. Positive semidefinite programming: mixed, parallel, and width-independent. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*. ACM, 2020.

[JLSW20] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games and its applications. In *STOC*, 2020.

[JY11] Rahul Jain and Penghui Yao. A parallel approximation algorithm for positive semidefinite programming. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2011.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing (STOC)*, 1984.

[Kha80] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[KM03] Kartik Krishnan and John E Mitchell. Properties of a cutting plane method for semidefinite programming. *submitted for publication*, 2003.

[KMS94] David Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. In *Proceedings 35th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1994.

[KTE88] Leonid G Khachiyan, Sergei Pavlovich Tarasov, and I. I. Erlikh. The method of inscribed ellipsoids. In *Soviet Math. Dokl*, volume 37, pages 226–230, 1988.

[Lee16] Yin Tat Lee. *Faster algorithms for convex and combinatorial optimization*. PhD thesis, Massachusetts Institute of Technology, 2016.

[LP20] Yin Tat Lee and Swati Padmanabhan. An $\widetilde\mathcalo(m/\varepsilonˆ3.5)$-cost algorithm for semidefinite programs with diagonal constraints. In Jacob D. Abernethy and Shivani Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, Proceedings of Machine Learning Research. PMLR, 2020.

[LS14] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $O(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2014.

[LS15] Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2015.

[LSW15] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2015.

[NN89] Yurii Nesterov and Arkadi Nemirovski. Self-concordant functions and polynomial time methods in convex programming. preprint, central economic & mathematical institute, ussr acad. *Sci. Moscow, USSR*, 1989.

[NN92] Yurii Nesterov and Arkadi Nemirovski. Conic formulation of a convex programming problem and duality. *Optimization Methods and Software*, 1(2):95–115, 1992.

[NN94] Yurii Nesterov and Arkadi Nemirovski. *Interior-point polynomial algorithms in convex programming*, volume 13. Siam, 1994.

[PT12] Richard Peng and Kanat Tangwongsan. Faster and simpler width-independent parallel algorithms for positive semidefinite programming. In *Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures (SPAA)*, pages 101–108, 2012.

[Ren01] James Renegar. *A Mathematical View of Interior-point Methods in Convex Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[Sho77] Naum Z Shor. Cut-off method with space extension in convex programming problems. *Cybernetics and systems analysis*, 13(1):94–96, 1977.

[Sid15] Aaron Daniel Sidford. *Iterative methods, combinatorial optimization, and linear programming beyond the universal barrier*. PhD thesis, Massachusetts Institute of Technology, 2015.

[Str91] Volker Strassen. Degeneration and complexity of bilinear maps: some asymptotic spectra. *J. reine angew. Math*, 413:127–180, 1991.

[Vai87] Pravin M Vaidya. An algorithm for linear programming which requires $o(((m+n)n^2 + (m+n)^{1.5}n)l)$ arithmetic operations. In *28th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 1987.

[Vai89a] Pravin M Vaidya. A new algorithm for minimizing convex functions over convex sets. In *30th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 338–343, 1989.

[Vai89b] Pravin M Vaidya. Speeding-up linear programming using fast matrix multiplication. In *30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 332–337. IEEE, 1989.

[VB96] Lieven Vandenberghe and Stephen P. Boyd. Semidefinite programming. *SIAM Review*, 1996.

[Woo49] Max A Woodbury. The stability of out-input matrices. *Chicago, IL*, 9, 1949.

[Woo50] Max A Woodbury. Inverting modified matrices. 1950.

[YN76] David B Yudin and Arkadi S Nemirovski. Evaluation of the information complexity of mathematical programming problems. *Ekonomika i Matematicheskie Metody*, 12:128–142, 1976.

[YTF+19] Alp Yurtsever, Joel A. Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming, 2019.

# A Matrix Multiplication: A Tensor Approach

The main goal of this section is to rederive, using tensors, some of the technical results from Section 3. In particular, we use tensors to derive upper bounds on the time to perform the following two rectangular matrix multiplication tasks (Lemma A.12 and A.13):

- Multiplying a matrix of dimensions $m \times n^2$ with one of dimensions $n^2 \times m$,
- Multiplying a matrix of dimensions $n \times mn$ with one of dimensions $mn \times n$.

Our hope is that these techniques will eventually be useful in further improving the results of this paper.

## A.1 Exponent of matrix multiplication

We recall two definitions to describe the cost of certain fundamental matrix operations, along with their properties.

**Definition A.1.** *Define $\mathcal{T}_{\text{mat}}(n, r, m)$ to be the number of operations needed to compute the product of matrices of dimensions $n \times r$ and $r \times m$.*

**Definition A.2.** *We define the function $\omega(k)$ to be the minimum value such that $\mathcal{T}_{\text{mat}}(n, n^k, n) = n^{\omega(k)+o(1)}$. We overload notation and use $\omega$ to denote the exponent of matrix multiplication (in other words, the cost of multiplying two $n \times n$ matrices is $n^\omega$), and let $\alpha$ denote the dual exponent of matrix multiplication. Thus, we have $\omega(1) = \omega$ and $\omega(\alpha) = 2$.*

**Lemma A.3** ([GU18])**.** *We have :*

- $\omega = \omega(1) \leq 2.372927$,
- $\omega(1.5) \leq 2.79654$,
- $\omega(1.75) \leq 3.02159$,
- $\omega(2) \leq 3.251640$.

**Lemma A.4** ([BCS97, Blä13])**.** *For any three positive integers $n, m, r$, we have*

$$\mathcal{T}_{\text{mat}}(n, r, m) = O(\mathcal{T}_{\text{mat}}(n, m, r)) = O(\mathcal{T}_{\text{mat}}(m, n, r)).$$

## A.2 Matrix multiplication tensor

The rank of a tensor $T$, denoted as $R(T)$, is the minimum number of simple tensors that sum up to $T$. For any two tensors $S = (S_{i,j,k})_{i,j,k}$ and $T = (T_{a,b,c})_{a,b,c}$, we write $S \leq T$ if there exist three matrices $A, B$ and $C$ (of appropriate sizes) such that $S_{i,j,k} = \sum_{a,b,c} A_{i,a} B_{j,b} C_{k,c} T_{a,b,c}$ for all $i, j, k$. For any $i, j, k$, denote $e_{i,j,k}$ the tensor with 1 in the $(i, j, k)$-th entry, and 0 elsewhere.

**Definition A.5** (Matrix-multiplication tensor)**.** *For any three positive integers $a, b, c$, we define*

$$\langle a, b, c \rangle := \sum_{i \in [a]} \sum_{j \in [b]} \sum_{k \in [c]} e_{i(b-1)+j, j(c-1)+k, k(a-1)+i}$$

*to be the matrix-multiplication tensor corresponding to multiplying a matrix of size $a \times b$ with one of size $b \times c$.*

It's not hard to show that for any $n_i$ and $m_i$ where $i = 1, 2, 3$, we have

$$\langle n_1, n_2, n_3 \rangle \otimes \langle m_1, m_2, m_3 \rangle = \langle n_1 m_1, n_2 m_2, n_3 m_3 \rangle.$$

Let $\langle n \rangle = \sum_{i \in [n]} e_{i,i,i}$ be the identity tensor. For any three tensors $S, T_1$ and $T_2$, if $T_1 \leq T_2$, then we have

$$S \otimes T_1 \leq S \otimes T_2.$$

**Lemma A.6** (Monotonicity of tensor rank, [Str91])**.** *Tensor rank is monotone under the relation $\leq$, i.e. if $T_1 \leq T_2$, then we have*

$$R(T_1) \leq R(T_2).$$

**Lemma A.7** (Sub-multiplicity of tensor rank, [Str91])**.** *For any tensors $T_1$ and $T_2$, we have*

$$R(T_1 \otimes T_2) \leq R(T_1) \cdot R(T_2).$$

**Lemma A.8.** *The tensor rank of a matrix multiplication tensor is equal to the cost of multiplying the two correponding sized matrices up to some constant factor, i.e.,*

$$R(\langle a, b, c \rangle) = \Theta(\mathcal{T}_{\mathrm{mat}}(a, b, c)).$$

### A.3 Implication of matrix multiplication technique

**Lemma A.9** (Sub-linearity)**.** *For any $p \geq q \geq 1$, we have*

$$\omega(p) \leq p - q + \omega(q).$$

*Proof.* We have

$$\langle n, n^p, n \rangle = \langle n, n^q, n \rangle \otimes \langle 1, n^{p-q}, 1 \rangle.$$

Applying tensor rank on both sides

$$\begin{aligned} R(\langle n, n^p, n \rangle) &= R(\langle n, n^q, n \rangle \otimes \langle 1, n^{p-q}, 1 \rangle) \\ &\leq R(\langle n, n^q, n \rangle) \cdot R(\langle 1, n^{p-q}, 1 \rangle), \end{aligned}$$

where the last line follows from Lemma A.7. Applying Lemma A.8, we have

$$\mathcal{T}_{\mathrm{mat}}(n, n^p, n) \leq O(1) \cdot \mathcal{T}_{\mathrm{mat}}(n, n^q, n) \cdot n^{p-q}$$

Using the definition of $\omega(p)$, we have

$$n^{\omega(p)+o(1)} \leq O(1) \cdot n^{\omega(q)+o(1)} \cdot n^{p-q}.$$

Comparing the exponent on both sides completes the proof. ∎

The next lemma establishes the convexity of $\omega(k)$ as a function of $k$.

**Lemma A.10** (Convexity of $\omega(k)$)**.** *The fast rectangular matrix multiplication time exponent $\omega(k)$ as defined in Definition A.2 is convex in $k$.*

*Proof.* Let $k = \alpha \cdot p + (1-\alpha) \cdot q$ for $\alpha \in (0,1)$. We have

$$\langle n, n^k, n\rangle = \langle n^\alpha, n^{\alpha\cdot p}, n^\alpha\rangle \otimes \langle n^{1-\alpha}, n^{(1-\alpha)p}, n^{1-\alpha}\rangle.$$

Applying the tensor rank on both sides,

$$\begin{aligned} R(\langle n, n^k, n\rangle) &= R(\langle n^\alpha, n^{\alpha\cdot p}, n^\alpha\rangle \otimes \langle n^{1-\alpha}, n^{(1-\alpha)p}, n^{1-\alpha}\rangle) \\ &\le R(\langle n^\alpha, n^{\alpha\cdot p}, n^\alpha\rangle) \cdot R(\langle n^{1-\alpha}, n^{(1-\alpha)p}, n^{1-\alpha}\rangle), \end{aligned}$$

where the last line follows from Lemma A.7. By Lemma A.8, we have

$$\mathcal{T}_{\mathrm{mat}}(n, n^k, n) \le O(1) \cdot \mathcal{T}_{\mathrm{mat}}(n^\alpha, n^{\alpha p}, n^\alpha) \cdot \mathcal{T}_{\mathrm{mat}}(n^{1-\alpha}, n^{(1-\alpha)p}, n^{1-\alpha})$$

By definition of $\omega(\cdot)$, we have

$$n^{\omega(k)+o(1)} \le O(1) \cdot n^{\alpha\cdot\omega(p)} \cdot n^{(1-\alpha)\omega(1-p)}.$$

By comparing the exponent, we know that

$$\omega(k) \le \alpha \cdot \omega(p) + (1-\alpha) \cdot \omega(1-p).$$

□

**Lemma A.11.** *Let $\mathcal{T}_{\mathrm{mat}}$ be defined as in Definition A.1. Then for any positive integers $a, b, c$ and $k$, we have*

$$\mathcal{T}_{\mathrm{mat}}(a, bk, c) \le O(\mathcal{T}_{\mathrm{mat}}(ak, b, ck)).$$

*Proof.* Notice that

$$\langle 1, k, 1\rangle \le \langle k, 1, k\rangle.$$

Therefore, we have

$$\begin{aligned} \langle a, bk, c\rangle &= \langle a, b, c\rangle \otimes \langle 1, k, 1\rangle \\ &\le \langle a, b, c\rangle \otimes \langle k, 1, k\rangle \\ &= \langle ak, b, ck\rangle. \end{aligned}$$

It then follows from Lemma A.6 that

$$R(\langle a, bk, c\rangle) \le R(\langle ak, b, ck\rangle).$$

Finally, using Lemma A.8 gives

$$\mathcal{T}_{\mathrm{mat}}(a, bk, c) \le O(\mathcal{T}_{\mathrm{mat}}(ak, b, ck)).$$

Thus we complete the proof. □

### A.4 General bound on $\mathcal{T}_{\text{mat}}(n, mn, n)$ and $\mathcal{T}_{\text{mat}}(m, n^2, m)$

**Lemma A.12.** *Let $\mathcal{T}_{\text{mat}}$ be defined as in Definition A.1.*
*If $m \geq n$, then we have*

$$\mathcal{T}_{\text{mat}}(n, mn, n) \leq O(\mathcal{T}_{\text{mat}}(m, n^2, m)).$$

*If $m \leq n$, then we have*

$$\mathcal{T}_{\text{mat}}(m, n^2, m) \leq O(\mathcal{T}_{\text{mat}}(n, mn, n)).$$

*Proof.* We only prove the case of $m \geq n$, as the other case where $m < n$ is similar. This is an immediate consequence of Lemma A.11 by taking $a = c = n$, $b = n^2$, and $k = \lfloor m/n \rfloor$, where $k$ is a positive integer because $m \geq n$. □

In the next lemma, we derive upper bounds on the term $\mathcal{T}_{\text{mat}}(m, n^2, m)$ when $m \geq n$ and $\mathcal{T}_{\text{mat}}(n, mn, n)$ when $m < n$, which is crucial to our runtime analysis.

**Lemma A.13.** *Let $\mathcal{T}_{\text{mat}}$ be defined as in Definition A.1 and $\omega$ be defined as in Definition A.2.*
*Property I. We have*

$$\mathcal{T}_{\text{mat}}(n, mn, n) \leq O(mn^{\omega+o(1)}).$$

*Property II. We have*

$$\mathcal{T}_{\text{mat}}(m, n^2, m) \leq O\left(\sqrt{n}\left(mn^2 + m^{\omega}\right)\right).$$

*Proof.* **Property I.**
Since

$$\langle n, mn, n\rangle = \langle n, n, n\rangle \otimes \langle 1, m, 1\rangle.$$

Applying the tensor rank on both sides, we have

$$\begin{aligned} R(\langle n, mn, n\rangle) &= R(\langle n, n, n\rangle \otimes \langle 1, m, 1\rangle) \\ &\leq R(\langle n, n, n\rangle) \cdot R(\langle 1, m, 1\rangle) \end{aligned}$$

Thus, we complete the proof.
**Property II.**
Let $m = n^a$, where $a \in (0, \infty)$. We have

$$\langle m, n^2, m\rangle = \langle n^a, (n^a)^{2/a}, n^a\rangle$$

It implies that

$$\mathcal{T}_{\text{mat}}(m, n^2, m) = n^{a\cdot\omega(2/a)+o(1)}$$

The Property II is then an immediate consequence of the following inequality, which we prove next:

$$\omega(2/a) < \max(1 + 2.5/a, \omega(1) + 0.5/a) \qquad \forall a \in (0, \infty).$$

Define $b = 2/a \in (0, \infty)$. Then the above desired inequality can be expressed in terms of $b$ as

$$\omega(b) < \max(1 + 5b/4, \omega(1) + b/4) \qquad \forall b \in (0, \infty). \tag{65}$$

Notice that the RHS of (15) is a maximum of two linear functions of $b$ and these intersect at $b^* = \omega(1) - 1$. By the convexity of $\omega(\,\cdot\,)$ as proved in Lemma A.10, it suffices to verify (15) at the endpoints $b \to 0$, $b \to \infty$ and $b = b^*$. In the case where $b = \delta$ for any $\delta < 1$, (15) follows immediately from the observation that $\omega(\delta) < \omega(1)$. For the case $b \to \infty$, by Lemma A.3 we have $\omega(2) \leq 3.252$. It then follows from Lemma A.9 that for any $b > 2$, we have

$$\omega(b) \leq b - 2 + \omega(2) \leq 1 + 5b/4.$$

The final case is where $b = b^* = \omega(1) - 1$, for which (15) is equivalent to

$$\omega(\omega(1) - 1) < 5\omega(1)/4 - 1/4. \tag{66}$$

By Lemma A.3, we have that $\omega(1) - 2 \in [0, 0.372927]$. Then to prove (66), it is sufficient to show that

$$\omega(t + 1) < 5t/4 + 9/4 \qquad \forall t \in [0, 0.372927]. \tag{67}$$

By the convexity of $\omega(\,\cdot\,)$ as proved in Lemma A.10 and the upper bound of $\omega(2) \leq 3.251640$ in Lemma A.3, we have for $k \in [1, 2]$,

$$\omega(k) \leq \omega(1) + (k - 1) \cdot (3.251640 - (t + 2)) = t + 2 + (k - 1) \cdot (1.251640 - t).$$

In particular, using this inequality for $k = t + 1$, we have

$$\begin{aligned} \omega(t + 1) - 5t/4 - 9/4 &\leq (t + 2) + t \cdot (1.251640 - t) - 5t/4 - 9/4 \\ &= -t^2 + 1.00164t - 1/4, \end{aligned}$$

which is negative on the entire interval $[0, 0.372927]$. This establishes (67) and finishes the proof of the lemma. □